Einsteigen - Verstehen - Beherrschen

DM 3,80 ÖS 30 sfr 3,80

computer kurs

Ein wöchentliches Sammelwerk

Heft 65

Zeit für Go

Der Digitalisierarm

Programmhilfen

Des Schneiders neuer Joyce

Programmierkurs COBOL

computer kurs

Heft 65

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.

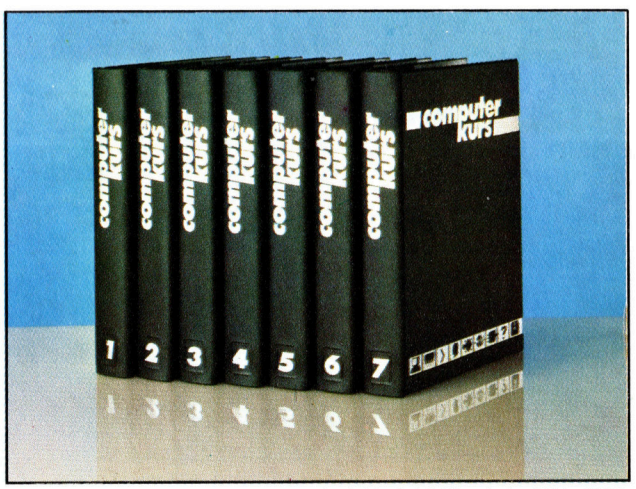
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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

Höret und staunet!

Computer als Musikinstrumente einzusetzen, ist für Amateurmusiker und Profis gleichermaßen aktuell. Hier zeigen wir, wie sich die Verbindungen zwischen Computer und Musik bis heute entwickelt haben.

Mit Einführung der digital-gesteuerten Klangsynthese wurden immer öfter Heimcomputer als programmierbare Musikinstrumente und im Profibereich als elektronische Kontrollinstrumente eingesetzt. Software-Autoren erkannten die Schwierigkeiten, auf die die meisten Möchtegern-Musiker beim direkten Programmieren der SoundChips stoßen, und produzierten anwenderfreundliche Programme. Hier eine Auswahl derzeit erhältlicher Programme.

Seit Einführung der ersten PET und Apple-Modelle Ende der 70er Jahre gibt es Musikprogramme für Microcomputer. Diese Rechner waren noch recht unkomfortabel und in ihren Anwendungsmöglichkeiten begrenzt. Zur gleichen Zeit kamen vergleichsweise einfache tragbare Synthesizer für Bühnengebrauch und Hobbyanwendung auf den Markt. Da westlicher Musik mathematische Gesetze zugrunde liegen, etwa bei der Generierung von Klängen oder bei ihrer rhythmischen Verknüpfung, erkannten viele Musiker bald die im Rechner steckenden Möglichkeiten.

Ursprünglich waren Geräuschkulissen für Computerspiele der Hauptgrund, Microcomputer mit Tonerzeugung auszustatten. Einige Hersteller wie Commodore, Acorn und Amstrad erkannten bald, daß ein Bedarf an echten Musikmöglichkeiten bestand und statteten ihre Rechner mit hochentwickelten Chips zur Klangentwicklung und -formung aus. Andere Hersteller wie Sinclair beließen es bei den „Beep"-Tongeneratoren. So gibt es wenig Musikprogramme für den Spectrum. Aber es wird Hardware-Zubehör angeboten, das die Klangerzeugungsmöglichkeiten verbessert.

Die einfachste Möglichkeit zur Tonerzeugung ist die Programmierung des Sound-Chips. Hersteller, die ihre Geräte mit hochentwickelten Sound-Chips ausstatten, implementieren auch BASIC-Befehle, mit denen der Anwender Tonhöhe und Dauer von Noten bestimmen und den Klang über Hüllkurven-Befehle festlegen kann. Bemerkenswerte Ausnahme dabei ist Commodore: Der zweifellos sehr leistungsfähige Sound-Chip des Computers wird nicht vom BASIC unterstützt. Statt dessen ist eine komplizierte PEEK- und POKE-Programmierung erforderlich.

Die meisten Chips sind dreistimmig angelegt, so daß bis zu drei Noten gleichzeitig gespielt werden können. Akkorde und dreistimmige Stücke lassen sich deshalb leicht erzeugen. Die Lautstärken-Hüllkurven-Steuerung wird gewöhnlich durch einen längeren Zahlenstring definiert, der umgekehrt wiederum Höhe und Schrittfolge jedes Hüllkurven-Teils bestimmt. Dazu stehen häufig verschiedene Wellenformen wie Dreieck und Sägezahn zur Verfügung.

Hauptproblem bei der Programmierung von Musik in BASIC ist die Langsamkeit, gemessen an der erforderlichen Timing-Kontrolle, die für die Erzeugung von natürlich klingenden Tonfolgen erforderlich ist. Selbst das Interrupt-Sound-System des Schneider CPC löst dieses Problem nur zum Teil. Die beste Lösung aus der Sicht des Anwenders ist der Einsatz von Maschinensprache.

Programmpakete lassen sich in zwei Haupt-

gruppen einteilen: Programme, die die Klangerzeugungsmöglichkeiten des Computers nutzen, und Programme, die den Computer zur Steuerung externer Klangerzeuger verwenden. Bestes Beispiel für den zweiten Fall ist MIDI, das „Netzwerk" von rechnergesteuerten Tastaturen. Die erste Kategorie ist in zwei Programmarten zu untergliedern: eine, die zusätzliche Hardware – wie z. B. ein Keyboard – erfordert, und eine auf reiner Software-Basis.

Musik in Echtzeit

Viele Programme verwandeln den Computer zu einem in Real-Time spielbaren Instrument: Ein Druck auf eine Taste wird sofort in einen hörbaren, korrespondierenden Klang umgesetzt. Bei preiswerteren Programmen wird die Tastatur des Computers genutzt. Teurere Ergänzungen aber arbeiten mit einer traditionellen Klaviatur. Die zugehörigen Programme sehen normalerweise auch Möglichkeiten vor, den Sound-Chip zu beeinflussen und so die Klangqualität zu verändern.

Im „Sequenzing" zeigen sich zahllose Möglichkeiten des Microcomputers. So kann man ein Stück schrittweise schreiben und dabei die Standard-Notation oder eine spezielle Musiksprache auf gleiche Art wie ein Textverarbeitungssystem verwenden. Das hat den Vorteil, daß man während des Komponierens das Stück hören und editieren kann. Zudem kann der Computer die Komposition analysieren und ein in Echtzeit gespieltes Stück in notierte Werte umsetzen. Ist man mit dem Ergebnis zufrieden, läßt sich die abgeschlossene Komposition für weitere Verarbeitung speichern und in Standard-Notation über einen normalen Matrix-Drucker ausdrucken. Dazu gibt es keine Parallele in der reinen Synthesizer-Welt, von echten Composern und Sequenzern oder der Verwendung eines Micro mit MIDI (wie Yamahas CX5M) abgesehen.

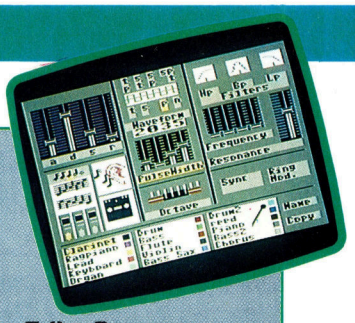
Die meisten Leute, die sich fürs Computern interessieren, werden bei der Programmierung von Musik unterschiedlich Erfolg haben. Musikprogramme erlauben dagegen schrittweises Programmieren von Musik, wobei selbst dem absoluten Laien die Komposition schöner Stücke gelingt. Fortgeschrittene Musiker sind von diesen Programmen begeistert, weil sie Experimente mit der vertrauten Standard-Notation beim Komponieren und weitreichendes Modifizieren erlauben. Systeme wie der CX5M, die Kompositionsmöglichkeiten mit Steuerungsmöglichkeiten externer elektronischer Musikinstrumente miteinander verbinden, maximieren das Potential.

MIDI-Instrumente-Benutzer haben aufgrund der relativ geringen Zusatzkosten für einen Heimcomputer mit Diskettenlaufwerk eine preiswerte Möglichkeit der Mehrfachkontrolle und können bis zu 16 MIDI-Instrumente gleichzeitig steuern.

Komponieren – leicht gemacht

Schrittweises Programmieren erlaubt selbst Anfängern, eigene Kompositionen einzugeben, die dann vom Computer gespielt werden können. Obwohl es ebensoviele Versionen dieser schrittweisen Programmierung wie Musik-Kompositionsprogramme gibt, sind die Grundlagen stets dieselben.

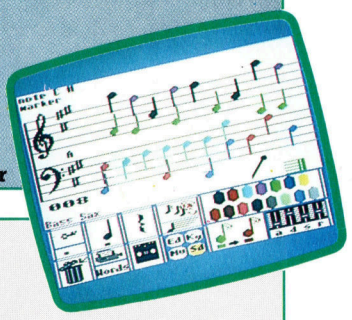
Noten werden über die Tastatur in den Computer eingegeben und auch gleich bei der Eingabe gespielt. Dann erfolgt die Darstellung der Note auf dem Bildschirm, oft in der üblichen Schreibweise mit dem fünflinigen Notensystem. Nach Vollendung des Stücks kann der Anwender seine Komposition überarbeiten, indem er Note für Note überprüft und sie gegebenenfalls verändert. Vorteil dieser Kompositionsmethode: Anwender können während der Noteneingabe hören, was sie spielen und haben eine sofortige Rückmeldung. So hören angehende Komponisten ihre Songs, die sie vielleicht auf einem herkömmlichen Musikinstrument nicht spielen könnten.



Sound Engineer

Tolles Programm
Das Piktogramm-gesteuerte „Music Studio" von Activision erlaubt die Programmierung des Sound-Chips und zugleich die Komposition von Stücken. Mit dem „Sound Engineer" kann man ein vorgegebenes Instrument spielen oder aber ein eigenes programmieren.

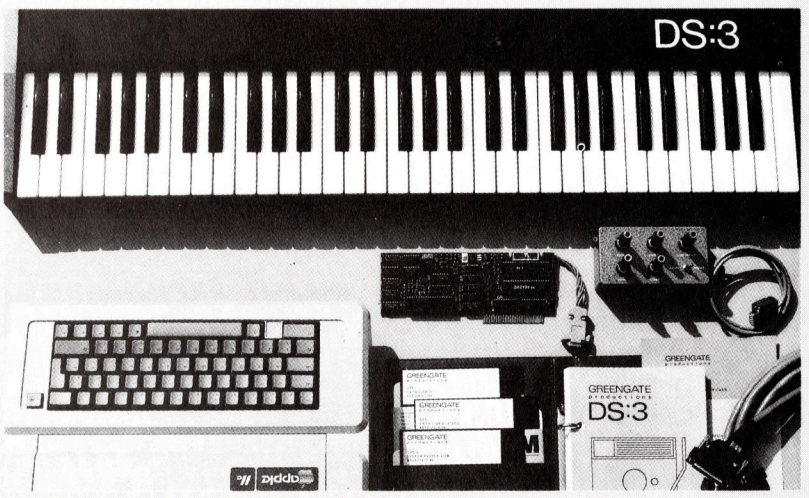
Mit dem „Music Editor" kann man schrittweise Lieder komponieren. Ein „Taktstock"-Cursor wählt die Klangelemente.



Music Editor

Sampling

Sampling (eigentlich: Sammeln) läuft häufig unter dem Begriff der Musiksynthese, obwohl es sich dabei eigentlich um Umwandlung und Speicherung eines Klanges (eines analogen Signals) in eine Reihe digitaler Werte handelt. Sie können in ein dem Original sehr nahes Signal (einen Ton) zum Spielen, Komponieren oder Sequenzen zurückverwandelt werden. Bisher war solche digitale „Aufzeichnung" nur mit sehr teuren Geräten möglich. Jetzt gibt es das preiswerte DS3-System für den Apple II zum vierstimmigen Sampling (polyphon). Monophone Systeme wurden beispielsweise für den Commodore 64 (Autographics Microsound 64) und den Spectrum (Ricoll Sound Sampler und Datel DDS) entwickelt.



Synthetischer Klang

Die ersten Synthesizer waren wie die frühen Computer gigantische Maschinen. Herzstück waren Oszillatoren, die aus einfachen Puls-, Dreieck- oder Sägezahn-Wellen, separat oder in Kombination verwendet, einen oder zwei Klänge erzeugten. Mit zusätzlichen Geräten wurde die Hüllkurve gestaltet, dazu kamen verschiedene Signal-Filter, die die Klangform änderten, indem bestimmte Frequenzen herausgenommen wurden.

Damit waren Synthesizer sehr vielseitig, konnten wegen ihrer Größe aber nur in Tonstudios verwendet werden. Ihre Nutzung als Musikinstrument war zudem sehr begrenzt, da die erzeugten Klänge dünn und farblos wirkten. Diese Geräte waren aber Grundlage für die Entwicklung von Synthesizern mit Tastatur. Man experimentierte mit verschiedenen Steuerungstechniken, so zum Beispiel mit einem Fingersatz, ähnlich wie bei einer Klarinette, und einem Griffsystem, das auf der Unterbrechung farbiger Lichtstrahlen basierte. Am einfachsten aber erwies sich die Umsetzung auf eine Klaviertastatur, auf der eine einzelne Note den Wert „ein" oder „aus" erhielt, ausgelöst durch einen entsprechenden Schalter unter der Taste.

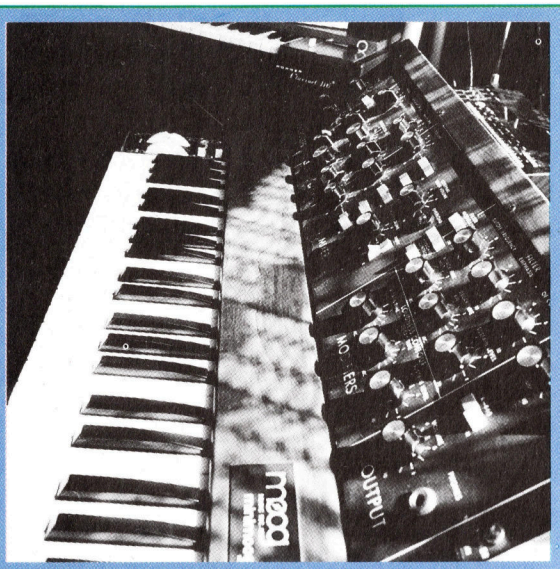
Die ersten tragbaren Instrumente waren monophon und arbeiteten auf den gleichen Grundlagen wie die gewaltigen Studiomaschinen. Durch ICs sparte man Gewicht, Kosten und Volumen. Doch auch diese Geräte waren noch völlig analog.

Anfang der 80er Jahre wurde die Digital-Technik eingeführt, zunächst um die von den Analog-Oszillatoren erzeugten Frequenzen zu stabilisieren, da sie unter starken Schwankungen litten. Wenig später kamen Schaltungen auf den Markt, die Potentiometer- und Schalterstellung speichern konnten, wodurch Parameter für bestimmte Klangformen aufgerufen und jederzeit durch einfachen Tastendruck aktiviert werden konnten.

Wegen der hohen Kosten für die Speicherchips konnten zunächst nur bis zu zwölf Klänge gespeichert werden. Doch in dem Maße, wie sich die Computer-Technologie weiterentwickelte, war bald auch bei den Synthesizern die Speicherung mehrerer hundert „Settings" (= Einstellungen) möglich. Die Massenherstellung der Klangerzeugungs-Chips reduzierte die Kosten ebenfalls.

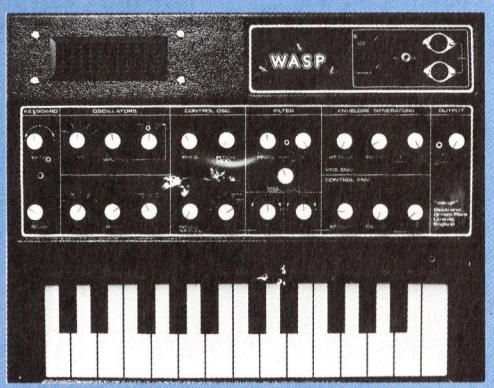
Die Digitalisierung der Parameter-Werte jedes einzelnen Klangerzeugers erlaubte den Herstellern den Verzicht auf herkömmliche Potentiometer, Gleitregler und Schalter. Damit konnte ein bestimmter Parameter „aufgerufen" und sein derzeitiger Wert als LED-Anzeige mit „+" oder „−" für die Wertänderung dargestellt werden. Ende 1982 waren bei fast allen Synthesizern die Klang-Parameter digital definierbar.

Im August 1983 einigte man sich auf den MIDI-Standard. Hierbei handelt es sich eigentlich um einen Software-Standard, wobei MIDI-Übertragungen bestimmte Bedeutungen haben. Das Bindeglied zwischen zwei, durch MIDI spezifierten Instrumenten ist lediglich ein asynchrones, serielles System, das bereits seit mehreren Jahren erfolgreich bei Computern Anwendung findet und die Kommunikation mit Peripheriegeräten erlaubt. Eines der ersten mit MIDI ausgestatteten Geräte war ein tragbarer, völlig digitalisierter Synthesizer – der Yamaha DX7 – ein Computer-System, speziell für Klangerzeugung und Steuerung entwickelt.

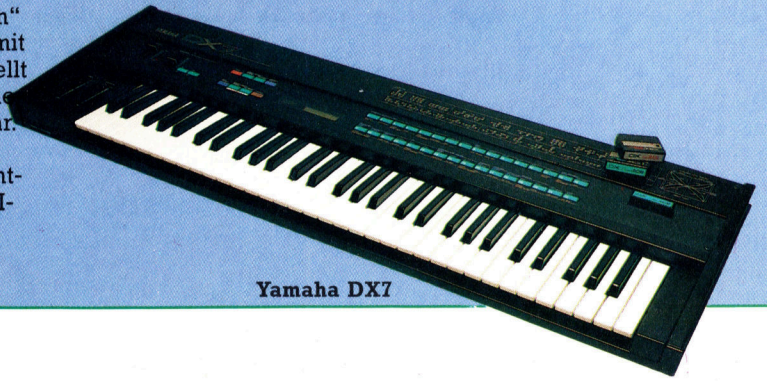


Minisystem
Der hier gezeigte Mini Moog war der erste von vielen kleinen, tragbaren Synthesizern auf IC-Basis.

Kostenreduzierung
Die Entwicklung der Synthesizer und der Heimcomputer weist mehrere Parallelen auf. Der hier gezeigte WASP-Synthesizer drückte die Preise für elektronische Synthesizer so, daß sie für einen größeren Interessentenkreis erschwinglich wurden. Grund: Die kostenintensive mechanische Tastatur wurde durch eine einfache Plastikmembran ersetzt wie beim ZX80 und ZX81, die gleichzeitig auf den Markt kamen.



Die durch MIDI spezifizierten Übertragungsraten sind so festgelegt, daß ein Heimcomputer als Empfänger, Quelle oder zwischengeschalteter Prozessor für MIDI-Bytes wirken kann. Damit ist das Bindeglied zwischen Computertechnik und Synthesizer hergestellt. Programme, die die Umwandlung eines Heimcomputers in ein MIDI-System und seine Nutzung als Hauptkontrolleinheit für perfekte Komposition, Spiel- und als Aufnahmesystem ermöglichen, gibt es reichlich.



Yamaha DX7

Zeit für Go

Neben Schach hat besonders das orientalische Spiel Go das Interesse der KI-Wissenschaftler geweckt. Wir beginnen die Entwicklung eines Go-Programms und erklären zunächst die Regeln.

In unserer Rubrik BASIC wollen wir ein Go-Programm entwickeln. Es stellt eine gute Einführung in das komplexe Spiel dar und dürfte für Anfänger ein interessanter Spielpartner sein. Das Programm kann problemlos modifiziert werden.

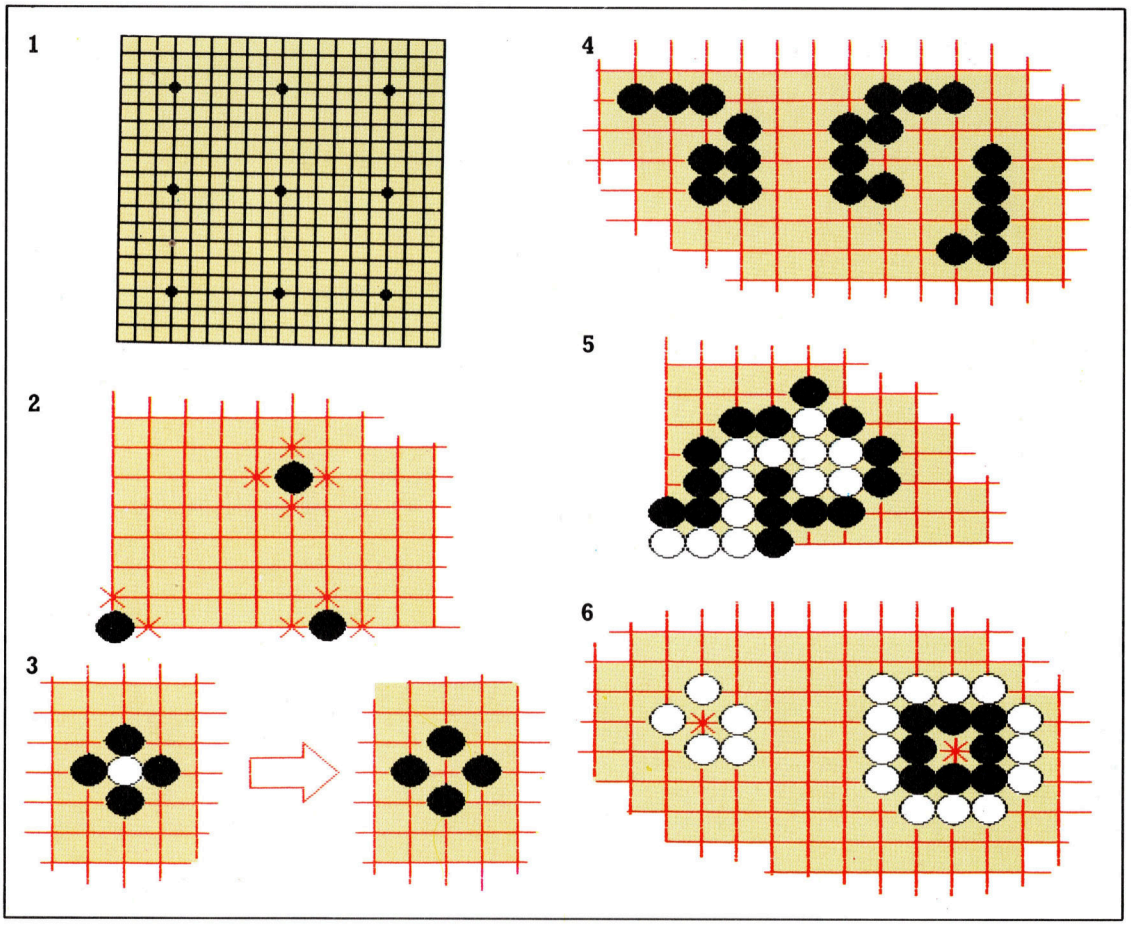
Go ist ein Spiel für zwei Spieler und wird auf einem Brett mit 361 Schnittpunkten gespielt (siehe Bild 1). Die Spieler setzen abwechselnd einen Stein (Spieler 1 = weiß, Spieler 2 = schwarz) auf einen beliebigen Schnittpunkt des Bretts. Beachten Sie, daß die Steine nicht in die Quadrate gesetzt werden.

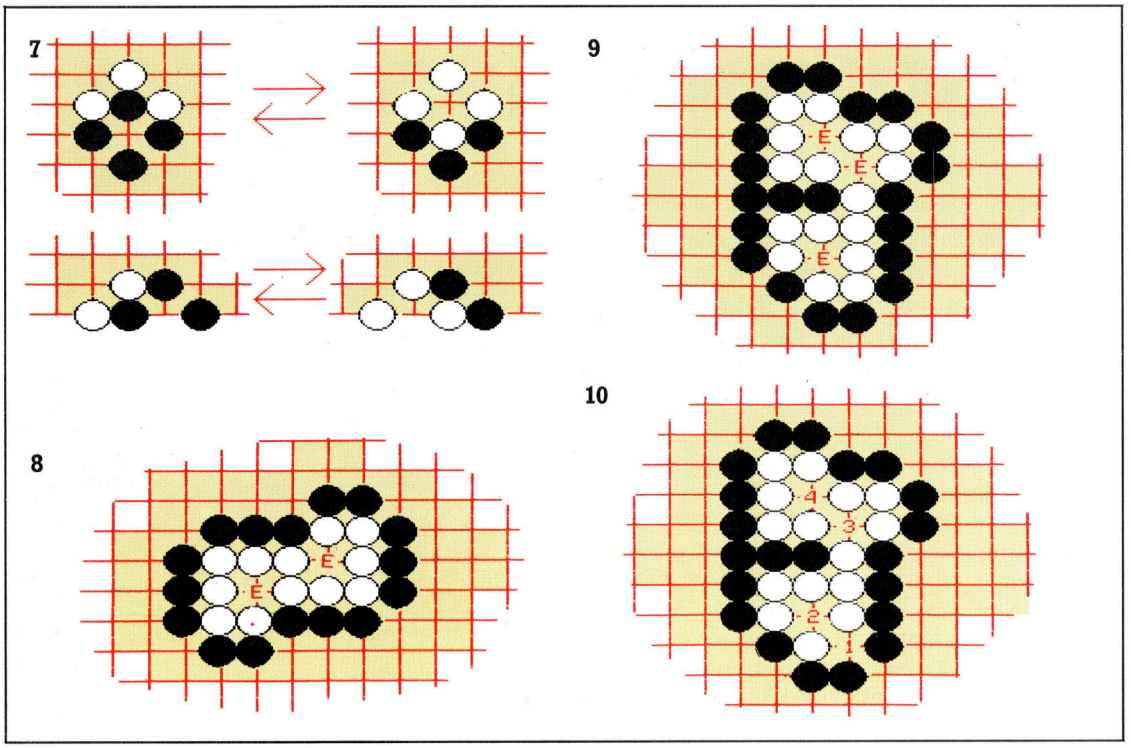
Ziel ist es, mit den eigenen Steinen Gebiete einzuschließen. Gewonnen hat, wer am Ende das größte Gebiet umschlossen hat.

Der erste Zug erfolgt bei Go immer von „Schwarz", dem normalerweise schwächeren Spieler. Variiert die Spielstärke der beiden Kontrahenten zu stark, setzt Schwarz im ersten Zug 2 bis 9 „Handicap"-Steine auf die in Bild 1 markierten Positionen des Spielbrettes.

Nur Gebiete einzuschließen, wäre etwas zu einfach – Steine können auch eingenommen und vom Brett entfernt werden. Jeder freie Schnittpunkt, der direkt über eine Linie mit einem Stein in Verbindung steht, heißt „Freifeld". Ein einzelner Stein kann somit 2, 3 oder 4 Freifelder haben, abhängig von seiner Position auf dem Brett (siehe Bild 2). Zum Einnehmen eines gegnerischen Steines müssen Sie Ihre Steine auf die Freifelder des gegnerischen Steines setzen. Nachdem also in Bild 3 der letzte schwarze Stein auf das letzte Freifeld des weißen Steines gesetzt wurde, kann dieser entfernt werden. Alle gewonnenen Steine werden zur Punktzahl des Spielers addiert.

Sind ein oder mehrere mit einem Stein verbundene Felder mit Steinen derselben Farbe besetzt, spricht man von einer Gruppe. Diagonale Verbindungen haben keine Gültigkeit. In Bild 4 befinden sich vier Gruppen.





Gruppen sind schwieriger einzunehmen als einzelne Steine, da sie als einzelne Einheiten entfernt werden müssen. In Bild 5 muß Schwarz 15 Freifelder besetzen, um die weiße Gruppe zu erobern.

Das ist prinzipiell alles, was Sie zum Spiel wissen müssen, doch gibt es noch einige Sonderregeln:

Selbstmord: Nach den bisherigen Regeln ist es möglich, einen Stein so zu setzen, daß er keine Freifelder hat. Schwarz könnte zum Beispiel auf eine der in Bild 6 markierten Positionen setzen. Ein solcher als Selbstmord bezeichneter Zug ist in Go verboten.

Ko: Dieser Ausdruck benennt die Situation, in der Steine so gesetzt werden, daß Schwarz und Weiß stets abwechselnd einen Stein gewinnen. Betrachten Sie hierzu Bild 7.

Um diese Situation zu verhindern, verbietet die Ko-Regel, einen Stein derart zu setzen. Diese Regel bringt die sogenannten Ko-Kämpfe mit sich. Wenn Weiß aufgrund der Ko-Regel vom direkten Zurückfangen eines Steines abgehalten wird, muß eine Zwischenlösung gewählt werden.

Der Stein wird normalerweise an eine Position gesetzt, die einen schwarzen Stein oder eine Gruppe bedroht. Man nennt das „Sente" (erzwungener Gegenzug). Da Schwarz den Ko-Punkt nicht besetzen kann, hat Weiß nun die Möglichkeit, den schwarzen Ko-Stein zu erobern. Schwarz darf nun auch keinen direkten Gegenzug ausführen und muß einen Sente-Zug finden. Der Ko-Kampf geht weiter.

Ein wichtiges Konzept, das aus diesen Regeln resultiert, ist das von „Leben und Tod" einer Gruppe. Wir haben bereits gesehen, daß Schwarz keinen Stein auf den Schnittpunkt innerhalb der Gruppe setzen darf (Bild 6). Beachten Sie, daß die schwarze Gruppe gefangen würde, wenn Weiß am Zug wäre.

Wäre die schwarze Gruppe nicht von weißen Steinen eingeschlossen, könnte auch Weiß keinen Stein auf diesen Punkt setzen. Will Weiß also die schwarze Gruppe einnehmen, muß der letzte weiße Stein, nachdem alle anderen Freifelder der schwarzen Steine besetzt sind, auf diesen Punkt gesetzt werden. Danach würde die schwarze Gruppe vom Brett entfernt, und der gerade gesetzte weiße Stein hätte vier Freifelder.

Ein so von Steinen derselben Farbe eingeschlossener Punkt heißt „Auge". Ein Auge ist immer das letzte Freifeld, das zum Einnehmen einer Gruppe besetzt werden muß.

Es könnte auch eine Gruppe mit zwei Augen entstehen, entsprechend Bild 8. Zum Erobern der Gruppe müßte der letzte schwarze Stein beide Augen gleichzeitig besetzen, was aber unmöglich ist, da man nur einen Stein pro Zug setzen darf.

Eine Gruppe mit mindestens zwei Augen ist demnach sicher und wird bis zum Spielende auf dem Brett bestehen (vorausgesetzt Weiß füllt nicht irrtümlich die Augen!).

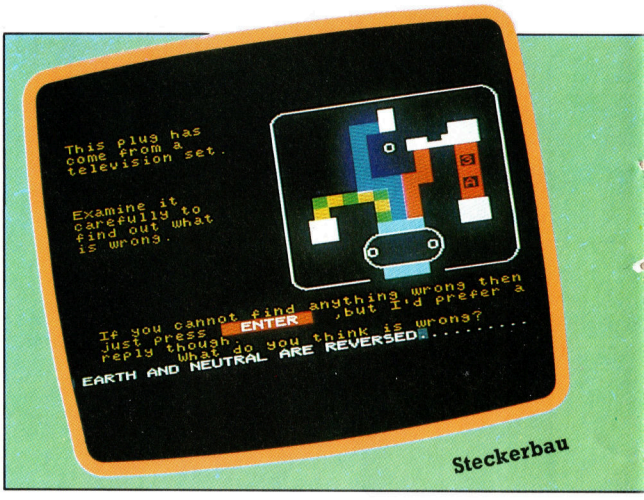
Es ist dabei nicht nötig, daß sich die Augen in einer Gruppe befinden. Die drei weißen Gruppen in Bild 9 teilen sich drei Augen, wodurch alle weißen Steine gesichert sind. Trotzdem ist Vorsicht geboten. Die ähnliche Formation in Bild 10 (nur ein Stein sitzt anders) kann durch Besetzen des untersten Auges („falsches Auge") von Schwarz eingenommen werden.

Das Spiel endet, wenn beide Spieler überzeugt sind, daß auf beiden Seiten nichts mehr zu gewinnen ist.

Wer gewinnt?
Bei Spielende wird die Punktzahl jedes Spielers wie folgt ermittelt:
1. Alle neutralen Punkte werden gefüllt. Das kann ein beliebiger Spieler machen, denn diese Steine werden ohnehin nicht mitgerechnet.
2. Alle Steine oder Gruppen, die eingenommen werden könnten, werden vom Brett entfernt. Es ist also nicht notwendig, diese „toten" Steine während des Spieles zu erobern. Jeder Stein zur Verteidigung erhöht die Punktzahl des angreifenden Spielers um 1, doch muß er auch einen Stein setzen, das Gebiet füllen, und so einen Punkt einbüßen.
3. Die Punktzahl eines Spielers wird nun errechnet, indem die Anzahl der freien von einem Spieler kontrollierten Schnittpunkte gezählt wird, abzüglich der vom Gegner gefangenen Steine. Gewinner ist der Spieler mit den meisten Punkten.

Programmierhilfen

In dieser zweiteiligen Serie untersuchen wir die Eigenschaften von Programmgeneratoren. Wir stellen vier Pakete vor und sehen uns zwei genauer an – „Sycero“ und „The Last One“.



Steckerbau

Ein Programm, das dem Programmierer Arbeit abnimmt und perfekten Code erzeugt, der nicht erst mühsam von Fehlern befreit werden muß, ist der Traum jedes Anfängers (und vermutlich auch der jedes erfahrenen Programmierers).

Eigentlich benutzt jeder BASIC-Programmierer bereits ein programmschreibendes Programm: Der BASIC-Interpreter (ob ROM oder Diskettenversion) nimmt die vom Anwender eingegebenen Programmzeilen und übersetzt sie in den Maschinencode, mit dem der Computer arbeitet. Dieser Prozeß ist weit einfacher als die Eingabe endloser Folgen von Nullen und Einsen. Selbst bei der Assemblerprogrammierung findet ein ähnlicher Vorgang statt, wenn die mnemotischen Assemblerkürzel in den Maschinencode übersetzt werden.

Die meisten Programmierer kennen Tricks, die per Programm weitere Programmzeilen generieren. Sie verwenden entweder den Tastaturbuffer mit einem inkrementierenden Zähler zur Erzeugung von Zeilennummern, oder sie setzen ein Programm mit POKE direkt in den Arbeitsspeicher. Maschinencodeprogrammierer lesen oft auch mit PEEK den Inhalt von Speicherstellen und schreiben die Werte als DATAs in ihre Programme.

Einige Sprachen (zum Beispiel COMAL) enthalten Fehlerprüfroutinen, die fehlerhafte Codezeilen zurückweisen. PRNT statt PRINT wird dabei nicht akzeptiert. FORTH weist Wörter zurück, die nicht im Vokabular der Programmiersprache enthalten sind.

Sie können jedoch Ihren Computer nicht anweisen, ein Programm zur Berechnung der Einkommenssteuer zu schreiben, auch wenn bereits Systeme mit Künstlicher Intelligenz entwickelt wurden, die dieses Ziel in nicht allzu weite Ferne rücken. Das Programm Microtext beispielsweise führt technisch nicht vorgebildete Anwender durch komplizierte Abläufe. Es eignet sich für interaktive Lernhilfen, Fehlersuchprogramme, Fragebögen und Software, die Informationen zusammenstellt. Dabei wird die gewünschte Anwendung mit dem „Autorensystem“ geschrieben, unter einem Bildschirmeditor getestet und die generierte Software in ein Ablaufsystem umgewandelt, das der Anwender nicht mehr verändern kann.

Microtext gab es bisher nur für teure CP/M-Maschinen, bei denen der hochentwickelte Bildschirmeditor seinen großen Wert beweisen kann. Es wird inzwischen aber zu einem günstigen Preis auch für andere Computer wie den Tatung Einstein angeboten.

„The Quill“ ist ein erstaunlich weit entwickelter Programmgenerator, der sich speziell für die Entwicklung von Abenteuerspielen eignet. Sie können darüber eine Datenbank mit Informationen (über Orte, Objekte, die Art ihrer Bewegung, etc.) einrichten, editieren und dann in ein Assemblerprogramm einsetzen.

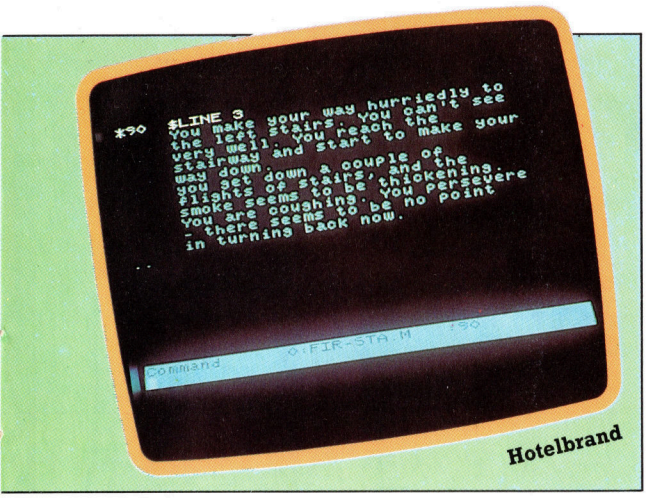
Quill wurde vor kurzem um den „Illustrator“ erweitert, der Textabenteuer durch Grafik bereichert. Quill gibt es für den Sinclair Spectrum, Schneider CPC, Commodore 64 und Oric. Der Illustrator ist für den Spectrum, den Schneider und bald auch für den Commodore verfügbar.

„The Last One“

Obwohl diese beiden Systeme ihren Eigenschaften nach Programmgeneratoren sind, gibt es für die Microindustrie eigentlich nur zwei Programme, die auch wirklich Programme schreiben: Eins hat den pompösen Namen „The Last One“ (das letzte Programm) und sollte gemäß Werbung die letzte Software sein, die Sie je kaufen. Das zweite ist das flexiblere, aber weniger bedienerfreundliche „Sycero“.

Das TLO- (The Last One) Programm wurde ursprünglich für CP/M-Maschinen mit 64 KByte entwickelt. TLO gibt es inzwischen auch für MS-DOS-Maschinen mit mindestens 256 KByte RAM. Dadurch sind zwar viele neue Fähigkeiten hinzugekommen, doch ist der Ablauf immer noch schleppend.

Die Designer von TLO behaupten, in das



Hotelbrand

Programm einige Ebenen Künstlicher Intelligenz eingebaut zu haben. Und so identifiziert das System auch richtig die DOS-Diskette als die Fehlerquelle, wenn es die gesuchte BASIC-Version dort nicht findet.

Obwohl Sycero und The Last One auf den ersten Blick recht ähnlich sind, weisen sie große Unterschiede in der Struktur auf, in der Anwendungsbreite und der Art, wie sie die Anforderungen des Anwenders herausfinden, um brauchbaren (und übertragbaren) Code produzieren zu können. Beide Systeme arbeiten mit BASIC und eignen sich für Eingabe, Verarbeitung, Speicherung, Abruf und Ausdruck von Daten. Im Gegensatz zu Programmen mit Datenbankfunktionen ist der von ihnen erzeugte Code vom Erzeugersystem unabhängig.

In gewisser Hinsicht richtet sich TLO nach den aktuellen Bedürfnissen des Anwenders, wenn man davon ausgeht, daß der Bildschirm das wichtigste Entwicklungsinstrument ist und Sie nicht mit Papier und Bleistift vorausplanen wollen. Diese Methode wird dem Potential von TLO und Sycero jedoch nicht gerecht. Der Sieben-Punkte-Plan von Sycero betont sogar die Wichtigkeit der Planung:

- Planen
- System beschreiben
- Bildschirmaufbau erstellen
- Daten prüfen
- Programm definieren
- Druckformate angeben
- Programm generieren

Die Handbücher beider Systeme demonstrieren, daß Organisation und vorbereitende Planung eine grundlegende Bedeutung für gute Programme haben. Zunächst Sycero:

„Die Beseitigung von Fehlern, die aufgrund eines schlechten Aufbaus entstanden, kann länger dauern, als das eigentliche Schreiben des gesamten Programms. Es reizt zwar, nach fünf Minuten Nachdenken den Computer anzuschalten und mit Sycero das System nach und nach zu entwickeln, doch führt das nicht zu guten Programmen."

„. . . Sie sollten immer mit einer Liste beginnen, die alles enthält, was der Computer für Sie erledigen soll, was Sie auf dem Bildschirm sehen wollen (‚On-Line' Information) und was die Druckausgaben (‚Hardcopies') enthalten sollen. Erst wenn Sie sich entschieden haben, was das System liefern soll, sollten Sie prüfen, welche Informationen notwendig sind, um dieses Ergebnis zu erzeugen."

TLO geht zwar weniger ins Detail, sagt aber das gleiche:

„Planen Sie den gesamten Programmfluß, und stellen Sie sich die Fehler vor, die der Anwender machen wird. Versuchen Sie, diese Fehler abzufangen. Entwickeln Sie Ihr Programm ebenso sorgfältig, wie Sie andere Aufgaben erledigen. Es ist geschickt, eine Reihe kurzer Programme zu entwickeln, die durch ein kurzes Menü miteinander verbunden werden. Später werden Sie herausfinden, daß es erst durch diesen modularen Aufbau möglich wird, umfangreiche und komplexe Programme zu entwickeln."

Flexible Systeme

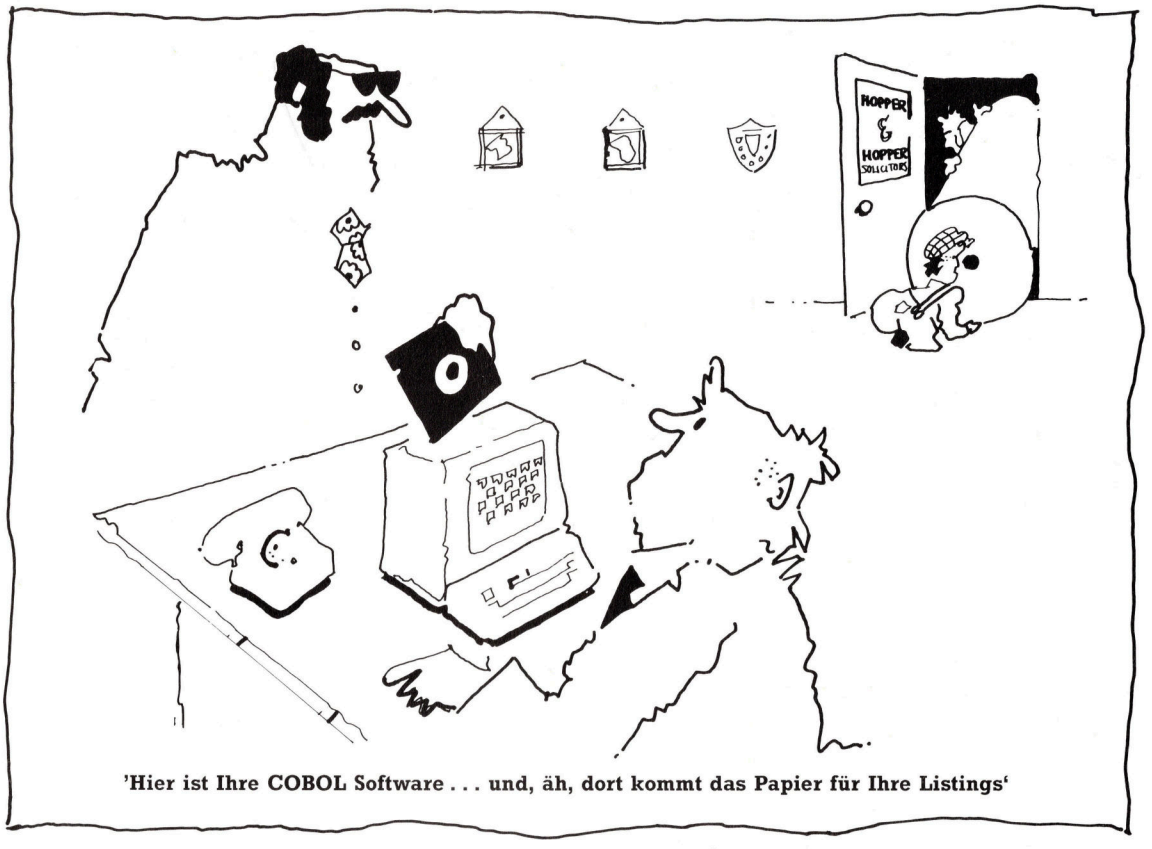
Keins der beiden Programme kann jedes beliebige Programm generieren. Sie können kein Spiel erzeugen – nicht einmal ein Abenteuerspiel, das ausschließlich Text verwendet –, kein Kalkulationssystem und auch keine Textverarbeitung. Die Stärke der Generatoren liegt darin, praktisch alle Arten von Daten bearbeiten, berechnen (von einfachen Mehrwertsteuerrechnungen bis zu den langen und komplizierten Gleichungen des Ingenieurwesens), speichern, abrufen und drucken zu können. Beide Systeme sind in dieser Hinsicht außerordentlich flexibel.

Es fragt sich, ob die Leistung der beiden Programmgeneratoren an Datenbanksysteme wie dBase II heranreichen können. Sie sind jedoch mit Sicherheit einfacher zu bedienen. In der nächsten Folge gehen wir genauer auf die Funktionsweise von Sycero und TLO ein.

The Last One: Für den Apple II und IIe, CP/M-80- und MS-DOS-Maschinen
Vertrieb: D J"AI" Systems Ltd, Summers Orchard, Speke Close, Ilminster, Somerset TA19 9BJ
Microtext: Für den Tatung Einstein und CP/M-80-Maschinen
Vertrieb: Transdata Ltd, 11 South Street, Havant, Hants
The Quill: Für den Spectrum, Commodore 64, Schneider CPC und Oric
Vertrieb: Gilsoft, 30 Hawthorne Road, Barry, South Glamorgan CF6 8LE
Sycero: Für MS-DOS-Maschinen
Vertrieb: System O Ltd, 7 Mill Street, Maidstone, Kent ME15 6XW

Die Bildschirme zeigen Programme, wie sie der Programmgenerator Microtext erzeugen kann. Das Testprogramm für einen Stekker (links im Bild) stellt dem Anwender Fragen über elektrische Systeme. Das rechte Programm prüft, ob Sie wissen, was Sie tun sollen, wenn Sie sich in einem brennenden Hotel befinden. Beide Programme vergleichen die Antworten des Anwenders mit den Informationen der internen Datenbanken. Diese Art der Anwendung eignet sich ideal für Programmgeneratoren.

COBOL für Micros



'Hier ist Ihre COBOL Software . . . und, äh, dort kommt das Papier für Ihre Listings'

COBOL wird auf Heimcomputern nur selten eingesetzt. Bei kommerziellen Anwendungen ist die Sprache jedoch führend, da die Programme sich leicht von einem Gerät auf andere übertragen lassen. In der ersten Folge unserer COBOL-Serie umreißen wir die vier Hauptabschnitte von COBOL und sehen uns den Datenteil genauer an.

COBOL (Common Business Oriented Language) ist die Programmiersprache, die weltweit mit Abstand am meisten eingesetzt wird. Zwar mag BASIC weiter verbreitet sein, doch wurden in COBOL mehr Programmzeilen geschrieben als in allen anderen Sprachen zusammen.

COBOL ist eine stark reglementierte Sprache. Ein ständiges „Komitee" (CODASYL) überwacht den Einsatz und die Entwicklung der Sprache. Der hohe Grad an Standardisierung hat verschiedene Gründe. Für kommerzielle Organisationen ist es wichtig, Software auf andere Geräte übertragen zu können, da Programme und Daten oft das wichtigste Firmenkapital darstellen und ohne wesentliche Änderungen (wenn möglich ohne jede Umstellung) auf anderen Computern laufen sollten.

Kommerzielle Programme unterscheiden sich wesentlich von den BASIC-Programmen der Heimcomputer. Sie sind sehr umfangreich, haben oft Zehntausende von Codezeilen und werden fast immer von Programmierteams entwickelt. Mit jedem dieser Faktoren wird der Code jedoch fehleranfälliger. Man kalkuliert daher bei kommerziellen Programmen eine Zeitspanne von mehreren Jahren ein, in denen Fehler beseitigt und Zusätze, Erweiterungen und Verbesserungen angefügt werden. Diese Überarbeitungen führen fast nie die Programmierer aus, die das Programm ursprünglich geschrieben haben.

Leider kann die COBOL-Programmierung durch die Notwendigkeit, Programme nach exakten Vorschriften erstellen zu müssen, extrem langweilig werden. Die Sprache selbst ist jedoch sehr interessant, keine andere eignet sich besser für komplizierte Dateiverarbeitung am Rechner.

COBOL-Programme haben vom Aussehen und den Abläufen her kaum Ähnlichkeit mit anderen Sprachen. Tief in ihrem umfangreichen Code verbergen sich jedoch die Kon-

zepte, die fast allen Programmiersprachen, die mit Prozeduren arbeiten, gemeinsam sind. Alle COBOL-Programme enthalten vier Abschnitte:
Benennungsteil (Identification Division): Dieser ‚Abschnitt dient zur Benennung und Beschreibung. Hier stehen die Namen von Programm und Programmierer, das Erstellungs- und Compilierungsdatum und Bemerkungen.
Maschinenteil (Environment Division): Hier wird der maschinen-spezifische Teil des Computers beschrieben, auf dem das Programm erstellt wurde. Da die modernen Betriebssysteme viele Aufgaben dieses Teils erledigen, stehen hier oft nur noch Informationen über die eingesetzten externen Dateien.

Problematische Variable

Datenteil (Data Division): Dieser Abschnitt beschreibt die im Programm vorkommenden Daten. Er entspricht etwa dem Deklarationsteil eines PASCAL-Programms, wobei COBOL normale Variablen jedoch nicht problemlos einsetzen kann. Der Datenteil ist in zwei Unterbereiche aufgeteilt: Der Dateibereich (File Section) enthält im wesentlichen die Datensatzlängen externer Dateien und der Arbeitsbereich (Working Storage Section) die Längen der vom Programm verwandten Datenfelder.
Verarbeitungsteil (Procedure Division): In diesem Teil stehen die Abläufe und Prozeduren, die die im Datenteil beschriebenen Daten bearbeiten. Der Verarbeitungsteil läßt sich je nach Bedarf in Abschnitte unterteilen, wobei jede Sektion (oder auch der gesamte Abschnitt) aus Paragraphen mit unterschiedlichen Namen besteht, die sich wiederum aus einzelnen Sätzen zusammensetzen. Ein Satz entspricht dabei einer oder mehreren „Anweisungen", die zur Steuerung des eigentlichen Ablaufs von „Bedingungen" verändert oder unterteilt werden können. Die Struktur des Verarbeitungsteils lehnt sich eng an das Umgangsenglisch. COBOL-Programme mit langen Feldnamen (bis zu 30 Zeichen) können daher nicht nur von Programmierern verstanden werden, sondern auch von EDV-Laien. In COBOL läßt sich aber auch unverständlicher „Spaghetticode" schreiben.

In diesem Artikel sehen wir uns die Rolle des Datenteils genauer an. COBOL versteht ihn als eine einzige lange Zeichenkette, die nach Belieben unterteilt wird. Am besten stellen Sie sich den Datenteil als Speicherblock vor. Jede Unterteilung dieses Blocks hat einen Namen und eine „Stufennummer", die die Rangordnung – und damit die Datenstruktur – festlegt.

Die Stufennummern reichen von 01 bis 49, einige weitere (höhere) Stufennummern bezeichnen spezielle Bereiche. Ein Teil des Datenbereichs wird unter Stufennummer 01 definiert. Diesen Teil können Sie weiter unterteilen, indem Sie den Subbereichen höhere Stufennummern (etwa 02) zuordnen. Die Nummern müssen nicht in ihrer natürlichen Reihenfolge vergeben werden. Einzelne Subbereiche lassen sich dann durch höhere Stufennummern stets nochmals unterteilen.

Programmierern, die nur BASIC kennen, muß die Struktur eines COBOL-Programms fremdartig vorkommen, doch eignet sich die Sprache ausgezeichnet für ihren Einsatzbereich: Sie ist leicht lesbar, kann auf andere Maschinen übertragen werden und eignet sich besonders gut für häufiges Prüfen und Aktualisieren von Daten.

Datenbezeichnungen können bis zu 30 Zeichen lang sein, dürfen allerdings nur Gedankenstriche, alphabetische und numerische Zeichen enthalten. COBOL ist mit einem umfangreichen Vokabular reservierter Wörter ausgestattet, die sich nicht als Datenbezeichnungen einsetzen lassen. Da aber in nur wenigen dieser Wörter Gedankenstriche vorkommen, entstehen dadurch keine Probleme.

Personendaten

Datendefinition für den Datensatz einer Personaldatei

```
01 ANGESTELLTER-DATENSATZ.
   02 ANGESTELLTER-NAME.
      03 ANGESTELLTER-ANFANGSBUCH-
         STABE PIC A.
      03 FILLER PIC X.
      03 ANGESTELLTER-NACHNAME
         PIC X(15).
   02 ANGESTELLTER-ABTEILUNG PIC XXX.
   02 ANGESTELLTER-GEHALTSSKALA PIC 9.
   02 ANGESTELLTER-GEB.
      03 GEB-TAG PIC 99.
      03 GEB-MONAT PIC 99.
      03 GEB-JAHR PIC 99.
```

Datensatz
Das Listing zeigt die Datentypen im Datensatz einer Angestelltendatei. PIC legt dabei die Datentypen fest, und 9 bezeichnet ein numerisches, A ein alphabetisches und X ein alphanumerisches Zeichen.

Datendefinition für die Kopfzeile eines Reports der oben angegebenen Datei:

```
01 HEADING-ZEILE.
   02 HEADING-1 PIC X(4) VALUE 'NAME'.
   02 FILLER PIC X(20) VALUE SPACES.
   02 HEADING-2 PIC X(10) VALUE
      'ABTEILUNG'.
   02 FILLER PIC X(3) VALUE SPACES.
   02 HEADING-3 PIC X(9) VALUE 'SAL SKALA'.
   02 FILLER PIC X(10) VALUE SPACES.
   2 HEADING-4 PIC X(3) VALUE 'GEB'.
```

Reportstruktur
Hier wird die Struktur eines Reports definiert. Dabei lassen sich den Datenelementen Anfangswerte zuweisen. HEADING-2 besteht beispielsweise aus zehn alphanumerischen Zeichen, die mit dem Wert „ABTEILUNG" versehen wurden.

Datendefinition für die Ausgabezeile des Reports:

```
01 OUTPUT-ZEILE.
   02 OUTPUT-NAME.
      03 OUTPUT-ANFANGSBUCHSTABE
         PIC A.
      03 FILLER PIC X VALUE'.'.
      03 OUTPUT-NACHNAME PIC X(15).
   02 FILLER PIC X(7) VALUE SPACES.
   02 OUTPUT-ABTEILUNG PIC XXX.
   02 FILLER PIC X(10) VALUE SPACES.
   02 OUTPUT-GEHALTSSKALA PIC 9.
   02 FILLER PIC X(18) VALUE SPACES.
   02 OUTPUT-DOB.
      03 OUTPUT-TAG PIC 99.
      03 FILLER PIC X VALUE'/'.
      03 OUTPUT-MONAT PIC 99.
      03 FILLER PIC X VALUE'/'.
      03 OUTPUT-JAHR PIC 99.
```

Lückenfüller
Dieser Code definiert den Inhalt einer Ausgabezeile der Personaldatei. In die Dateninhalte lassen sich mit dem reservierten Wort FILLER Zeichen einsetzen. Der Verarbeitungsteil spricht den Inhalt dieser Felder nicht an. (FILLER formatiert hier die Ausgabe mit Leerzeichen.)

Datendefinition eines eindimensionalen Arrays mit 20 dreistelligen Zahlen und eines zweidimensionalen Arrays mit 10*20 dreistelligen Zahlen (mit Vorzeichen und Dezimalstelle):

```
01 DATENTABELLE-1.
   02 DATENTABELLE-EINTRAG PIC S99V9
      USAGE
      COMPUTATIONAL OCCURS 20 TIMES.
01 DATENTABELLE-2.
   02 DATENTABELLE-ROW OCCURS 10
      TIMES.
      03 DATENTABELLE-EINTRAG-2 PIC S99V9
      USAGE
      COMPUTATIONAL OCCURS 20 TIMES.
```

Mit Hochgeschwindigkeit
Mit der Deklaration von Daten als COMPUTATIONAL legt dieser Programmteil fest, daß die entsprechenden Werte im Binärformat und nicht als Zeichen gespeichert werden. Mit Zahlen im Binärformat laufen mathematische Prozesse weitaus schneller ab.

COBOL unterscheidet zwei Datentypen: Elementardaten und Gruppendaten. Elementardaten lassen sich nicht weiter unterteilen. Der Verarbeitungsteil kann beide Datentypen ansprechen. Jedes Elementarfeld muß mit einer PICTURE-Eintragung genauer definiert werden. Die (wahlfreie) USAGE-Eintragung beschreibt mit COMPUTATION und DISPLAY das Zahlenformat.

COMPUTATIONAL speichert Zahlen im Binärformat statt als String und beschleunigt damit die mathematischen Abläufe. Ein Datenelement braucht jedoch nicht als COMPUTATIONAL definiert sein, um sich für Berechnungen zu eignen. DISPLAY wird vom System vorgegeben und speichert Zahlen automatisch als Zeichenkette. Jedes der beiden USAGEs läßt sich weiter verfeinern: So legt COMPUTATIONAL-3 das Fließkommaformat fest.

PICTURE definiert den Inhalt jeder Zeichenposition eines elementaren Datenelementes. Dabei steht die 9 für ein numerisches Zeichen, A für alphabetisch, X für alphanumerisch und V für die Position des Dezimalpunktes in einer Zahl. Dezimalpunkte werden normalerweise nicht gespeichert, doch merkt sich COBOL ihre Position, um Berechnungen korrekt durchführen zu können.

Ein S zeigt an, daß negative Werte möglich sind. Wird es weggelassen, speichert COBOL nur den absoluten Wert der Zahl. (Statt PICTURE läßt sich auch die Kurzform PIC einsetzen.) Ein Datenelement kann schließlich noch mit einem Anfangswert initialisiert oder als Arrayelement definiert werden.

Lochkarten als Ursprung

Wie FORTRAN war auch COBOL ursprünglich für die Arbeit mit Lochkarten bestimmt. Eine Programmzeile durfte daher 72 Zeichen nicht überschreiten und mußte – falls nötig – in einer Fortsetzungszeile weitergeführt werden. Viele Systeme erlauben jedoch eine flexiblere Struktur, wenn Programme nicht für andere Maschinen bestimmt sind. Die ersten sechs Spalten sind für Zeilennummern reserviert. Zwar braucht COBOL keine Zeilennummern, doch sind sie bei der Fehlersuche eine große Hilfe. Spalte 7 markiert entweder eine Fortsetzungszeile oder zeigt mit einem Stern an, daß der Zeileninhalt ein Kommentar ist. Die Spalten 8 bis 11 sind der A-Bereich. Hier beginnen die Namen der Abteilungen, Bereiche und Paragraphen und auch die Datenelemente der Stufennummer 01. Andere Daten und die Anweisungen des Verarbeitungsteils beginnen im B-Bereich – Spalte 12 bis 72. Die meisten Datendeklarationen sind Sätze und enden mit einem Punkt. Mit FILLER werden Felder bezeichnet, die der Druckgestaltung dienen und die das Programm nicht verarbeitet.

In der nächsten Folge sehen wir uns den Verarbeitungsteil genauer an.

Ereignisreich

Ein „Ereignis“ (englisch: Event) ist eine Art abschaltbares Interrupt. Wir untersuchen einige Ereignisse, die auf dem Acorn B zur Verfügung stehen.

Es gibt eine ganze Reihe von Situationen, in denen der Acorn B „Ereignissignale“ erzeugt: beim Eintrag von Zeichen in den Eingabebuffer, nach der Generierung eines „Vertical Sync Signals“, nach der Umwandlung eines Signals durch den A/D-Wandler und beim Drücken der Escape-Taste. Einige dieser Mechanismen erzeugen Interrupts, die dann über den IROV1-Vektor bearbeitet werden.

Mit diesen Ereignissen lassen sich daher – quasi durch die Hintertür – IRQs mit hoher Priorität einsetzen. Ereignisse müssen „angeschaltet“ werden, damit der Computer sie erzeugt und bearbeitet. Ist ein bestimmtes Ereignis angeschaltet, übergibt die CPU die Steuerung an eine Routine, deren Startadresse in dem Vektor bei &220 und &221 gespeichert ist. Um zu verhindern, daß diese Routine bei jeder Auslösung angesprochen wird, können Ereignisse auch „abgeschaltet“ werden. Das An- und Abschalten geschieht über zwei OSBYTE-Routinen. Die Tabelle zeigt die wichtigsten Ereignisse und die CALLs von OSBYTE, die sie an- und abschalten.

Befehle, die Ereignisse an- und abschalten

Ereignis	An	Aus
Ausgabebuffer wird entleert	*FX14,0	*FX13,0
Eingabebuffer wird gefüllt	*FX14,1	*FX13,1
Zeichen erreicht Eingabebuffer	*FX14,2	*FX13,2
A/D-Umwandlung beendet	*FX14,3	*FX13,3
Start des Vertical Sync	*FX14,4	*FX13,4
Intervalltimer ist abgelaufen	*FX14,5	*FX13,5
Escape wurde gedrückt	*FX14,6	*FX13,6

Um beispielsweise das Ereignis „Interval-Timer ist abgelaufen“ auszulösen, muß in BASIC nur *FX14,5 ausgeführt werden (oder der entsprechende Maschinencodebefehl). Nach Anschalten des Ereignisses löst der Timer jedesmal, wenn er Null erreicht, die Routine aus, deren Adresse im Ereignisvektor gespeichert ist. Das Programm stürzt ab, wenn sich dort kein Maschinencodeprogramm befindet.

Sie haben sicher bemerkt, daß es nur einen Ereignisvektor gibt, aber eine ganze Reihe von Ereignissen. Alle angeschalteten Ereignisse sprechen die gleiche Routine an, die jedes mögliche Ereignis bearbeitet. Wie weiß nun die Routine, von welcher Quelle sie ausgelöst wurde? Wenn nur ein Ereignis angeschaltet ist, ist die Situation eindeutig, bei mehreren gibt der Inhalt des A-Registers beim Einsprung in die Routine Auskunft über die Quelle. Die Tabelle zeigt den Registerinhalt beim Einsprung in die Routine, auf die der Ereignisvektor zeigt.

Registerinhalt beim Eintritt in die Ereignisroutinen

Ereignis	A-Register	X-Register	Y-Register
Ausgabebuffer	0	Buffernummer	#
Eingabebuffer wird entleert	1	Buffernummer	#
Zeichen aus Eingabebuffer	2	#	Zeichen
A/D-Umwandlung	3	#	Kanalnummer
Start des Vertical Sync	4	#	#
Intervalltimer	5	#	#
Escape wurde gedrückt	6	#	#

Bei einem # ist der Registerinhalt nicht festgelegt. Das bedeutet, daß die Ereignisroutine anfangs den Wert des A-Registers überprüfen kann und so feststellt, welches Ereignis die Routine ausgelöst hat.

Wenn Sie eine Routine zur Bearbeitung eines Ereignisses schreiben, müssen Sie zuerst den Registerinhalt sichern, da die entsprechende OS-Routine des Acorn B nicht wie bei Interrupts den Inhalt des A-Registers und das Status-Register auf den Stapel schiebt. Weiterhin gibt diese OS-Routine die Steuerung mit RTS an das unterbrochene Programm zurück. Doch genug der Theorie, sehen wir uns einige Beispiele an.

Große Ereignisse

- **Ereignis 0** wird beim Leeren des Ausgabebuffers erzeugt. Die Buffernummer ist die gleiche wie bei OSBYTE 21.
- **Ereignis 1** wird erzeugt, wenn im aktuellen Eingabebuffer ein weiteres Zeichen gespeichert werden soll. Auch hier entspricht die Buffernummer OSBYTE 21. Dabei ist der ASCII-Code, der nicht in den Buffer geschrieben werden konnte, beim Ansprechen der OS-Routine im Y-Register gespeichert.
- **Ereignis 2** wird erzeugt, wenn ein Zeichen den Eingabebuffer erreicht – normalerweise nach einem Tastendruck. Das Programm zeigt die Grundelemente einer Ereignisroutine, die bei jedem Tastendruck ein VDU 7 erzeugt. Das Ereignis tritt auch während des Ablaufs eines BASIC-Programms ein.

```
10 DIM mc% (100)
20 FOR I%=0 TO 2 STEP 2
30 P%=mc%
40 [OPT I%
50 PHP
60 PHA
70 TXA:PHA
80 TYA:PHA
90
100 LDA#7
110 JSR&FFE3
120
130 PLA:TAY
140 PLA:TAX
150 PLA
160 PLP
170 RTS
180
190 ]:NEXT I%
200 ?&220=mc%MOD256:REM niederwertiges Byte der im Vektor gespeicherten Adresse
210 ?&221=mc%DIV 256:REM höherwertiges Byte der Adresse
220 *FX14,2
230 REM das Ereignis nur anschalten, nachdem
240 REM die Routine angelegt wurde
250 REPEAT
260 PRINT I%
270 I%=I%+1
280 UNTIL FALSE
```

- **Ereignis 3** wird erzeugt, wenn einer der vier Kanäle des A/D-Wandlers im Acorn B eine Umwandlung vollendet hat. Je nach Wandlungsrate findet das Ereignis alle fünf oder zehn Millisekunden statt.
- **Ereignis 4** wird am Anfang eines Vertical Sync Impulses generiert. Es tritt pro Sekunde 50mal auf und kann somit gut als Quelle für Zeitimpulse dienen.
- **Ereignis 5:** Wie die Variable TIME wird auch der Intervalltimer jede Hundertstelsekunde aktualisiert. Dieser Timer wurde schon bei der Behandlung von OSWORD erwähnt. Er wird inkrementiert und löst beim Erreichen von Null ein Ereignis aus. Normale Abläufe lassen sich damit leicht in regelmäßigen Abständen unterbrechen. Über OSWORD mit A=4 wird in den Intervalltimer geschrieben und über OSWORD mit A=3 daraus gelesen.

Um im Acorn B nach einer Sekunde ein Ereignis auszulösen, wird der Intervalltimer mit seinem Maximalwert −100 geladen. Nach 100 Inkrementierungen erreicht der Timer dann den Wert Null. Der Timer hat eine Kapazität von fünf Bytes und somit den Maximalwert &FFFFFFFFFF. Das Programm zeigt diesen Vorgang und erzeugt dabei in regelmäßigen Abständen einen Signalton:

```
10 DIM mc% (100), delay%10
20 FOR I%=0 TO 2 STEP 2
30 P%=mc%
40 [OPT I%
50 PHP:PHA
60 TXA:PHA
70 TYA:PHA
80 LDA#7:JSR &FFE3
90 .clock set/reset für das nächste Interrupt
100 LDX# delay% MOD 256/ OSWORD anlegen
110 LDY #delay%DIV 256
120 LDA #4
130 JSR&FFF1
140 PLA:TAY
150 PLA:TAX
160 PLA:PLP
170 RTS
180 ]:NEXT I%
190 ?&220=mc%MOD 256
200 ?&221=mc%DIV 256
210 !delay%=&FFFFFF9C:delay%?4=&FF
220 *FX14,5
230 CALL clock set:REM clock starten
240 END
```

- **Ereignis 6** wird durch Drücken der Escape-Taste ausgelöst.

Es gibt drei weitere Ereignisse, die entweder einen Fehler des RS423-Systems, einen Econet-Fehler oder ein vom Anwender programmiertes Ereignis anzeigen. Unsere nächste Folge enthält einen zusammenfassenden Überblick über das OS des Acorn B.

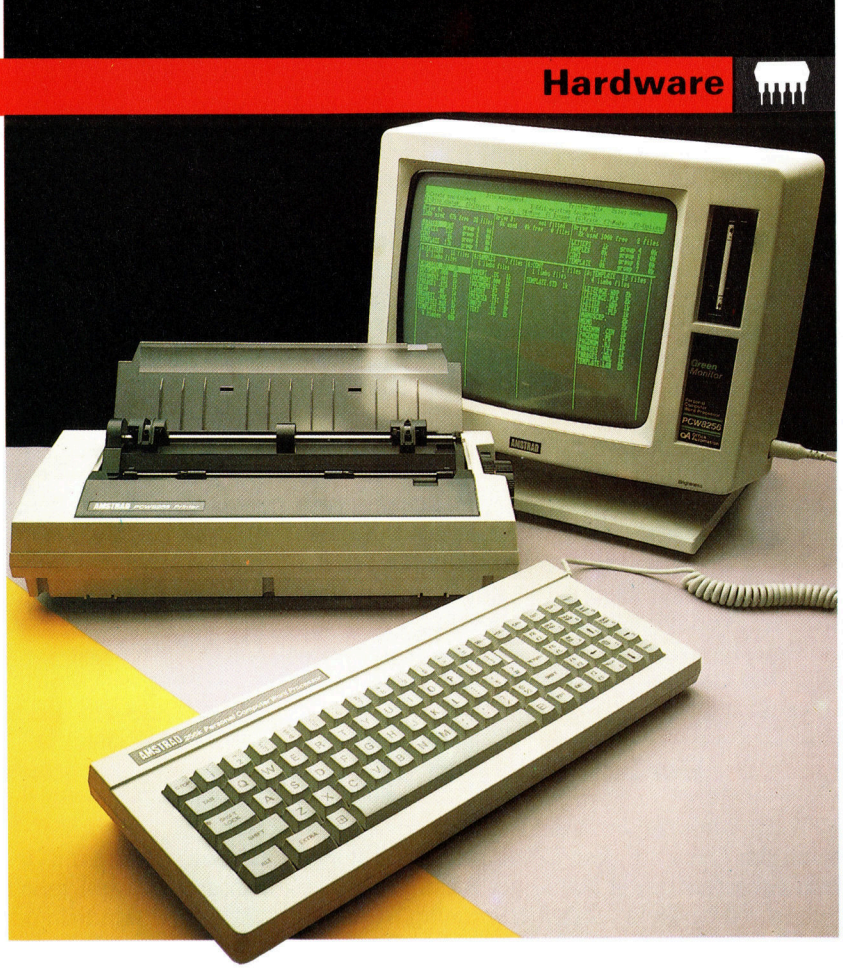
Testen Sie Ihre Kenntnisse

Das folgende Programm enthält Mechanismen, die in der Serie über das Betriebssystem des Acorn B erklärt wurden. Sehen Sie sich das Programm genau an, geben Sie es ein und lassen Sie es ablaufen. Versuchen Sie dabei herauszufinden, was das Programm bewirkt und wie es arbeitet. In der nächsten Folge bringen wir eine ausführliche Beschreibung und weitere Programmbeispiele.

```
 20 PROCassemble
 30 END
 50 DEFPROCassemble
 70 DIM MC% &FF
 80 oldvec=?&20E+256*?&20F
 90 FOR pass=0 TO 3 STEP 3
100 P%=MC%
120 [OPT pass
130.temp EQUB 00
140.start
150 PHP
160 STA temp
170 TXA:PHA
180 TYA:PHA
200 JSR check
230 PLA:TAY
240 PLA:TAX
250 LDA temp
260 PLP
270 JMP oldvec
290 .check
300 LDA temp
310 CMP#65
320 BMI out
330 CMP #91
340 BPL out
350 ADC #32
360 STA temp
370 .out RTS
390 ]:NEXT
400 ?&20E=start MOD 256
410 ?&20F=start DIV 256
420 ENDPROC
```

Der Texter

Hält das preisgünstige komplette Textverarbeitungssystem Joyce, ausgerüstet mit Monitor, Drucker und Diskettenstation, was der Name Schneider verspricht?



Der PCW 8256 Joyce ist Schneiders erster Micro für den Business-Markt. Er wird zwar als Textverarbeitungssystem angeboten – ausgestattet mit Monitor, Drukker, Diskettenstation und umfangreichem Textverarbeitungs-Software-Paket – ist aber ebenso ein leistungsfähiger 8-Bit-Standard-Computer.

Amstrad Consumer Electronics bringt traditionell Hochleistungs-Technologie im Komplettpaket. Der Schneider Joyce unterscheidet sich zwar von seinen Vorgängern, ist aber auf ähnliche Art gebaut. Als integriertes Textverarbeitungssystem entwickelt, gehören ein eingebautes Diskettenlaufwerk, ein Matrix-Drucker und die Textverarbeitungs-Software zum Lieferumfang.

Selbst wenn man die Produktions- und Marketing-Philosophie Amstrads schon kannte, war die Industrie doch von der Einführung des PCW 8256, Name „Joyce" (nach Alan Sugars Sekretärin), überrascht. Andere Schneider-Computer waren für den Massenmarkt entwikkelt worden, wogegen das jüngste Kind des Unternehmens in einem Spezialbereich plaziert wurde. Der Schneider Joyce kann zwar wie jeder andere Computer benutzt werden, ist aber vornehmlich zur Textverarbeitung für kleinere Betriebe gedacht.

Wie bei anderen Schneider-Computern beherbergt das Monitorgehäuse auch die Stromversorgung. Ferner befinden sich die Diskettenstation sowie der Anschluß für den Drucker in diesem Gehäuse. Ein Spiralkabel verbindet die Tastatur mit dem Monitor.

Die dicht gesetzten 82 Tasten entsprechen dem QWERTZ-Standard. Hinzu kommen Tasten wie Alt und Extra, mit denen zusätzliche Zeichenbelegung möglich ist. Rechts außen sind die vier programmierbaren Funktionstasten angeordnet, die mit bis zu acht Funktionen zu belegen und daher vielseitig einzusetzen sind. Rechts unten am Keyboard befindet sich die Cursor-Steuerungstastatur, und darüber liegen mehrere Tasten mit speziellen Textverarbeitungsaufgaben. Die Tastatur ist gegenüber der herkömmlichen zwar verbessert, klappert aber bei der Texteingabe doch beträchtlich.

Beim Monitor handelt es sich um ein „grünes" Standardmodell, das eine Auflösung von 90 mal 32 Zeichen bietet. Auf dem Bildschirm werden die Zeichen vor dem gleichen Hintergrund, aber mit verbesserter Lesbarkeit größer als bei anderen Schneider-Modellen dargestellt. Die Diskettenstation befindet sich in der oberen rechten Ecke des Monitor-Gehäuses. Darunter liegt ein Schacht für ein zweites Laufwerk.

Der fünf mal acht Punkt Matrix-Drucker des Joyce kann sowohl einzelne DIN A4-Blätter als auch das übliche Standard-Endlospapier verarbeiten. Er ist in zwei Modi betriebsfähig. Der erste, der sogenannte „draft" (Skizzen)-Modus, zeigt die Beschränktheit des Systems auf: Er ist ideal für die Erstellung von Kopien oder Notizen, weist aber bei genauerer Betrachtung ein unsauberes Schriftbild auf. Er kann nicht einmal eine Linie sauber ausdrucken.

Die ersten Klagen

Schneider stellt keine eigenen Drucker her, sondern läßt das Modell in Lizenz produzieren. Klagen von Anwendern sind die Folge. Entscheidender aber ist, daß für ein Textverarbeitungssystem ein Schönschreibdrucker Voraussetzung ist. Deshalb bleibt die Frage, warum Schneider die Entscheidung traf, ausgerechnet dieses Modell mit dem PCW 8256 zu liefern. Das Unternehmen hat die Unzulänglichkeiten erkannt und kündigte an, daß eine RS232- sowie eine Centronics-Schnittstelle lieferbar sein werden, um leistungsfähigere Drucker an den Rechner anzuschließen.

Entscheidet man sich aber für einen anderen Drucker, wird man mit verschiedenen Problemen konfrontiert. LogoScript, die mit dem Rechner gelieferte Textverarbeitungs-Software, hat Kommunikationsschwierigkeiten mit anderen Druckern. Eine Lösung böte die Modifizierung des CP/M 3.0-Betriebssystems, um

Dateien im ASCII-Format zu lesen und an den Drucker senden zu können. Das bedeutet aber: Man muß aus LogoScript aussteigen, um die Datei zu drucken. Und die zahlreichen attraktiven Druck-Formatierungsmöglichkeiten, die LogoScript bietet, lassen sich bei einem solchen Verfahren nicht benutzen.

Der „hochqualitative" Print-Modus überspielt die Hardware-Beschränkungen. Der Drucker bringt jede Zeile zweimal leicht versetzt zu Papier und schließt so das Schriftbild. Auf dieselbe Art erfolgt Fettdruck. Er verbessert zwar die Druckqualität, reduziert aber die Druckgeschwindigkeit drastisch von 90 Zeichen pro Sekunde (cps) auf 20 cps. Zweifel kommen auch bei der Verläßlichkeit des Druckers auf. Die mittlere Nadelreihe fiel im Test nach nur zweitägigem Gebrauch aus und hinterließ Lücken in den Zeichen.

Der bewährte Z80

Wie die anderen Schneider-Computer basiert auch der Joyce auf dem bekannten Z80-Prozessor, ausgestattet mit 256 KByte RAM. Die inzwischen verbreitete „bank switching"-Technik wurde in den Rechner integriert. Etwa 110 KByte wurden auf einer RAM-Diskette organisiert, die der Prozessor als Floppy behandelt. Da die eigentliche Information in RAM-Chips enthalten ist, ist fast sofortiger Zugriff möglich.

Die mit dem Joyce gelieferte Software umfaßt CP/M 3.0, eine verbesserte Version des mit anderen Schneider-Computern gelieferten Betriebssystems. Dazu gehören ebenfalls das AMSDOS-Disketten-Betriebssystem und das GSX-Grafikpaket von Digital Research. Damit kann der Anwender mit CP/M Grafiken darstellen (was in der Entwicklung ursprünglich nicht vorgesehen war). Nun laufen auch mehrere CP/M-Business-Programme, die ebenfalls diese Grafikfähigkeiten nutzen. Dazu liefert Schneider das Mallard BASIC, DR. LOGO und LogoScript.

Nach Inbetriebnahme zeigt sich der LogoScript-Schirm mit TAB-Markierungen auf dem Schirm, die beim Formatieren von Text helfen sollen. Viele Anwender werden aber den Eindruck gewinnen, daß sie der Bildschirmdarstellung eher schaden, da sie den Text nicht gerade lesbarer machen. Die Programmierer haben aber die Option vorgesehen, die Marker zu entfernen.

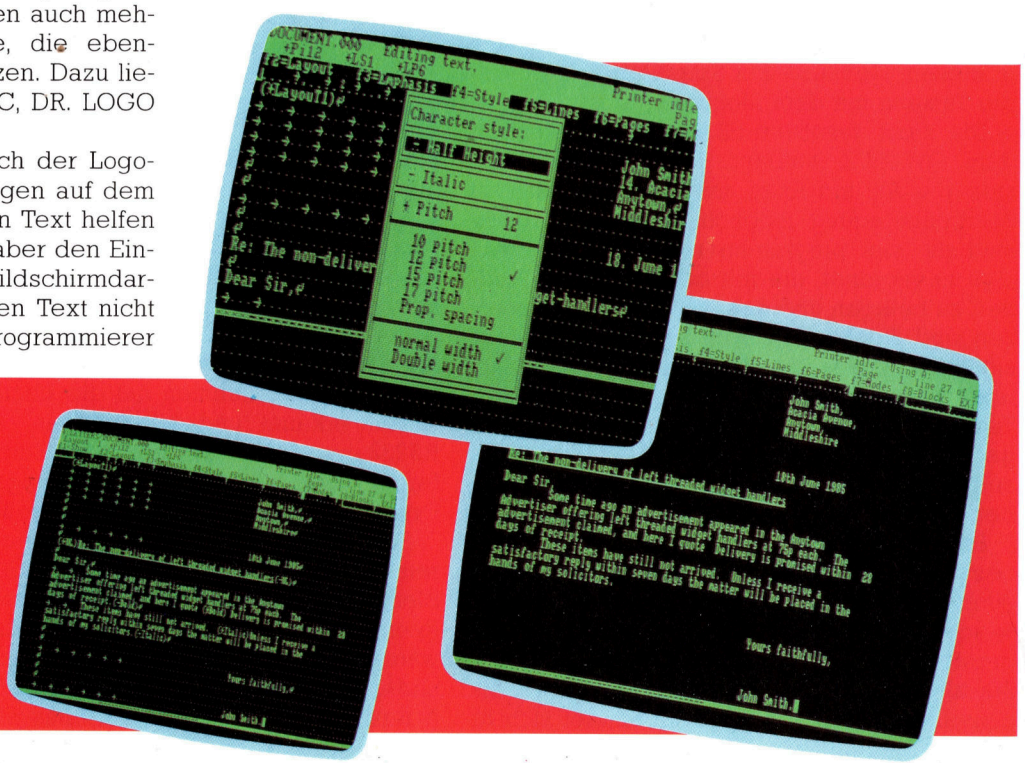
Die Software erweckt den Eindruck eines leistungsfähigen und flexiblen Textverarbeitungs-Systems, wie man es von teureren Rechnern kennt. LogoScript gibt die Möglichkeit, Text im bearbeiteten Dokument beliebig zu bewegen, den Cursor zeichenweise zu steuern oder Wörter und Absätze umzusetzen, auch Wörter und Sätze zu lokalisieren – Arbeiten, bei denen die speziellen Textverarbeitungstasten zum Einsatz kommen. Will man einen Textteil des Dokumentes in einen anderen kopieren, wird der Cursor auf den Anfang des zu schiebenden Abschnitts gesetzt und danach die „Copy"-Taste betätigt. Am Ende des Abschnitts wird der Vorgang wiederholt. Die Verschiebung erfolgt durch Steuerung des Cursors auf die gewünschte Position und Drücken der Paste-Taste.

Über Funktionstasten werden Korrekturen, Zeichenveränderung und Druck-Formate möglich. Ein „Pull-down"-Menü mit einer Reihe von Optionen kann mit den Cursortasten und Enter auf den Schirm geholt werden. Dieses System von Menüs und Untermenüs wurde entwickelt, um den Zugriff auf die zahlreichen Möglichkeiten der Textverarbeitung zu verbessern.

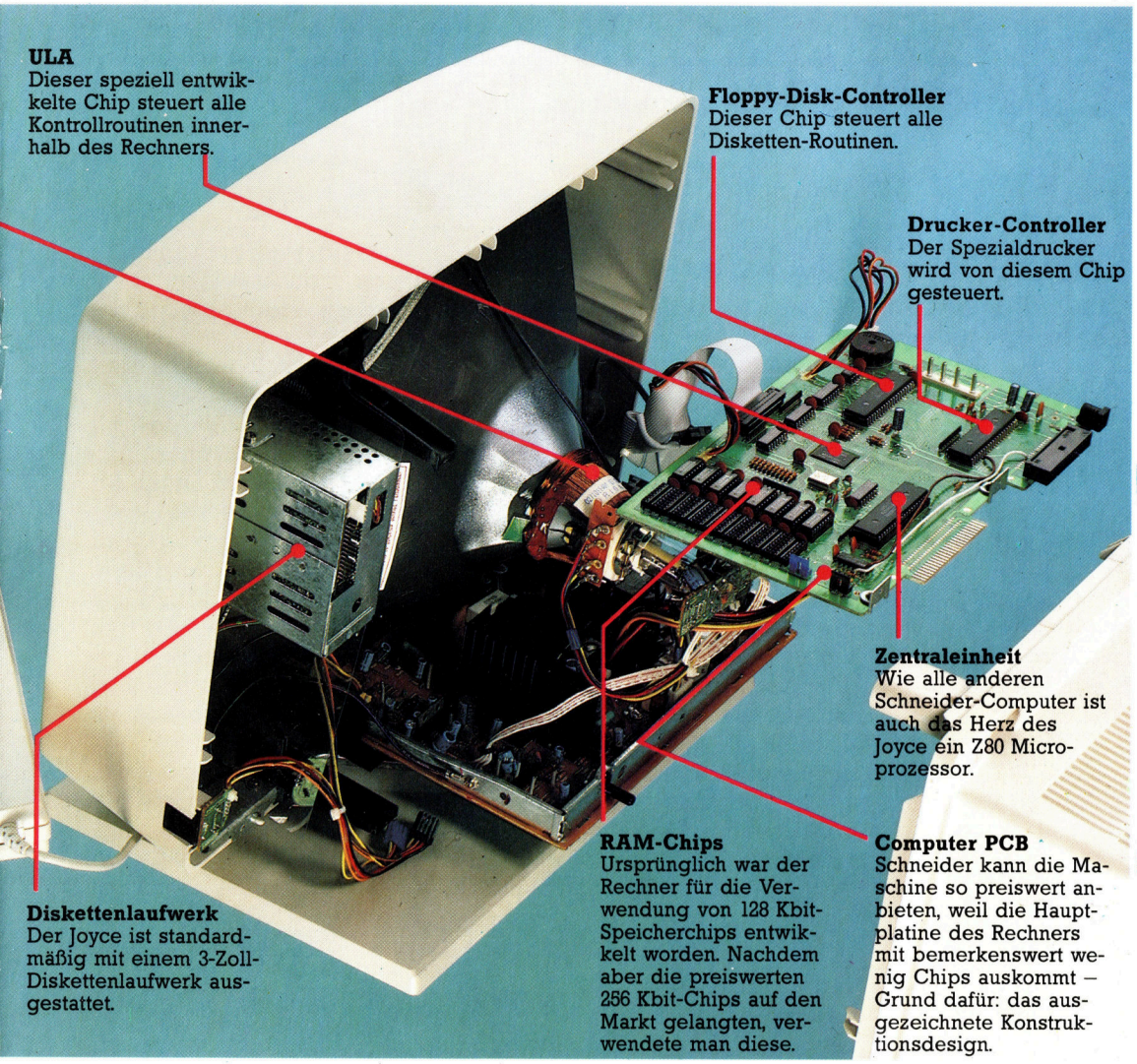
Wie in unserer Serie über Textverarbeitung dargelegt, verwenden die meisten Programme übermäßig viele Kontrollzeichen und separate Bildschirm-Menüs, um die vielfältigen Aufgaben auszuführen. Verständlich, daß die Programmierer des Joyce sich zur Menü- und Fenstersteuerung für das System der programmierbaren Funktionstasten entschieden haben, um das System zu vereinfachen. Obwohl man damit nicht mehr die zahlreichen Kontroll-

Monitor
Die Bildschirmdarstellung des Rechners erfolgt auf einem grünen Monitor.

Tastatur
Wie viele andere spezielle Textverarbeitungssysteme verfügt auch der Joyce über zusätzliche Tasten für die Bearbeitung von Texten.



Vor Ort
LogoScript, das speziell für den Joyce entwickelte Textverarbeitungssystem, bietet eine Fülle von Möglichkeiten. Einige stehen über Pull-Down Menüs am oberen Bildschirmrand zur Verfügung, andere sind in den Text integriert. Obwohl es nützlich ist, diese Befehle auf dem Bildschirm zu haben, könnten viele Anwender sie beim Schreiben eher als hinderlich betrachten. Zum Glück kann man sie vom Bildschirm löschen.



Schneider Joyce

ABMESSUNGEN
375 × 326 × 309 mm

SPEICHERKAPAZITÄT
256 K RAM, davon 110 K als RAM-Disk konfiguriert

ZENTRALEINHEIT
Z80

BILDSCHIRM-DARSTELLUNG
90 × 32 Zeichen
24 Zeilen × 90 Zeichen Textdarstellung bei LogoScript

SCHNITTSTELLEN
Anschluß für zweites Disketten-Laufwerk, Druckerschnittstelle, Tastatur-Anschluß

MITGELIEFERTE SOFTWARE
CP/M Version 3.0, LogoScript Textverarbeitung, Mallard BASIC, DR. LOGO, GSX Grafik

DOKUMENTATION
Wie bei Schneider-Computern üblich – sehr ausführlich

STÄRKEN
Der Joyce ist extrem preiswert. Der Verbraucher bekommt für wenig Geld einen 256 KByte Computer, Monitor, Diskettenstation und Drucker.

SCHWÄCHEN
Die Hardware hat einige Mängel. So ist besonders der Drucker zu langsam und qualitativ nicht hochwertig. Ferner scheint es, als ob selbst mit speziellem Interface ein anderer Drucker viele der Vorteile von LogoScript nicht nutzen kann.

zeichen auswendig lernen muß, wie es bei den meisten Textverarbeitungssystemen nötig ist, und bei diesem System jede Funktionstaste zu einem eigenen Menü führt, sind dennoch einige Beschriftungen der Tastatur vom Sinn her mißverständlich. Das bedeutet: Um den gewünschten Effekt zu erzielen, muß man auch hier einiges lernen.

Warum so umständlich?

LogoScript bietet nicht immer eine erstaunlich schnelle Befehlseingabe. Zu beiden Seiten der Leertaste befinden sich Tasten, die mit ± und = beschriftet sind. Um eine Zeile rechtsbündig auszudrucken, muß zunächst die ±-Taste, gefolgt vom Befehl RJ, gedrückt werden. Das = beendet die Ausführung. Das erfordert mehr Zeit als die Verwendung eines einzelnen Steuerzeichens, das es unter Umständen auch getan hätte.

Das Editieren am Bildschirm macht die Verwirrung komplett. Hat man einen Absatz geändert, machen einige Zeilen einen verrutschten Eindruck, da die Software den Absatz nicht automatisch richtig umlaufen läßt. Der Anwender muß das durch Drücken der Relay-Taste tun. Diese Art manueller Korrektur ist aus älteren Textverarbeitungsprogrammen bekannt. Man dürfte aber erwarten, daß ein 1985 entwickeltes Programm auf solche Umständlichkeiten verzichten kann.

Eine andere kuriose Besonderheit von LogoScript: Es scheint, als könne das Betriebssystem des Computers verschiedene Files nicht lesen. Das liegt am CP/M 3.0, das verschiedene Anwender bedienen kann. Wird das Betriebssystem geladen, geht es automatisch auf User 0. Nach Durchsuchen der Files, die unter verschiedenen Benutzer-Nummern gelistet sind, stößt der Anwender irgendwann auf die „fehlenden" LogoScript-Dokumente, und die Überraschung ist groß.

Trotz dieser Probleme bietet der Joyce viel fürs Geld. Umfang und Vielzahl der Druck- und Seitenformatierungsbefehle ähneln Programmen, die nur auf erheblich teureren Rechnern laufen. In diesem Fall aber scheint es, als sei Amstrad mit seiner Politik der Niedrigpreise über das Ziel hinausgeschossen. Ein klein wenig mehr Komfort für einen etwas nach oben korrigierten Preis wäre ideal.

Säbelrasseln

Zusätzlich zu den verschiedenen Ereignissen, die unsere Reise stören können, besteht die Gefahr einer Meuterei. Durch sie wird das Spiel vorzeitig beendet.

Bisher gibt es diverse Ereignisse, die die Fahrt positiv oder negativ beeinflussen können. Einige Ereignisse werden am Anfang jeder Woche zufällig ausgewählt und ausgeführt, kommen jedoch nur einmal pro Reise vor. Ein Ereignis kann jedoch jederzeit während des Spieles eintreten, wenn bestimmte Kriterien erfüllt sind: Werden die Zustände auf dem Schiff untragbar, meutert die Mannschaft.

Insgesamt gibt es acht Faktoren, die eine Meuterei auslösen können. Obwohl auf dem Schiff Platz für 16 Personen ist, gilt eine Anzahl über 12 für die Leute als Grund zur Unzufriedenheit. Wurde kein Koch angeheuert oder stirbt er während der Fahrt, muß die Mannschaft selbst kochen. Dadurch leidet die Essensqualität und der Unfriede steigt.

Da das Sichten des Albatros als gutes Zeichen gilt, wird die Mannschaft wieder beruhigt. Wird der Albatros jedoch getötet, steigt die Gefahr einer Meuterei. Ist die Mannschaft mit einem Vorrat auf halbe Ration gesetzt, wächst die Unruhe. Außerdem wurde der Mannschaft beim Anheuern gesagt, daß die Reise acht Wochen dauere. Jede zusätzliche Woche verstärkt den Unmut der Männer.

Um zu überprüfen, ob die Bedingungen für eine Meuterei ausreichen, wird der Meuterei-Faktor MF angelegt. Jede Bedingung wird durch Aufruf einer Unterroutine aus dem Hauptprogramm am Wochenanfang abgefragt. Ist das Ergebnis aller Tests positiv, wird ein Wert zu MF addiert. Das geschieht, bis MF den Wert 100 erreicht und eine Meuterei ausbricht. Außerdem gibt es einen Zufallsfaktor bis 30, der in die wöchentliche Zuordnung des Faktors MF integriert ist.

In Zeile 879 verzweigt das Programm zur Unterroutine bei Zeile 7200, die MF wöchentlich überprüft (Anfangswert = 0). Wird die Mannschaft während der Fahrt auf halbe Ration gesetzt, wird der Variablen H$ „Y" zugewiesen (und in Zeile 7215) durch die Meuterei-Routine

Zehntes Modul: Meuterei

Meuterei-Routine

```
879 GOSUB7200
```

Ergänzung zur Haupt-Reise-Routine

```
7200 REM MUTINY
7210 MF=0
7215 IFH$="Y"THENMF=MF+30
7220 NC=0
7225 FORT=1TO16
7228 IFTS(T,1)=5ANDTS(T,2)<>0ANDTS(T,2)<>-999THENN
C=1:T=16
7230 NEXT
7235 IFNC=0THENMF=MF+30
7240 IFA$="Y"THENMF=MF-20
7245 IFB$="Y"THENMF=MF+30
7250 IFCN.12THENMF=MF+30
7255 IFWT>NOTHENMF=MF+30
7260 IFWK>8THENMF=MF+((WK-8)*10)
7275 MF=MF+INT(RND(1)*30)
7280 IFMF<75 THENRETURN
7282 PRINTCHR$(147)
7284 IFMF>100THEN7300
7285 S$="CONDITIONS ON THE SHIP*":GOSUB9100
7286 S$="ARE GETTING WORSE*":GOSUB9100
7287 S$="AND SOME OF THE CREW*":GOSUB9100
7288 S$="ARE TALKING OF MUTINY'*":GOSUB9100
7290 PRINT:GOSUB9200
7292 S$=K$:GOSUB9100
7294 GETI$:IFI$=""THEN7294
7299 RETURN
7300 PRINTCHR$(147)
7305 PRINT:GOSUB9200
7310 S$="THE CREW HAVE MUTINIED*":GOSUB9100
7312 S$="BECAUSE:*":GOSUB9100
7313 X=0
7314 IFH$<>"Y"THEN7320
7315 GOSUB9200:X=X+1:PRINTX;
7316 S$="THEY HAVE BEEN ON 1/2 RATIONS*":GOSUB9100
7318 S$="   FOR SOME OF THE VOYAGE*":GOSUB9100
7320 IFNC<>0THEN7325
7321 GOSUB9200:X=X+1:PRINTX;
7322 S$="THERE IS NO COOK*":GOSUB9100
7324 S$="   AND THE FOOD IS AWFUL*":GOSUB9100
7325 IFB$<>"Y"THEN7330
7326 GOSUB9200:X=X+1:PRINTX;
7327 S$="THE ALBATROSS WAS KILLED!*":GOSUB9100
7330 IFCN<13THEN7335
7331 GOSUB9200:X=X+1:PRINTX;
7332 S$="THE SHIP IS OVERCROWDED*":GOSUB9100
7335 IFMO>=WT THEN7340
7336 GOSUB9200:X=X+1:PRINTX;
7337 S$="THERE ISN'T ENOUGH GOLD*":GOSUB9100
7338 S$="   FOR THEIR WAGES*":GOSUB9100
7340 IFWK<=8 THEN7350
7341 GOSUB9200:X=X+1:PRINTX;
7342 S$="THEY HAVE BEEN AT SEA*":GOSUB9100
7343 S$="   FOR MORE THAN 8 WEEKS*":GOSUB9100
7350 PRINT:GOSUB9200
7360 S$="THE CREW SEIZE THE SHIP*":GOSUB9100
7362 GOSUB9200
7363 S$="AND SAIL AWAY*":GOSUB9100
7370 GOSUB9200
7372 S$="LEAVING YOU ADRIFT IN *":GOSUB9100
7373 S$="AN OPEN BOAT*":GOSUB9100
7374 S$="LET'S HOPE YOU ARE PICKED UP":GOSUB9100
7375 PRINT:GOSUB9200
7380 PRINT"       GAME OVER"
7382 END
```

BASIC-Dialekte

Spectrum:
Führen Sie folgende Änderungen durch:

```
7282 CLS
7294 LET I$=INKEY$:IF I$=""THEN GO TO 7294
7300 CLS
```

Acorn B:
Führen Sie folgende Änderungen durch:

```
7282 CLS
7294 I$=GET$
7300 CLS
```

der Wert 30 zu der Variablen MF addiert.

Anschließend wird festgestellt, ob ein Koch an Bord ist, indem die Variable NC auf 0 gesetzt, eine Schleife von 1 bis 16 aktiviert und dann das erste Element des Stärke Typ Arrays TS(,) überprüft wird, ob der Wert 5=Koch vorhanden ist. Ist die Stärke des Kochs nicht 0 oder -999, wird NC auf 1 gesetzt. Wurde kein Koch angeheuert oder ist er gestorben, bleibt NC=0. In Zeile 7235 wird NC abgefragt, und beim Wert 0 die Zahl 30 zu MF addiert.

Wurde während der Reise bereits der Albatros gesichtet, wird in Zeile 6055 A$ auf Y gesetzt. Das bedeutet Glück und bewirkt (Zeile 7240) eine Reduzierung des Meuterei-Faktors MF um 20. Haben Sie den Albatros geschossen, setzt Zeile 6162 B$ auf Y, was Pech bedeutet, worauf in Zeile 7245 MF um 30 erhöht wird. Zeile 7250 überprüft, ob die Mannschaftsanzahl größer 12 ist, und addiert bei positivem Ergebnis 30 zu MF. Dadurch wird der Unwille der Mannschaft über die Enge ausgedrückt.

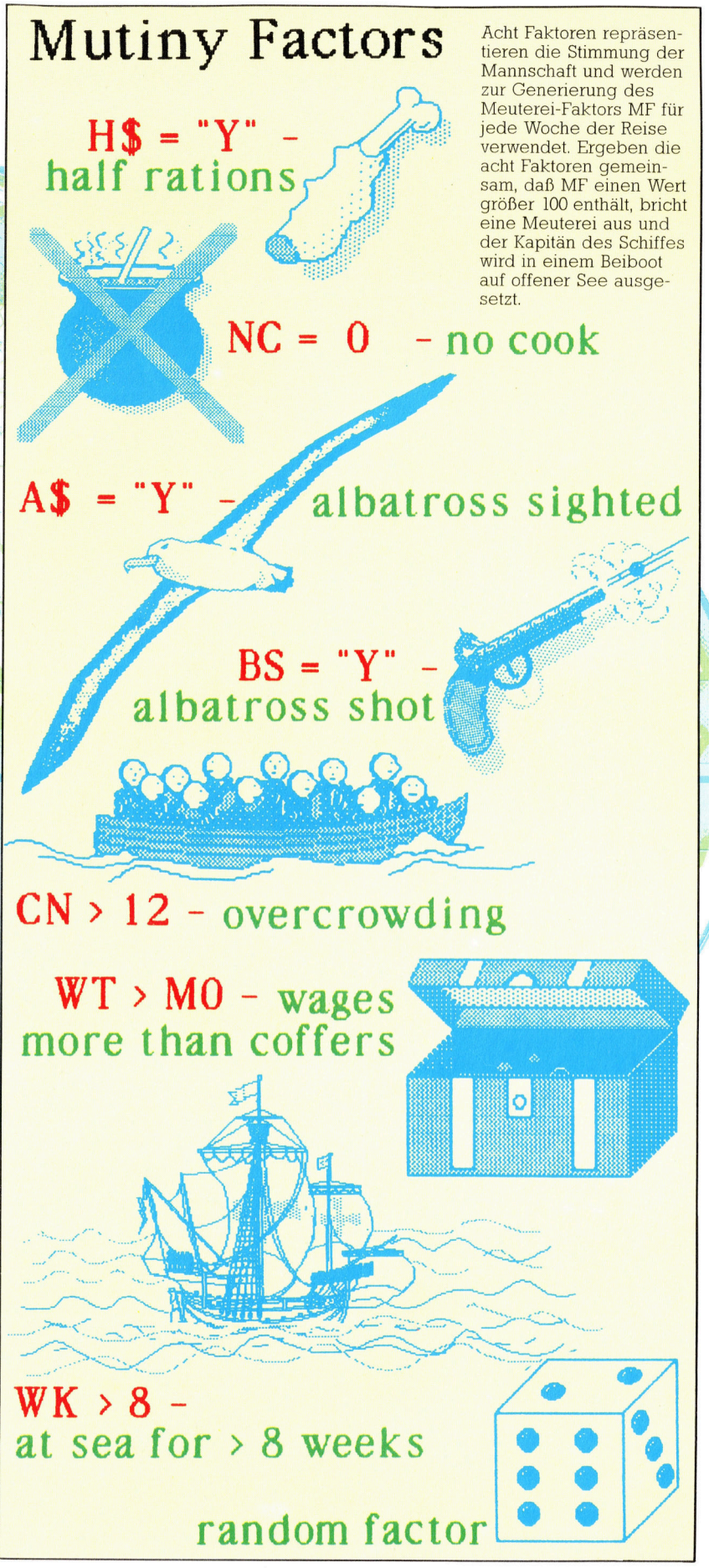
Acht Wochen Reisedauer

Die gesamte Lohnrechnung wird durch WT repräsentiert, das übrige Geld durch MO. Zeile 7255 überprüft, ob die Lohnsumme höher ist als das übrige Geld und addiert, bei positivem Ergebnis, 30 zu MF. Nach acht Wochen Reisedauer wird MF für jede weitere Woche um 10 erhöht. In Zeile 7260 wird getestet, ob die Reise wirklich schon über acht Wochen dauert, und, falls ja, von WK (Anzahl der Wochen) 8 subtrahiert, der Rest mit 10 multipliziert und zu MF addiert. In Zeile 7275 wird ein Zufallswert zwischen 0 und 29 zu MF addiert.

Ist nach Überprüfung aller Bedingungen der Wert von MF kleiner 74, erfolgt durch Zeile 7280 der Rücksprung zum Hauptprogramm. Ist der Faktor größer als 100, verzweigt Zeile 7284 das Programm nach 7300 – dort findet die Meuterei statt. Liegt der Faktor zwischen 75 und 100, werden Sie vor dem Rücksprung durch die Zeilen 7285 bis 7288 gewarnt, daß die Mannschaft eine Meuterei plant.

Meutert die Mannschaft, bestimmt das Programm die Ursache und stellt sie dar. Zeile 7314 überprüft, ob H$="Y" und meldet, daß die Mannschaft während der Fahrt auf halbe Ration gesetzt war. In Zeile 7320 wird getestet, ob NC=0, also kein Koch an Bord ist und somit die Meuterei Ergebnis zu schlechten Essens ist. Zeile 7325 stellt fest, ob der Albatros getötet wurde oder nicht, und gibt die entsprechende Meldung aus (Glück oder Pech). Zeile 7330 überprüft, ob zu viele Männer an Bord sind (mehr als 12), Zeile 7335 bestimmt, ob ausreichend Lohngelder vorhanden sind, und Zeile 7340 untersucht die Reisedauer.

Abschließend wird Ihnen mitgeteilt, daß viele Männer getötet wurden, die Rebellen die Kontrolle über das Schiff an sich gerissen haben und fortgesegelt sind.



Aus- und Umwege

Besonders wichtig für ein gutes Adventure-Spiel ist der scheinbar uneingeschränkte Handlungsspielraum der Spielfiguren. Wir entwickeln für diesen Zweck eine neue Baumstruktur.

Die im letzten Abschnitt hinzugekommenen Zeilen des Grundprogramms erlauben den Spielfiguren bereits einige Handlungen: Gegenstände können genommen und mit gewissen Einschränkungen auch benutzt werden. Bevor wir uns an die Entscheidungsbäume für die Interaktion der Personen und die Spielhandlung selbst heranwagen, sollten wir uns den Umgang mit Gegenständen noch etwas genauer ansehen.

Ganz allgemein lassen sich die Handlungen der Spielfiguren in drei verschiedene Stufen einteilen. Auf der höchsten Stufe werden Aktionen wie das Geben oder Nehmen von Gegenständen sowie das Absetzen einer Nachricht auf dem Bildschirm bearbeitet. Die zweite Stufe verwaltet Aktionen, die den Ablauf zwar nicht direkt beeinflussen, durch Informationen aber zur „Atmosphäre" des Spiels beitragen. Die dritte Stufe – etwa die Veränderung des „mood"-Flags in unserem Programm – gibt keine Nachrichten über den Bildschirm aus.

Beim Programmieren eigener Spiele sollte man diesen Stufenaufbau nie aus den Augen verlieren – jede Handlung kann zu diesem Zweck mit einem Wert versehen werden, der die entsprechende Stufe angibt.

Zurück zum Programmteil für die Verwaltung der Gegenstände: Das Löschen von Zeile 5010 sorgt dafür, daß unabhängig von der Anwesenheit des Spielers für jede Figur Meldungen angezeigt werden. Sie werden sehen, daß die Spielfiguren meist versuchen, ihre „eigenen" Gegenstände zu bekommen oder etwas zu trinken. Das entspricht auch unserem Wunsch, die Handlung einigermaßen realistisch zu gestalten. Sie können hier aber durch Veränderung der vorgegebenen Werte in Zeile 6030 und 6040 eingreifen. Durch Änderung des Besitzstandes von Fiona Frappe, Steve Swigg und Molly Mixer entsprechend auf 9, 12 und 8 und nachfolgendes Starten des Programms entwickelt sich die Situation in der Bar völlig anders als bisher. Versuchen Sie, auch andere Spielfiguren zu „beeinflussen"!

Eine Veränderung der Spielfiguren können Sie auch durch Überspringen der Zeilen 560 bis 580 sehr schnell erreichen. Dadurch würden alle Personen bei jedem Aufruf des Steuerprogramms bearbeitet und nicht erst dann, wenn ihr jeweiliges Zählflag bei 0 angekommen ist. Verändern Sie hierfür Zeile 550:

```
550 FOR c=1 TO 6: GOTO 590
```

Jetzt werden die Handlungen der einzelnen Figuren nacheinander dargestellt. Falls die Figuren nicht schnell genug durch das Hauptprogramm auf den neuesten Stand gebracht werden, können Sie die Flags des „handle"-Unterprogramms in Zeile 6030 und 6040 verändern.

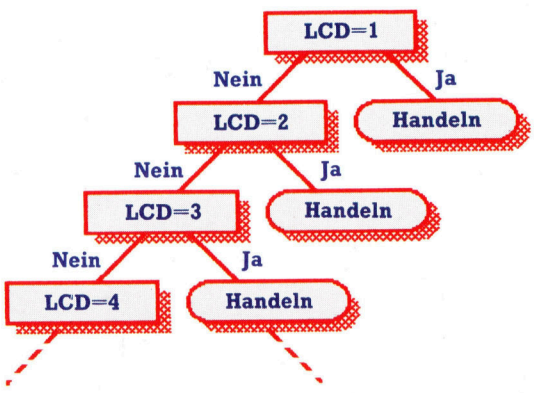
Der nächste Schritt betrifft die Interaktion der Spielfiguren. Das Array t in Zeile 190 ist bereits DIMensioniert, um die Daten für die drei Bäume mit ihren jeweils 25 Verzweigungspunkten aufzunehmen. Es kann sich herausstellen, daß wir diese Zahl verändern müssen. Folgenreicher ist aber die Feststellung, daß die für den Umgang mit Gegenständen verwendete Baumstruktur für eine Interaktion der Spielfiguren nicht leistungsfähig genug ist. Das leuchtet sofort ein, wenn wir uns den Gegenstands-Baum auf Seite 1731 ansehen: Diese Struktur liefert uns an jedem Verzweigungspunkt nur zwei Alternativen.

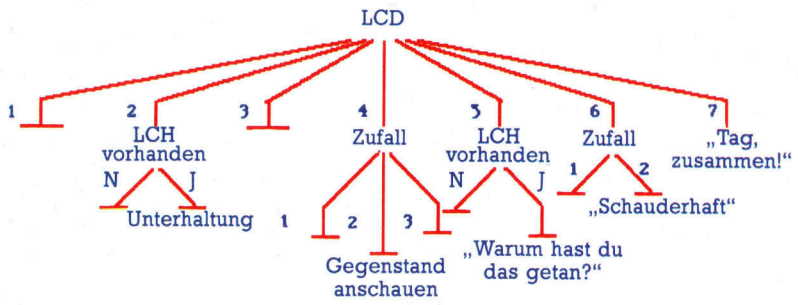
Bisher war das auch sinnvoll – für jede der zu prüfenden Tatsachen im Verzweigungsbaum gab es nur zwei Möglichkeiten – wahr oder falsch. Was aber, wenn es mehr als zwei Möglichkeiten gibt? Wollte man den Wert des LCD-Flags c$(n,9) mit seinen sieben Möglichkeiten durch eine Baumstruktur prüfen, ergäbe sich eine Struktur wie in der ersten Zeichnung. Sie ist jedoch eher unpraktisch, wie ein Vergleich mit dem Ablaufplan in der zweiten Zeichnung deutlich macht.

Es gibt zum Glück einen einfachen Weg zu diesem Aufbau, bei dem wir die Daten aus dem Array t für den neuen Baum weiterverwenden können. Der erste Baum arbeitet mit t(n,n,1) und t(n,n,2), wo die Nummern der Verzweigungspunkte standen, zu denen gesprungen werden sollte. Entscheidend war, ob der

Einfache Baumstruktur

Die hier abgebildete Baumstruktur bietet bei jedem Knotenpunkt nur jeweils zwei Wege. Bei der Abfrage mehrerer Bedingungen ist die Leistungsfähigkeit der Struktur durch diesen Aufbau stark eingeschränkt.





Freie Wahl

Dieser Baum zeigt die Verarbeitung des Code-Flags (c$(c,9)) durch eine Struktur, bei der an jedem Knotenpunkt mehr als zwei Wege verfügbar sind. Die Baumstruktur für unser Programm wird dadurch erheblich kompakter.

Wert im Array c eins oder zwei lautete. Wir können nun aber in t(n,n,1) eine Basisnummer für einen Knotenpunkt speichern, zu der ein Offset-Wert aus t(n,n,2) hinzugezählt wird. Der Knotenpunkt 1 der zweiten Zeichnung würde dann den Wert 2 in das Array-Element t(n,1,1) und den Wert von LCD minus 1 in das Element t(n,1,2) einlesen. Der Baum würde dann mit dieser Formel durchlaufen:

Neuer Knotenpunkt = t(Baumnummer, aktueller Verzweigungspunkt, 2) + t(Baumnummer, aktueller Verzweigungspunkt, 1)

Für unseren Interaktionsbaum werden wir diese Methode einsetzen. Sie hat zwar auch Mängel, erlaubt aber eine sehr kompakte Struktur für das Figurenmodul. Eine vorläufige Baumstruktur nach diesem Verfahren ist in der dritten Zeichnung dargestellt. Sie prüft zuerst die Spielstärke der Personen – ein „Toter“ oder „Bewußtloser“ ist kaum zu mehr nütze, als die Aufmerksamkeit der anderen Figuren anzuziehen. Als nächstes entscheidet das Programm durch Prüfen (und gegebenenfalls Verändern) des „move“-Flags in c$(n,11), ob eine Person bewegt werden soll. Zum Schluß wird einer der drei Unter-Bäume gewählt – Interaktion von Spielfiguren, Wahrnehmung von Gegenständen oder Meldung einer Aktivität.

Basic-Dialekte

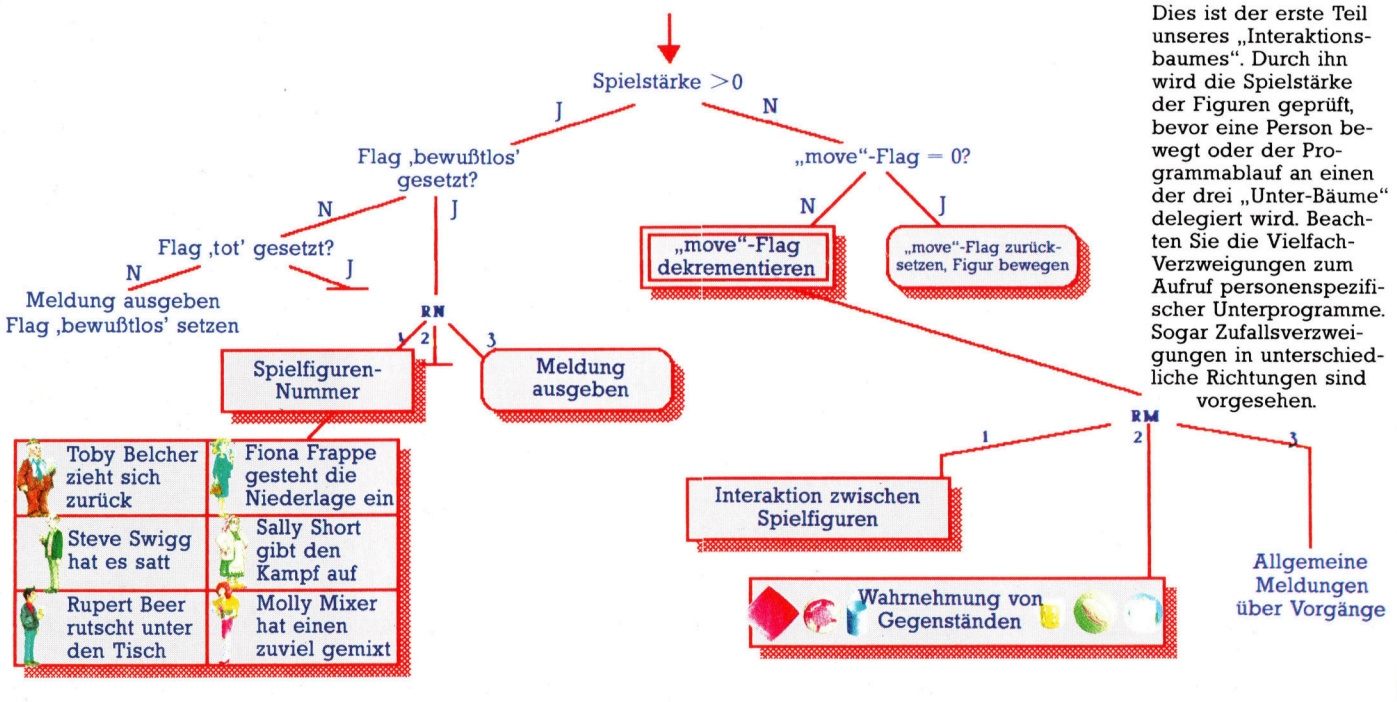
Hier die Ergänzungen und Veränderungen im Listing des letzten Abschnittes:

Spectrum:

```
4180 q=INT(RND*2)+1: RETURN
5080 IF n=23 THEN GOSUB 2540: GOTO 5040
5085 GOSUB 2640: GOTO 5040
5090 RESTORE 9900: FOR e=1 TO (n-23): READ h:
NEXT e: GOTO h
9900 DATA 5100, 5130, 5160, 5180, 5210, 5240,
5260, 5270, 5280, 5300, 5310, 5330, 5340, 5360,
5370, 5430
```

Acorn B

```
4180 q=RND(2): RETURN
```



Spielstärken

Dies ist der erste Teil unseres „Interaktionsbaumes“. Durch ihn wird die Spielstärke der Figuren geprüft, bevor eine Person bewegt oder der Programmablauf an einen der drei „Unter-Bäume“ delegiert wird. Beachten Sie die Vielfach-Verzweigungen zum Aufruf personenspezifischer Unterprogramme. Sogar Zufallsverzweigungen in unterschiedliche Richtungen sind vorgesehen.

Letzter Aufruf

Bei unserer Untersuchung des Betriebssystems des Acorn B legten wir dar, daß ROM-Routinen in bestimmten Situationen die Reaktion des OS stark beeinflussen. Wir beschließen diese Serie mit einigen Programmbeispielen aus der Praxis.

OSWRCH hat unter anderem die Funktion, BASIC-Programmlisten auf den Bildschirm zu bringen. Unser Maschinencode prüft nun den ASCII-Code jedes Zeichens, das während der OSWRCH-Routine im Akkumulator auftaucht. Liegt der ASCII-Wert zwischen 64 und 91 (Großbuchstabe), dann wird für den entsprechenden Kleinbuchstaben 32 addiert. Nach Ablauf unseres Codes geht die Steuerung mit einem Sprung (JMP) auf den ursprünglichen OSWRCHV-Vektor wieder an die eigentliche OSWRCH-Routine zurück. Alle Großbuchstaben der Listings erscheinen nun als Kleinbuchstaben auf dem Bildschirm. Die BREAK-Taste setzt OSWRCHV wieder auf den alten Wert.

Das erste Listing zeigt ein Dienstprogramm, das während der Programmierung die Speicherkapazität beobachtet. Der Ausdruck

```
HIMEM—(?2+256*?3)
```

liefert die Bytezahl, die noch zur Verfügung steht (Speicherstelle 2 und 3 enthalten die Adresse der Obergrenze der BASIC-Variablentabelle). Da es lästig ist, diesen Wert ständig abzufragen, arbeitet die Dienstroutine mit dem Ereignis „Zeicheneingabe in den Eingabebuffer" und bewertet bei jedem Tastendruck automatisch den obenstehenden Ausdruck. Fällt der verbleibende freie Speicher unter einen bestimmten Wert, erklingt ein Signalton. Die Routine wird bei jedem Tastendruck aktiviert und läuft beim Eingeben eines Programms ständig im Hintergrund ab.

Begründete Schwellenangst

Zunächst fragt das Programm nach einem Schwellenwert, von dem an es melden soll, daß Speicherkapazität knapp wird. Das Modul PROCselect-memory bei Speicherstelle &0A00 legt diesen Schwellenwert im lo-hi Format fest. Es schiebt zunächst die Registerinhalte zur Sicherung auf den Stapel. Nach Ablauf der Subroutine werden die Register zurückgeladen, während RTS die Steuerung an das Programm zurückgibt, das das Ereignis ausgelöst hatte. Nun geht es ins Detail.

Das eigentliche Submodul ruft zwei weitere Subroutinen auf, die den verbleibenden Speicher berechnen. Die Routine „inc" nimmt den in Speicherstelle 2 und 3 abgelegten Wert, addiert die nieder- und höherwertigen Bytes des Schwellenwertes und speichert das Ergebnis in &70 und &71. Die Routine „sub" zieht dann mit einer 16-Bit-Berechnung den Inhalt von &70 und &71 von HIMEM ab. Der Wert von HIMEM befindet sich in den Speicherstellen &06 und &07 der Zero Page. Wir dürfen hier absolute Speicheradressen verwenden.

Nach Ausführung der Subtraktion prüft Zeile 490 das Vorzeichen des Ergebnisses. Bei einem positiven Wert sind mehr Bytes frei, als der Schwellenwert angibt. Ein negatives Ergebnis dagegen löst über OSWRCH mit A=7 einen Signalton aus, und zeigt damit an, daß der Speicherplatz ab dieser Stelle im Programm nicht mehr genügt.

Speicherüberprüfung

```
100 REM **** Using Key Pressed Event ****
110 REM **** To keep an eye on memory use ****
120 MODE 1
130 PROCselect_memory
140 PROCassemble
150 END
160
170 DEFPROCselect_memory
180 INPUT"Beep when how many K left",n:n=n*1024
190 hibyte=n DIV 256
200 lobyte=n MOD 256
210 ENDPROC
220
230 DEFPROCassemble
240 oswrch=&FFEE
250
260
270 FOR pass=0 TO 3 STEP 3
280 P%=&0A00
290
300 [OPT pass
310 .start
320 PHP
330 PHA
340 TXA:PHA
350 TYA:PHA
360
370 JSR memory_check
380
390
400 PLA:TAY
410 PLA:TAX
420 PLA
430 PLP
440 RTS
450
460 .memory_check
470 JSR inc
480 JSR sub
490 BPL room
500 LDA #7
510 JSR oswrch
520 .room RTS
530
540 .sub SEC
550 LDA &06:SBC &70:STA &70
560 LDA &07:SBC &71:STA &71
570 RTS
580 .inc CLC
590 LDA &02:ADC #lobyte:STA &70
600 LDA &03:ADC #hibyte:STA &71
610 RTS
620
630 ]:NEXT
640 ?&220=start MOD 256
650 ?&221=start DIV 256
660 *FX14,2
670 ENDPROC
```

Unser nächstes Programm löst bei dem Ereignis „Intervalltimer ist abgelaufen“ eine Tonfolge aus, während der Computer an anderen Aufgaben weiterarbeitet. Da die Ereignisroutine im Hintergrund abläuft, kann so das Sichern, Editieren, Drucken und der Ablauf eines anderen Programms mit Musik untermalt werden. Sie könnten mit diesem Modul aber auch Spiele mit Melodien unterlegen, die beispielsweise beim Laden von Dateien erklingen, bis der Vorgang abgeschlossen ist.

Der Programmstart liegt bei &0C00. Zuerst wird eine Initialisierungsroutine mit folgenden Aufgaben ausgeführt: Der Ereignisvektor EVENTV bei &220 und &221 wird auf unsere Routine umgestellt. Danach aktiviert OSBYTE 14 mit X=5 das Ereignis „Intervalltimer ist abgelaufen“ an. Ein „Notenzähler“ bei &70 wird auf Null gestellt und schließlich der Ereignistimer mit einem Aufruf der Subroutine „Clock“ gestartet.

„Soundtable“

Die eigentliche Routine schiebt zunächst die CPU-Register auf den Stapel und ruft dann eine Subroutine auf, die eine einzelne Note spielt. Danach wird der Timer wieder auf Null gesetzt, und schließlich werden die Register zurückgeladen. Beachten Sie, daß alle Ereignisroutinen die Register standardmäßig speichern und wiederherstellen. Es muß daher nur der Code zwischen diesen beiden Programmblöcken verändert werden.

Jedesmal wenn der Ereignistimer die Null erreicht, ertönt eine aus der Notentabelle „notetable“ ausgewählte Note. Diese Tabelle besteht nur aus einer Serie von Bytes, die die Tonhöhe der gewünschten Töne festlegen. Ein Zähler in &70 bestimmt, welcher der fünf Töne gespielt werden soll. Der Tonhöhenwert wird nun gelesen und auf dem entsprechenden Platz von „soundtable“ abgelegt. „Soundtable“ ist der Parameterblock für die OSWORD-Routine, mit der wir später den Ton erzeugen. Danach wird der Zähler inkrementiert, der nun auf die nächste Position der Notentabelle zeigt. Da sich in der Tabelle nur fünf Werte befinden, wird der Zählerwert in &70 wieder auf Null gesetzt, wenn er die Fünf erreicht hat.

Die Subroutine „clock“ aktiviert den Ereignistimer über OSWORD mit A=4. Im Speicherbereich „time“ befindet sich der Wert für das Timerregister, das die Pausen zwischen den einzelnen Tönen festlegt. Die Subroutine „sound“ arbeitet (wie der BASIC-Befehl SOUND) mit OSWORD mit A=7, wobei „soundtable“ den Parameterblock des OSWORD-Aufrufs darstellt. Die BASIC-Befehle in den Zeilen 840 und 850 legen die Daten für „notetable“ und „soundtable“ fest.

BREAK setzt den Inhalt von EVENTV wieder auf seine ursprünglichen Werte und stoppt so das Programm. Der Aufruf von „initialise“ startet die Routine dann wieder neu.

Unser letztes Programm zeigt, wie Sie mit Ereignissen einen einfachen grafischen Umriß (in diesem Fall ein Rechteck) bei jedem Eintreten des Ereignisses auf dem Schirm um eine Position verschieben können. Die Geschwindigkeit der Bewegung hängt vom Wert des Ereignistimers ab, der in Zeile 1260 gesetzt wird. Die Definition der zu bewegenden Form ist mit PLOTzahlen in „shapetable“ gespeichert. „Shapetable“ wird von einem DATA-Befehl definiert und in Zeile 1220 mit POKE in den Speicher gesetzt. Zeile 1230 und 1250 setzen EVENTV auf unser Programm und schalten das entsprechende Ereignis an.

Subroutinenaufrufe

Zeile 200 bis 230 sichern die Register, während die Subroutinenaufrufe der Zeilen 250 bis 280 die eigentliche Arbeit ausführen. Die Register werden schließlich wiederhergestellt und ein RTS ausgeführt. Die Zeilen 370 bis 440 sichern den aktuellen Status des Grafikcursors und von GCOL, um sie dann schließlich nach dem Rücksprung von der Ereignisroutine zurückladen zu können.

Die Zeilen 460 bis 520 bewegen den Umriß, und Zeile 540 bis 630 versetzen die Grafikfarbe und den Grafikcursor wieder in den Zustand vor dem Ansprung der Routine.

Die Zeilen 650 bis 720 führen mit der OSWRCH-Routine, die dem BASIC-Befehl MOVE entspricht, die Bewegung zu den gewünschten X- und Y-Positionen aus. Diese Positionen werden in xpos und xpos+1 für die

Backgroundmusik

```
10 REM **** SOUND EVENTS ****
50
60 PROCassemble
80 CALL initialise
90 END
100
110 DEFPROCassemble
120 note_count=&70
130 oswrch=&FFEE
140 osbyte=&FFF4
150 osword=&FFF1
160
170 FOR pass=0 TO 2 STEP 2
180 P%=&0C00
190
200 [OPT pass
210 .initialise
220 LDA #event MOD 256:STA &220
230 LDA #event DIV 256:STA &221
240 LDA #14:LDX #5:JSR osbyte
250 LDA #0:STA note_count
260 JSR clock
270 RTS
280
290 .event
300 PHP
310 PHA
320 TXA:PHA
330 TYA:PHA
340
350 JSR play_note
360 JSR clock
370
380
390 PLA:TAY
400 PLA:TAX
410 PLA
420 PLP
430 RTS
440
450 .play_note
460 LDY note_count
470 .sloop LDA notetable,Y
480 STA soundtable+4
490 JSR sound
500 INC note_count
510 LDA note_count:CMP#5:BNE out
520 LDA #0:STA note_count
530 .out RTS
540
550
560 .clock
570 LDX #time MOD 256
580 LDY #time DIV 256
590 LDA #4
600 JSR osword
610 RTS
620
630 .sound PHA
640 TXA:PHA
650 TYA:PHA
660 LDX #soundtable MOD 256
670 LDY #soundtable DIV 256
680 LDA #7
690 JSR osword
700 PLA:TAY
710 PLA:TAX
720 PLA
730 RTS
740
750 .notetable
760 EQUD &00000000
770 EQUD &00000000
780 .soundtable
790 EQUD &00000000
800 EQUD &00000000
810 .time EQUD &FFFFFFCF
820 EQUB &FF
830 ]:NEXT
840 FOR I%=0 TO 4:READ data:?
    (notetable+I%)=data:NEXT
850 FOR I%=0 TO 7:READ data:?
    (soundtable+I%)=data:NEXT
860 ENDPROC
870 DATA 69,73,81,89,97
880 DATA 1,0,&FA,&FF,0,0,10,0
```

X-Koordinate und ypos und ypos+1 für die Y-Koordinate festgelegt. Die Zeilen 740 bis 800 schließlich zeichnen – wiederum mit OSWRCH – den Umriß. Das Y-Register dient dabei als Indexregister für die Bytetabelle „shapetable", in der sich der Umriß befindet.

Bei der Bewegung des Umrisses über den Schirm wird deutlich, daß die Positionen von x und y bei jedem Eintritt des Ereignisses aktualisiert werden müssen. Die Routinen „incx" und „incy" in Zeile 820 bis 900 führen diese Aufgabe aus. Sobald das Rechteck die obere Bildschirmgrenze erreicht, stellt die Routine „overflow" die X- und Y-Koordinaten auf Null zurück. Die clock-Routine schaltet über OSWORD mit A=4 den Ereignistimer aus, wobei „gcol" dem BASIC-Befehl GCOL3,1 entspricht.

Erhöhter Zeitwert

Das Programm ist nicht ganz ohne Probleme. Solange das Scrollen noch nicht eingesetzt hat, bewegt sich der Umriß recht unsicher über den Bildschirm, während Sie andere Vorgänge ausführen können, die dadurch nicht gestört werden. Wenn Sie die Bewegung jedoch beschleunigen, indem Sie den Wert von „time" erhöhen, funktioniert z. B. das Listen eines Programms bei der schnelleren Bewegung des Rechtecks zwar immer noch, doch tauchen seltsame Zeichen auf dem Schirm auf. Zwischen unserer Routine, die bei jedem Ereignis ausgelöst wird, und den normalen OS-Abläufen eines Listings entsteht ein Konflikt. Da beide Routinen mit OSWRCH arbeiten, findet die Listroutine nach der Unterbrechung durch unsere Ereignisroutine OSWRCH nicht mehr in dem Zustand vor, den sie erwartet.

Wenn beim Unterbrechen von OSWRCH die unterbrechende Routine wiederum OSWRCH aufruft, kann OSWRCH nur einmal verlassen werden. Die Firma Acorn schlägt daher vor, OS-Routinen nicht in Ereignis- und Interruptroutinen zu verwenden. Sie lassen sich zwar einsetzen, doch sollten Sie dabei sehr vorsichtig sein. Wenn bei Interrupts keine OS-Routinen verwandt werden, entstehen auch keine Probleme. Hier haben wir eine der wenigen Situationen, in denen der Einsatz von ROM-Routinen nicht zu empfehlen ist. Grafik und Daten lassen sich aber auch direkt in den Bildschirmspeicher schreiben.

Bewegte Grafik

```
 10 REM **** GRAPHICS USING  EVENT TIMER ****
 30 MODE 1
 40 PROCassemble
 50 CALL clock
 60 END
 70 DEFPROCassemble
 80 xpos=&70:ypos=&72
 90 ?xpos=0:?(xpos+1)=0
100 ?ypos=100:?(ypos+1)=0
110 oswrch=&FFEE
120 osbyte=&FFF4
130 osword=&FFF1
140 DIM code% 600,time% 10
150
160 FOR pass=0 TO 2 STEP 2
170 P%=code%
180
190 [OPT pass
200 PHP
210 PHA
220 TXA:PHA
230 TYA:PHA
240
250 JSR preserve
260 JSR draw
270 JSR restore
280 JSR clock
290
300
310 PLA:TAY
320 PLA:TAX
330 PLA
340 PLP
350 RTS
360
370 .preserve
380 LDX #pgpos MOD 256
390 LDY # pgpos DIV 256
400 LDA #&0D
410 JSR osword
420 LDA &35B:STA ccol
430 LDA &35C:STA ccol+1
440 RTS
450
460 .draw
470 JSR gcol
480 JSR move
490 JSR shape
500 JSR incx
510 JSR incy
520 RTS
530
540 .restore
550 LDA ccol:STA &35B
560 LDA ccol+1:STA &35C
570 LDA #25:JSR oswrch
580 LDA #4:JSR oswrch
590 LDA cgpos:JSR oswrch
600 LDA cgpos+1:JSR oswrch
610 LDA cgpos+2:JSR oswrch
620 LDA cgpos+3:JSR oswrch
630 RTS
640
650 .move
660 LDA #25:JSR oswrch
670 LDA #4:JSR oswrch
680 LDA xpos:JSR oswrch
690 LDA xpos+1:JSR oswrch
700 LDA xpos+2:JSR oswrch
710 LDA xpos+3:JSR oswrch
720 RTS
730
740 .shape
750 LDX#24:LDY #0
760 .sloop LDA shapetable,Y
770 JSR oswrch
780 INY:DEX
790 BNE sloop
800 RTS
810
820 .incx CLC
830 LDA #2:ADC &70:STA &70
840 LDA #0:ADC &71:STA &71
850 RTS
860 .incy CLC
870 LDA #2:ADC &72:STA &72
880 LDA #0:ADC &73:STA &73
890 LDA &73:CMP #3:BCS overflow
900 RTS
910 .overflow
920 LDA #0:STA &72:STA &73
930 LDA #0:STA &70:STA &71
940 RTS
950
960 .clock
970 LDX #time% MOD 256
980 LDY #time% DIV 256
990 LDA #4
1000 JSR osword
1010 RTS
1020
1030
1040 .gcol
1050 LDA #18:JSR oswrch
1060 LDA #3:JSR oswrch
1070 LDA #1:JSR oswrch
1080 RTS
1090
1100
1110 .ccol EQUD &0000
1120 .pgpos EQUD &00000000
1130 .cgpos EQUD &00000000
1140 .shapetable
1150 EQUD &00000000
1160 EQUD &00000000
1170 EQUD &00000000
1180 EQUD &00000000
1190 EQUD &00000000
1200 EQUD &00000000
1210 ]:NEXT
1220 FOR I%=0 TO 23:READ data:?(shapetable+I%)=data:NEXT
1230 ?&220=code% MOD 256
1240 ?&221=code% DIV 256
1250 *FX14,5
1260 !time%=&FFFFFFEF:time%?4=&FF
1270 ENDPROC
1280 DATA 25,1,60,0,0,0,25,1,0,0,50,0,25,1,&C4,255,0,0,
```

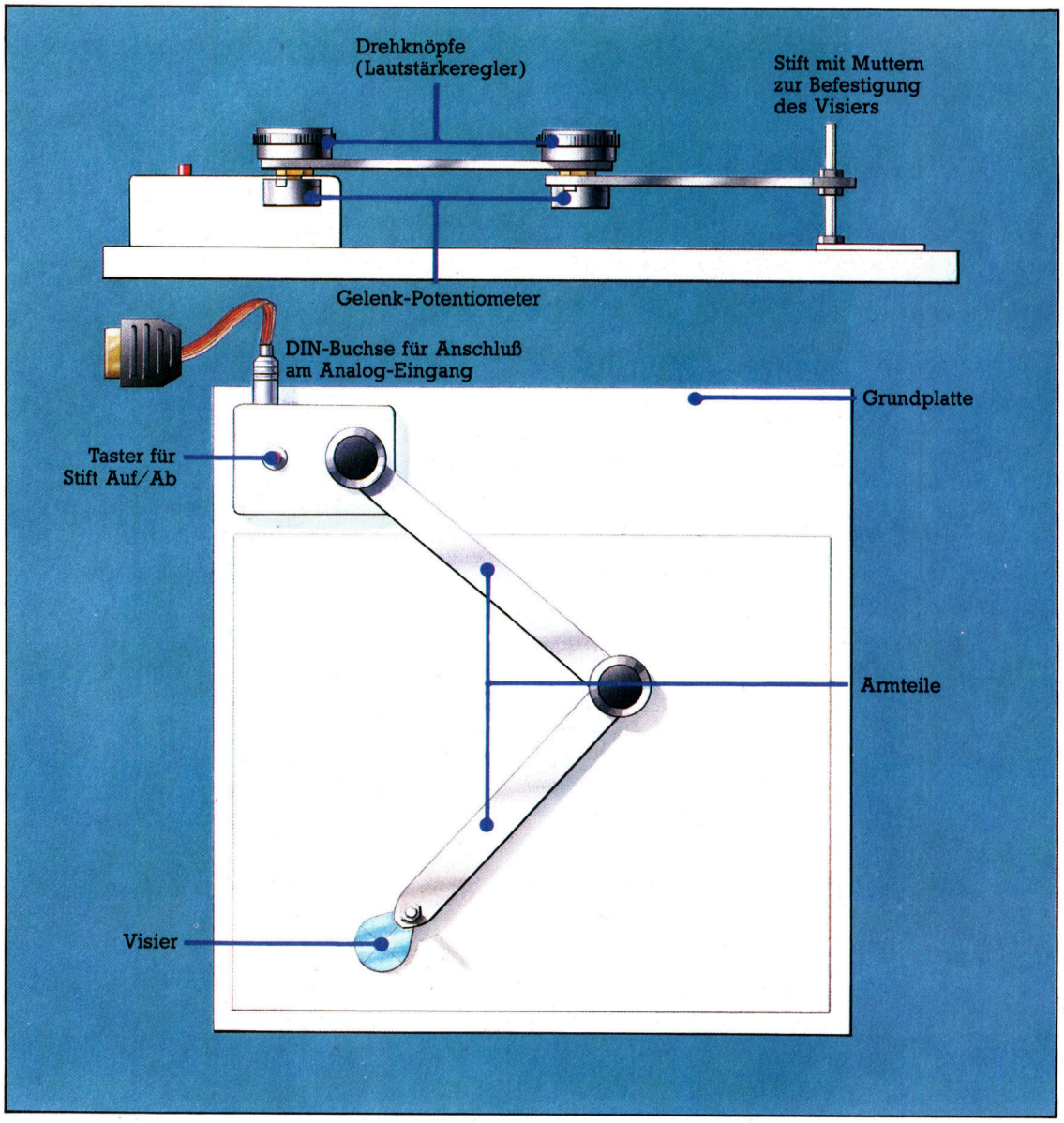
Abtaster

Bei unserem neuen Selbstbauprojekt soll ein Digitalisierarm (Tracer) für den Acorn B entstehen. Zunächst beschreiben wir kurz die Eigenschaften dieses Peripheriegeräts.

Ein Digitalisierarm oder „Tracer" ist ein relativ einfach konzipiertes Peripheriegerät, mit dem Umrisse oder Zeichnungen abgetastet und auf dem Bildschirm dargestellt werden können. Sind die Formen erst einmal digitalisiert, können sie auch problemlos auf Disketten oder Cassetten abgespeichert werden. Das wichtigste im Tracer ist ein beweglicher Metallarm mit zwei Gelenken. Diese Gelenke sind mit Potentiometern versehen, die den aktuellen Stellwinkel von Ober- und Unterarm exakt messen.

Mit dem integrierten Analog-Digitalwandler des Acorn B werden die analogen Meßwerte digitalisiert. Sobald das Gerät kalibriert ist, können diese Werte nach einer mathematischen Umwandlung genau angeben, wo sich das Ende des Arms im Verhältnis zum Schultergelenk gerade befindet.

Unsere Bilder zeigen die wichtigsten Konstruktionselemente: Als Basis des Geräts dient eine quadratische Platte zum Auflegen der Zeichnung, die digitalisiert werden soll. Der Metallarm wird an einem kleinen Kunststoffge-



In unserem selbstgebauten Abtast-Arm leiten zwei Potentiometer ihre Meßwerte zum Analog-Port des Acorn B. Die Software ermittelt aus diesen Werten, über welcher Stelle der Grundplatte das Fadenkreuz gerade steht.

häuse befestigt, das mit einer 5poligen DIN-Buchse für das Kabel zum Analog-Eingang des Computers versehen ist. Mit einem Taster kann der Stift an der Armspitze gehoben und gesenkt werden.

Am Ende des Arms befestigen wir als „Visier" ein kleines Stück Acrylglas mit einem Fadenkreuz darauf. Es wird verstellbar angebracht, damit man auch beim Abtasten unterschiedlich dicker Materialien arbeiten kann.

Bauteilliste

Anzahl	Bauteil
2	100k Ohm – Potentiometer (linear)
2	Drehknöpfe, 40 mm Durchmesser
1	Taster (Schließer)
1	Kunststoffgehäuse, 114 mm x 76 mm x 38 mm
1	5-poliger DIN-Stecker
1	5-polige DIN-Buchse
1	Meter 10-poliges Flachbandkabel
1	Paket Schrauben M5
1	Paket Schrauben M3
1	Paket Muttern M3
1	Paket selbstschneidende Schrauben
1	15-poliger D-Stecker
1	Gehäuse für D-Stecker

Verschiedenes

1 Aluflachprofil, 600 mm x 25 mm x 2,5 mm*
1 Melaminharzbeschichtete Holzplatte 460mmx460mm**
1 Kleines Stück transparenten Kunststoff***

Anmerkungen

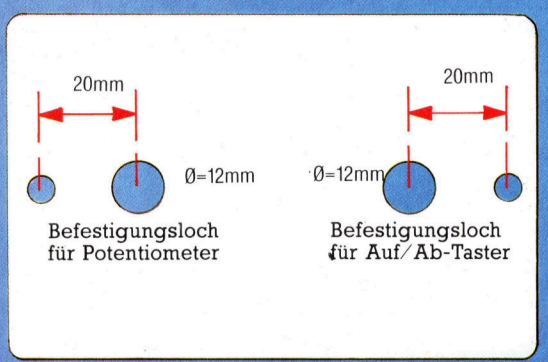
* In den meisten Heimwerkermärkten und im Eisenwarenhandel erhältlich. Als Ersatz können Sie auch Holzleisten verwenden.

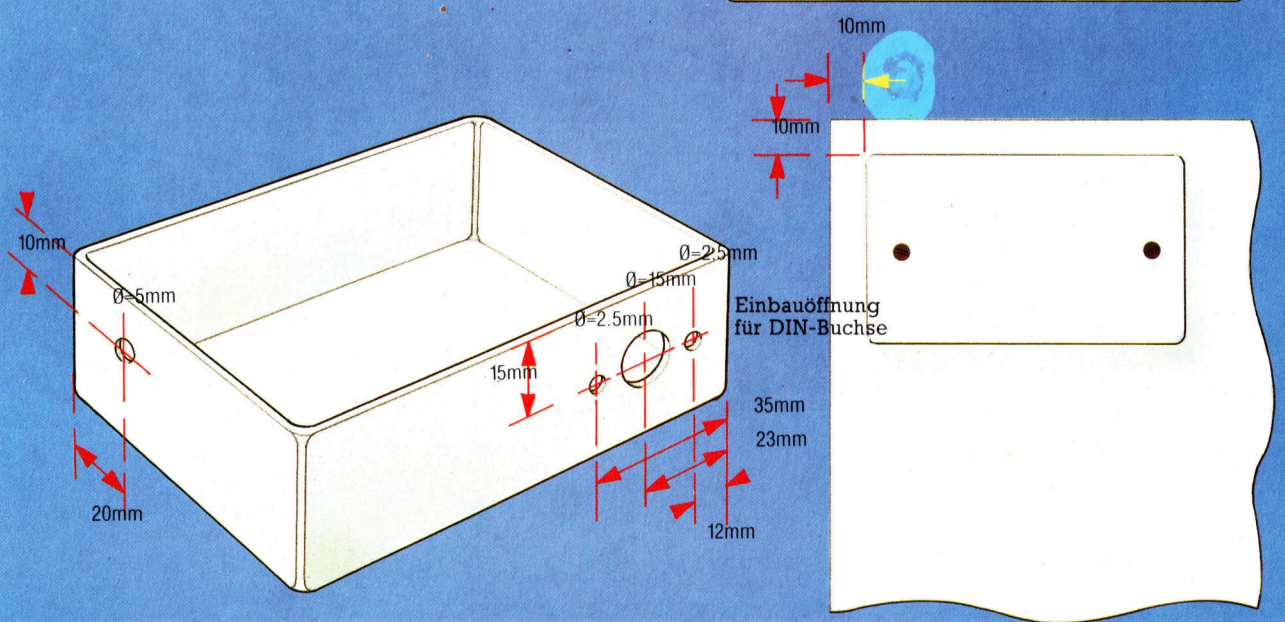
** Wird in vielen Bau- und Heimwerkermärkten und auch beim Tischler auf Wunsch zugeschnitten.

*** Sie können dafür den durchsichtigen Teil einer Cassettenhülle verwenden.

Erster Schritt: Gehäuse zuschneiden

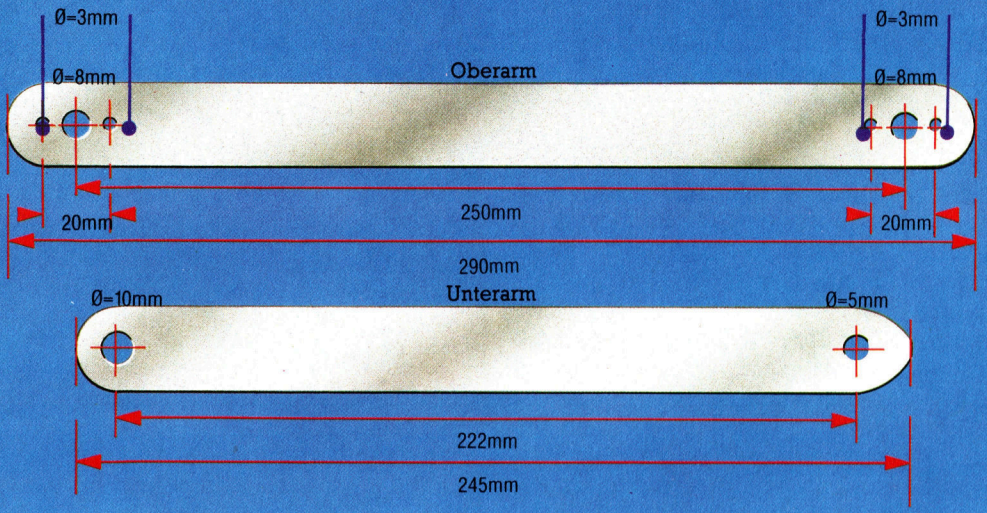
Der erste Schritt zur Konstruktion des Tracers ist die Anfertigung des Gehäuses. Das Kästchen muß mit mehreren Löchern versehen werden, in denen später der Arm selbst sowie der Taster und die DIN-Buchse befestigt werden. Nach dem Bohren der Löcher wird das Gehäuse in der linken oberen Ecke der Grundplatte festgeschraubt. Zum Rand der Platte sollte ein Abstand von 10 mm gehalten werden.





Zweiter Schritt: Armteile ausschneiden

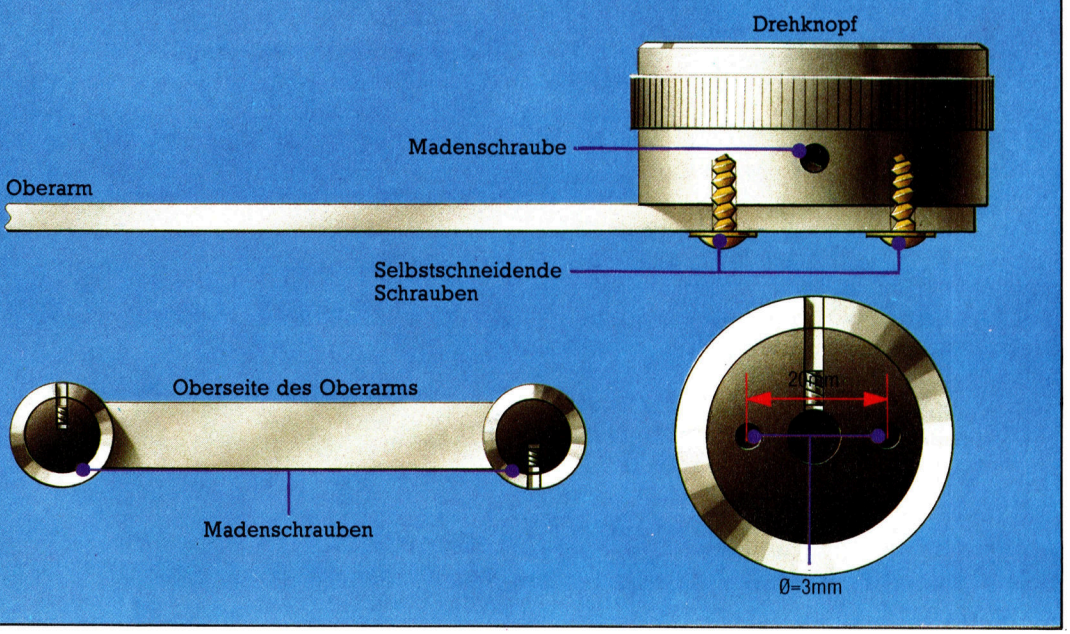
Das Aluprofil wird auf die angegebenen Längen geschnitten und mit den entsprechenden Bohrungen versehen. Die Gesamtlänge ist unwichtig, der Abstand zwischen den Bohrungen sollte aber sehr genau eingehalten werden. Dieser Abstand spielt bei der Software-Entwicklung noch eine wichtige Rolle – er geht als entscheidender Faktor in die geometrischen Berechnungen zur Ermittlung der Fadenkreuz-Position ein. Vor dem Bohren also genau messen und exakt anzeichnen!



Dritter Schritt: Drehknöpfe befestigen

Die letzte Aufgabe dieses Abschnitts ist die Befestigung der zwei Drehknöpfe (Lautstärkeregler) rechts und links am „Oberarm". Sie stellen die Verbindung zu den Potentiometern her, die im nächsten Bauabschnitt montiert werden sollen. Die Achse der Potis wird durch kleine Madenschrauben festgeklemmt, die seitlich in den Drehknöpfen liegen. Achten Sie bei der Montage der Knöpfe darauf, daß die Madenschrauben in gegensätzlicher Richtung liegen.

Zum Befestigen von unten je zwei 3-mm-Löcher in die Drehknöpfe bohren. Danach können sie mit selbstschneidenden Schrauben (auf die Orientierung achten!) am Oberarm montiert werden. Im nächsten Kursabschnitt vervollständigen wir den mechanischen Aufbau des Digitalisierarms und stellen die Verbindung zum Acorn B her.



Die Erfolgsformel

Im 16. und 17. Jahrhundert wurden Muße und Wissenschaft noch auf eine Weise verquickt, die in unserem modernen Zeitalter der Spezialisten unverantwortlich erscheinen würde. Kein Wunder, daß die Mathematik und die Spielleidenschaft damals eine fruchtbare Ehe eingehen konnten. Der oben abgebildete Pfarrer Bayes entwikkelte eine Gleichung für die Berechnung von Gewinnchancen. Viele seiner berühmten Zeitgenossen – D'Alembert, Pascal und Fermat – haben sich ebenfalls mit den möglicherweise profitablen Rätseln der Wahrscheinlichkeit beschäftigt.

Mit Gleichungen zur Berechnung der Wahrscheinlichkeit einer Hypothese können Fußballergebnisse vorhergesagt werden – natürlich wie immer ohne Gewähr. Viel Glück!

Sie können Ihren Computer zur Errechnung einer bedingten Wahrscheinlichkeit einsetzen – ein Verfahren, das den Buchmachern sicher gutes Geld einbringt. Eigentlich wollen wir mit dem Computer aber den Gewinner herausfinden – ohne eine Bewertung der aktuellen Form geht das nicht, weil einfach zu viele Informationen erfaßt werden müssen. Dabei gibt es zwei Schwierigkeiten:

- Die Unterscheidung von wichtigen und unwichtigen Bedingungen bzw. Vorinformationen.
- Die Verknüpfung nicht vergleichbarer Bedingungen bzw. Vorinformationen.

Hier hilft eine Methode aus der Statistik weiter – die nach einem Pfarrer des 18. Jahrhunderts benannte „Bayessche Formel".

P(H E) = P (E H) x P (H) / P(E)

die wir für unsere Zwecke so schreiben wollen:

O (H E) = O (H) x LR (H E)

Das sieht komplizierter aus, als es tatsächlich ist. P (H E) in der ersten Gleichung ist die Wahrscheinlichkeit P einer Hypothese H, unter der Voraussetzung, daß das Ereignis E bereits eingetreten ist.

O (H E) in der zweiten Gleichung ist dasselbe wie P (H E), wird aber hier als Verhältnis (zugunsten) ausgedrückt. Die zweite Gleichung liest man so: Die Chancen einer Hypothese H unter der Bedingung E sind gleich der früheren Wahrscheinlichkeit (bevor E bekannt ist) der Hypothese multipliziert mit dem Wahrscheinlichkeitsverhältnis der Bedingung. Wahrscheinlichkeitsverhältnisse werden wir sofort erklären, versuchen wir es zuerst einmal mit einem Beispiel.

Fußball zum Beispiel

Nehmen wir an, Sie hätten eine Liste mit Fußballergebnissen und möchten Heimsiege voraussagen. Zwei Punkte gelten als unumstritten: Wenn die Sportzeitungen einen Auswärtssieg vorhersagen, gibt es selten einen Sieg der Gastgebermannschaft – höchstens ein Unentschieden. Zum zweiten gewinnt die gastgebende Mannschaft fast immer, wenn sie zu den ersten neun Mannschaften der Liga gehört. Eine Auswertung von 131 Spielen erbrachte diese Verteilung:

Bedingung	Ergebnis	
	Heim-siege	Auswärts-siege
Zeitungen tippen Auswärtssieg	3	23
Zeitungen sagen Auswärtssieg nicht voraus	60	45
Gesamt	63	68

Bedingung	Ergebnis		
	Heim-siege	Auswärts-siege	Gesamt
Gastgeber unter den 9 Ligaersten	31	25	56
Gastgeber nicht unter den 9 Ersten	32	43	75
Gesamt	63	68	131

Mit diesen Verteilungen können wir das Wahrscheinlichkeitsverhältnis ermitteln, das meist so definiert wird:

LR (H E) = P (E H) /P (E nicht — H)

Für 2*2 – Tabellen (siehe oben) können wir die Wahrscheinlichkeitsverhältnisse mit dieser Tabelle berechnen:

Bedingung	Ergebnisse		
	Hypothese	nicht-Hypothese	Gesamt
	a c	b d	(a+b) (c+d)
Gesamt	(a+c)	(b+d)	(a+b+c+d)

P(E H) ist die Zahl der Ergebnisse, die das Verhältnis von Bedingung/alle Ergebnisse zugunsten der Hypothese stützen, also P (E H)= a/(a+c). Ähnlich P (E H)= b/ (b+d). Unser Wahrscheinlichkeitsverhältnis LR(H E) = a/(a+c)/b/(b+d) kann umgestellt werden zu:

LR(H E) =(a * (b +d)/b * (a+c))

Für unsere zwei Zabellen ergibt sich:

Bedingung	LR(Heimsieg Bedingung)
Zeitungen sagen Auswärtssieg voraus	0.1408
sagen keinen Auswärtssieg voraus	1.4392
Bedingung	**LR(Heimsieg Bedingung)**
Gastgeber unter den 9 Ligaersten	1.3384
Gastgeber nicht unter den Ersten	0.8032

Für den, der Worte den nackten Zahlen vorzieht: Das letzte Ziffernpaar bedeutet, daß die Vorhersage eines Sieges mit 1,3384-fach größerer Sicherheit gemacht werden kann, wenn man weiß, daß der Gastgeber zu den ersten neun Mannschaften seiner Liga gehört. Ist das Fußballteam aber nicht unter den ersten neun, sinkt die Chance auf nur 0,8032.

Nehmen wir einmal an, daß die zweite Bedingung erfüllt ist, die erste aber nicht. Wir beginnen mit der Grundchance zugunsten eines Heimsieges. Sie liegt bei 63/68= 0,9265 (Heimspiele/Auswärts- bzw. unentschiedene Spiele). Damit ergibt sich eine Wahrscheinlichkeit von 0,4809. Als nächstes wird mit den entsprechenden Wahrscheinlichkeitsverhältnissen multipliziert. Die erste Bedingung war nicht erfüllt, wir verwenden also 1,4392; die zweite war erfüllt, also nehmen wir 1,3384. Daraus ergibt sich:

Nachträglich berechnete Chance
=0,9265 x 1,4392 x 1,3384 = 1,7846
Mit der Formel P=F/(1+F) rechnen wir in Wahrscheinlichkeit um:
Nachträglich berechnete Wahrscheinlichkeit
=1,7846/2,7846 = 0,6409

Heimsieg fast sicher

Die Gewißheit eines Heimsiegs liegt also bei etwa 64 Prozent, vorausgesetzt, daß die Fußballzeitungen keinen Auswärtssieg prophezeihen und die Gastgeber unter den ersten 9 der Liga plaziert sind. Beide begünstigenden Bedingungen haben die Wahrscheinlichkeit von 48 auf 64 Prozent gesteigert. Sollte also eine Wette mit der Quote 6:4 oder besser angeboten werden, kann man sie getrost annehmen.

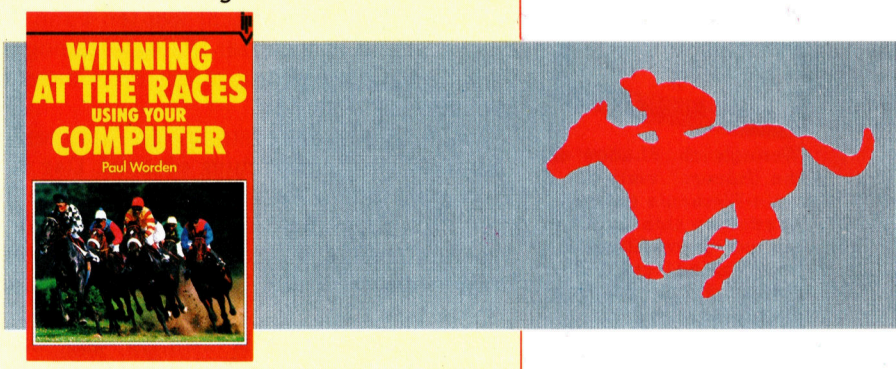
Bayes' Regel bietet den Vorzug, auch für die Bewertung und Zusammenfassung unterschiedlicher Vorbedingungen eine wissenschaftliche Grundlage zu bieten – dadurch kann sie auch beim Computereinsatz nützliche Dienste leisten. Hat man die Wahrscheinlichkeitsverhältnisse einmal ermittelt, bleibt eine reine Rechenaufgabe übrig. Man kann relativ leicht ermitteln, ob eine angebotene Wette günstig oder ungünstig ist – magische Zahlen und seltsame Bewertungen gibt es bei diesem System jedenfalls nicht.

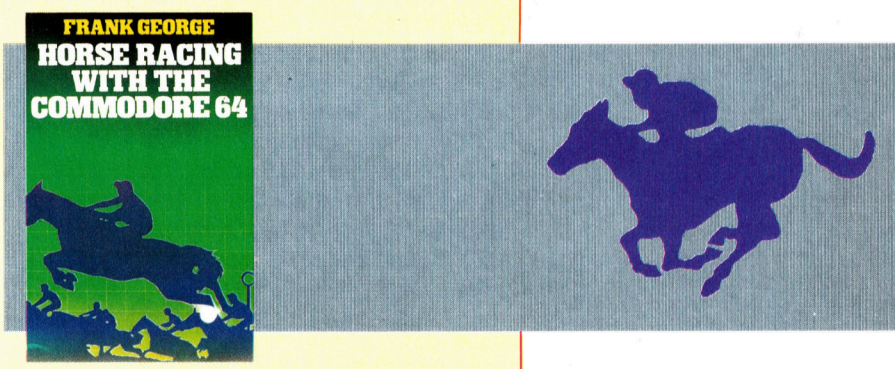
Ein Problem der Methode ist, daß sie bei korrelierten Bedingungen zu verfälschten Ergebnissen führt. Der Sportjournalist, der für die Fußballzeitung schreibt, berücksichtigt natürlich bei seiner Voraussage auch, daß eine Mannschaft unter den ersten neun Teams der Liga ist – damit sind die beiden Bedingungen nicht mehr unabhängig voneinander, was sie wegen der Multiplikation aber sein sollten. Trotzdem ist Bayes' Formel ein guter Weg zur Verbindung unterschiedlicher Indikatoren in einem Vorhersage-Programm. Wenn Sie versuchen, ein solches Programm selbst zu erstellen, sollten Sie aber neu hinzukommende Datengruppen (Bedingungen) einzeln integrieren und sie nur dann dauerhaft in das Programm einbauen, wenn sich dessen Leistungen dadurch merklich verbessern. Falls eine zusätzliche Variable aber die Genauigkeit der Vorhersage mindert, ist es sehr wahrscheinlich, daß die gleiche Bedingung in unterschiedlicher Form doppelt gemessen und bewertet worden ist.

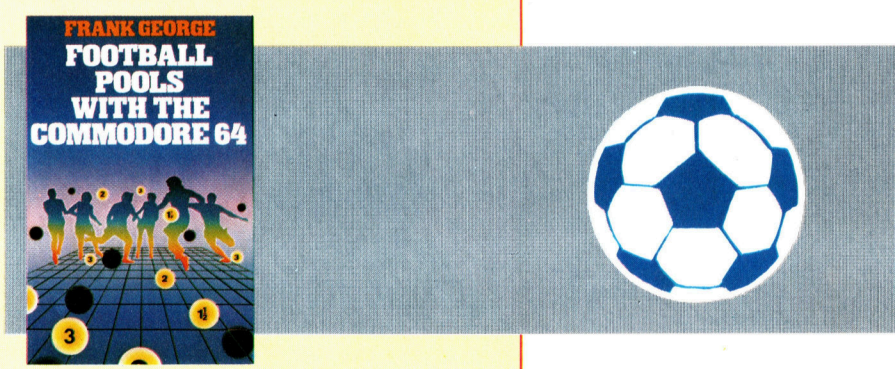
Der gesunde Menschenverstand in Verbindung mit einigen grundlegenden statistischen Kenntnissen kann Ihnen helfen, Fortuna auf Ihre Seite zu bringen.

Zusätzliche Literatur

In England, wo das Wetten auf Pferde, Fußball und andere Sportarten weiter verbreitet ist als bei uns, sind eine Reihe von Büchern zum Thema Wetten und Computer herausgekommen.
„Winning at the Races" ist erhältlich über den Verlag Interface Publications, 9–11 Kensington High Street, London W8. Die beiden anderen Bücher von Professor Frank George sind im Collins-Verlag erschienen.



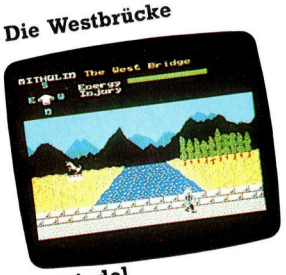




Gesicherte Zwerge

Das illegale Kopieren von Software bringt den Softwarehändlern jährlich Verluste in Millionenhöhe. Mikro-Gen's „Shadow of the Unicorn" bedient sich der „dongle"-Technologie, von der die Industrie größere Sicherheit erwartet.

Die Westbrücke



Vor Olindel

Wenngleich „Shadow of the Unicorn" viele Arcade-Elemente enthält, handelt es sich im Prinzip um ein traditionelles Abenteuerspiel. Der Spieler steuert mehrere Darsteller, die nach Hinweisen suchen, um schließlich die Mächte des Bösen zu besiegen.

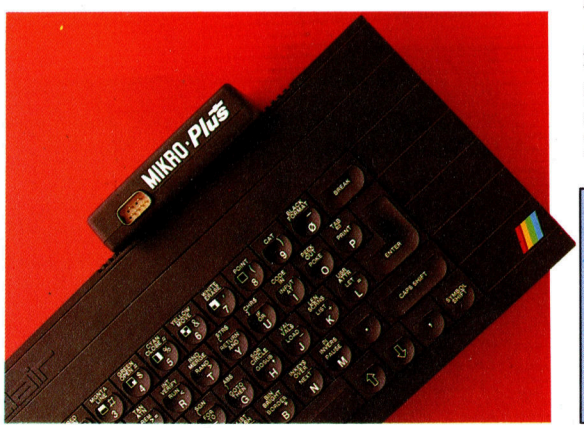
Eines der größen Probleme für Softwarehäuser ist die Programmpiraterie. Die betroffenen Unternehmen schätzen den dadurch entstehenden jährlichen Verlust auf mehrere Millionen Mark. Das gefährdetste Trägermedium für Programme ist die Cassette, die – zum Leidwesen der Software-Häuser – ausgesprochen leicht zu kopieren ist.

Die Idee des „dongle", eines Hardware-Zusatzteils, das in den Computer gesteckt werden muß, existiert seit geraumer Zeit. Das „dongle" enthält Programmteile, ohne die das Hauptprogramm (auf Diskette oder Cassette) nicht zum Laufen gebracht werden kann. Problematisch ist jedoch der Preis der „dongles": Ihre Herstellung ist teurer als die Cassettenproduktion.

Mikro-Gen ist der erste Spiele-Anbieter, der „dongles" als Kopierschutz einsetzt. Das erste Programm dieser Art heißt „Shadow of the Unicorn". Ein Programmteil befindet sich wie üblich auf Cassette, der „dongle"-Teil wird in den Erweiterungsport des Spectrum gesteckt und von dort geladen.

Das „dongle" selbst besteht aus einem 16KByte EPROM, einem Joystick-Anschluß und einem Joystick Decoder-Chip. Auf dem EPROM sind die Laderoutinen für die Cassette sowie im Spiel verwendete Grafik-Routinen untergebracht. Nach dem Einschalten wird der EPROM-Inhalt in jenen Speicherbereich gebracht, der normalerweise vom BASIC-ROM belegt ist. Damit steht für das Programm zusätzlicher Platz zur Verfügung.

Es handelt sich dabei um ein Abenteuerspiel mit Arcade-Elementen – vergleichbar den „Lords of Midnight". Die Story spielt in den Königreichen Oronfal und Falforn. Ziel ist es, die bösen Mächte, die das Land bewohnen, zu bezwingen. Zu Beginn steuert der Spieler drei Darsteller: Mithulin (König von Oronfal), Ulin-Gail (einen Satyr) und Avarath (einen Zauberer). Dazu kommen im Spielverlauf mehrere andere Charaktere, die nach ihrem Eintritt ins Geschehen ebenfalls vom Spieler zu führen sind. Einer von ihnen ist Holdin, der Kapitän von Falforn, der das Spiel an der gleichen Position wie der Zauberer beginnt und so sofort steuerbar ist.



Das Mikro-Plus „dongle", mit dem „Shadow of the Unicorn" geliefert wird, bietet dem Spieler zusätzliche 16KByte Speicherkapazität, wodurch Umfang und Qualität des Spiels verbessert werden. Das Mikro-Plus enthält ferner ein Joystick-Interface, da das Interface 2 mit dem „dongle" nicht kompatibel ist.

Zwerge kosten Energie

Während des Spiels werden der „Energie"- und der „Verletzungs"-Grad des Akteurs auf dem Bildschirm oben als Balken dargestellt. Eine Auseinandersetzung mit den Zwergen kostet Energie, und der Verletzungsgrad wächst, sobald ein Zwerg den Darsteller erreicht hat. Als Energieschub und Nahrungsquelle dienen die vielen Früchte, die überall im Königreich wachsen. Verletzungen indes heilen zwar von alleine, aber nur langsam.

Während die Akteure sich im Königreich bewegen, gelangen sie an die verschiedensten Gebäude, in die jedoch nicht immer leichtes Hineinkommen ist. Der Versuch, durch eine Tür zu gehen, hat oftmals zur Folge, daß die Szene wechselt. Der Akteur steht plötzlich auf der anderen Seite des Gebäudes. Natürlich gibt es Mittel und Wege, die Gebäude zu betreten. Aber eben nur auf eine Art.

Wie andere Abenteuer wird auch „Shadow of the Unicorn" mit einem ausführlichen Begleitbuch geliefert, das viele Lösungshinweise für das Programm enthält.

Mikro-Gen hat sich bemüht, das Programm für die Spieler so attraktiv wie möglich zu gestalten. Es bleibt abzuwarten, ob es am Markt akzeptiert wird und das Problem Software-Piraterie so zu lösen ist.

Shadow of the Unicorn: Für Sinclair Spectrum
Hersteller: Mikro-Gen, Unit 15, Western Centre, Bracknell, Berks
Autor: Dale McLoughlin
Steuerknüppel: Option
Format: Cassette und EPROM

Fachwörter von A bis Z

Mail Box = Briefkasten
Ein (elektronischer) Briefkasten ist ein Speicherbereich im Computer zur Aufnahme von Daten, die als „elektronische Post" von einem Computer zum anderen übermittelt werden. Die Sendestation versieht die Nachricht mit der „Adresse" des Empfängers, unter der die Information dann bis zum Abruf verbleibt. Der Inhalt der Mail Box ist gewöhnlich gesichert; vor der Freigabe einer Nachricht muß sich der Empfänger durch ein Paßwort ausweisen. Findet er nach dem Einloggen ins System eine für ihn bestimmte Nachricht vor, kann er sie zum eigenen Computer übertragen und ausdrukken oder auf Diskette abspeichern.

Großrechner sehen heute in der Realität selten wie die raumfüllenden Ungetüme aus, die aus Science-fiction-Filmen bekannt sind. Der berühmte Cray-1, ein Supercomputer der neueren Generation, ist in einer Anzahl von handlichen kleinen Schränken untergebracht, die der arglose Betrachter zunächst eher für Möbelstücke halten würde.

Mainframe = Großrechner
Vor der Microprozessor-Revolution waren Computer äußerst voluminöse und unhandliche Anlagen; die CPU und der Arbeitsspeicher wurden damals als Mainframe bezeichnet. In-

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.

zwischen gibt es Rechner der verschiedensten Größen- und Leistungsklassen, und die Bedeutung des Wortes hat sich verschoben.

Die Gruppe der Microcomputer reicht von den Taschenmodellen über die Handheld- und Portable-Typen bis zu den Schreibtisch-PCs. Dann kommen die „Minicomputer" – leistungsfähige Anlagen für Prozeßsteuerung oder Mehrbenutzerbetrieb, die heute meist mit 32-Bit-Prozessoren und einer Hauptspeicherkapazität von 20–40 Megabyte arbeiten. Systeme, die noch eine Stufe schneller sind und über mehr Speicherraum verfügen, werden gelegentlich als „Super-Minis" bezeichnet. Darauf folgen die Mainframes im neueren Sinn: Großrechneranlagen, die in kommerziellen Rechenzentren verwendet werden und etliche Millionen Mark kosten.

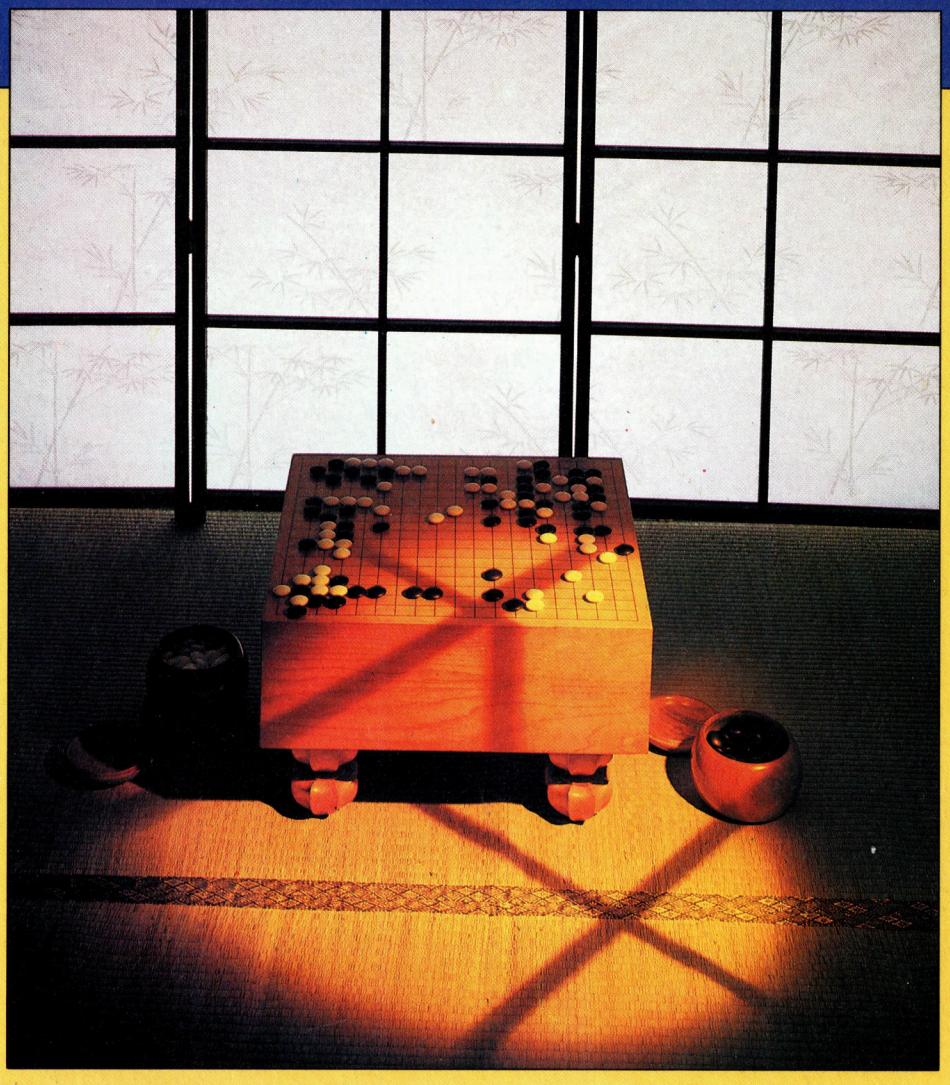
Der Begriff Mainframe wird dabei für die gesamte Anlage und nicht nur für CPU und Zentralspeicher verwendet. Umsatzgrößter Hersteller ist IBM, der 70% des Weltmarkts abdeckt und an dem sich die Konkurrenz orientiert. Die Mainframes sind jedoch heute nicht mehr die leistungsfähigsten Systeme.

Management Information System = Management-Informationssystem
Viele große Firmen leisten sich ein „Management-Informationssystem" (MIS), um die nötigen Informationen für Management-Entscheidungen zu sammeln und über Dialog-Auskunftverfahren zugriffsbereit zu halten. Diese Einrichtungen kamen mit dem Einzug der Datenverarbeitung in die Unternehmensführung Ende der sechziger Jahre in Gang. Beim Aufbau eines MIS muß zunächst ein Systemanalytiker die Firmenorganisation unter die Lupe nehmen, um festzustellen, welche Daten jeder Manager benötigt und wie er sie am besten auf Knopfdruck abrufen kann. Wenn das MIS einmal läuft, wird es von einer speziellen MIS-Abteilung betreut, die die Informationen aufbereitet und verwaltet.

Eine Variation des MIS ist das „Decision Support System" (Entscheidungshilfesystem), das mehr Flexibilität bietet. Dabei muß sich jeder Manager seine individuelle Datenbank aufbauen, indem er nach Bedarf Informationen aus dem Zentralrechner der Unternehmen übernimmt. Die Decision Support Systems sind erst durch den vermehrten Einsatz von Arbeitsplatzrechnern möglich geworden, und eine wachsende Anzahl von Entscheidungsträgern macht inzwischen davon Gebrauch, um sich unmittelbar mit „Rohdaten" zu versorgen.

Mass Storage = Massenspeicher
Massenspeicher sind periphere Speicher zur Aufnahme umfangreicher Dateien, deren Platzbedarf die RAM-Kapazität des Rechnersystems übersteigt. Während früher eine Massenspeichergröße von einem Megabyte schon als durchaus ansehnlich galt, haben heute Micros wie der IBM PC oder der Apricot bereits Festplatten mit über zehn Megabyte zu bieten. Zu den Massenspeichern im eigentlichen Sinn zählt beispielsweise das automatisierte Magnetbandsystem 3850 von IBM, das bis zu 472 Gigabyte aufnehmen kann.

Bildnachweise
1793: Liz Heaney
1795, U4: Marcus Wilson-Smith
1800: Mike Cowles
1800–1802: David Vymer
1801, 1810, 1811: Caroline Clayton
1807: Chris Stevens
1808, 1809: Ian KcKinnell
1815–1817: Kevin Jones



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

computer kurs

Go in Prozeduren

Um das Brettspiel Go zu programmieren, beginnt man am besten mit den Prozeduren.

Der Prozessor 6510

Praktische und ungewöhnliche Aspekte hat das Betriebssystem des Commodore 64. Wir zeigen einmal eine genaugehende Bildschirmuhr.

Vom Plan zur Tat

Die zwei populärsten Programmgeneratoren SYCERO und THE LAST ONE.

Der Jet Set

Der führende Matrixdrucker-Hersteller für Microcomputer ist Epson, sein Spitzenmodell SQ2000 – ein Tintenstrahldrucker.