Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

# computer kurs

Die Welt der Roboter

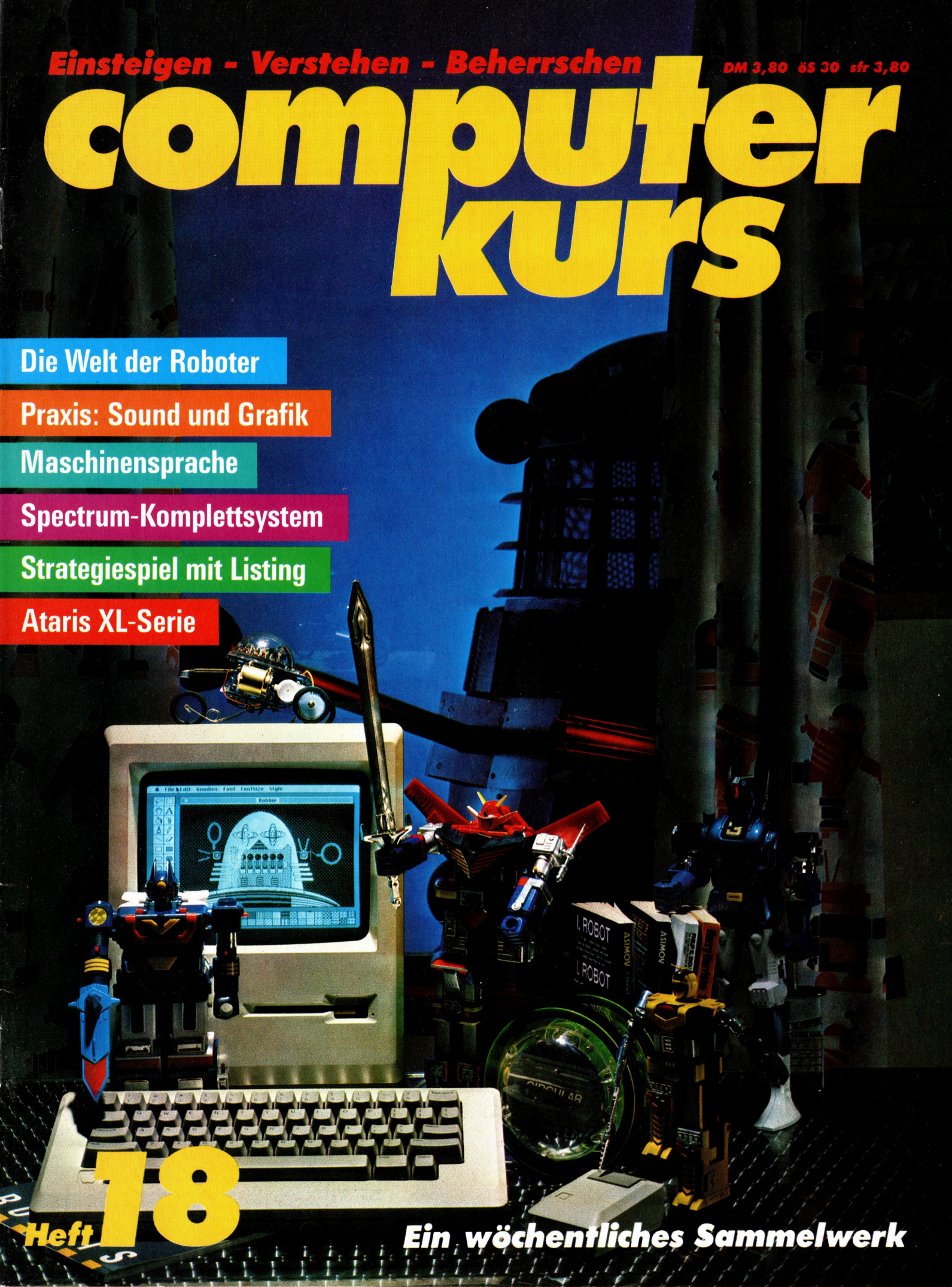
Praxis: Sound und Grafik

Maschinensprache

Spectrum-Komplettsystem

Strategiespiel mit Listing

Ataris XL-Serie




Heft 18

Ein wöchentliches Sammelwerk

# computer kurs

## Heft 18

## Inhalt


### Computer Welt

**Enten und Androiden** 477
Beginn einer kleinen Reihe über Roboter

**Zukunftsmusik** 490
Der Heimcomputer in den neunziger Jahren

**Vannevar Bush** 498
Der Erfinder des Differential-Analysators

### Software

**Ameisen greifen an** 480
Quicksilvas dreidimensionales Labyrinthspiel

**Lehren und Lernen** 499

**Durch die Wüste** 504
Ein Strategiespiel mit Listing

### LOGO 18

**Kollisionen** 482
Sprite-Überschneidungen beim Atari-LOGO

### Hardware

**Ataris XL-Serie** 485

### Tips für die Praxis

**Klanggebäude, Lichtwellen** 488
Acorn B und Atari-Heimcomputer im Einsatz

### BASIC 18

**Montage der Module** 492
Das „Adreßbuch" wird anwenderfreundlich

### Peripherie

**Spectrum-Komplettsystem** 496

### Bits und Bytes

**Interne Funktionsabläufe** 502
Die Maschinensprache des 6502 und des Z80

**Fachwörter von A—Z**


## 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 oS 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. Bei Bestellungen aus Österreich oder Schweiz senden Sie Ihren Auftrag bitte auch an die Hamburger Adresse. Berechnung und Zahlung erfolgen in Landeswährung zum Ladenpreis.

**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.

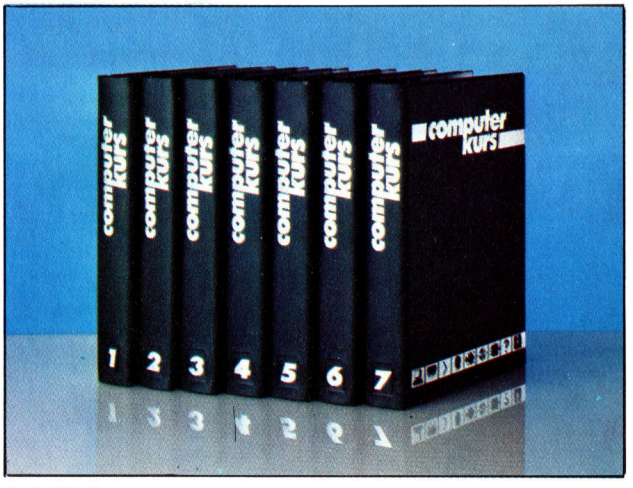
**Schweiz:** Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk, dort werden Sie jederzeit die gewünschten Exemplare erhalten.

## 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), Joachim Seidel, Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

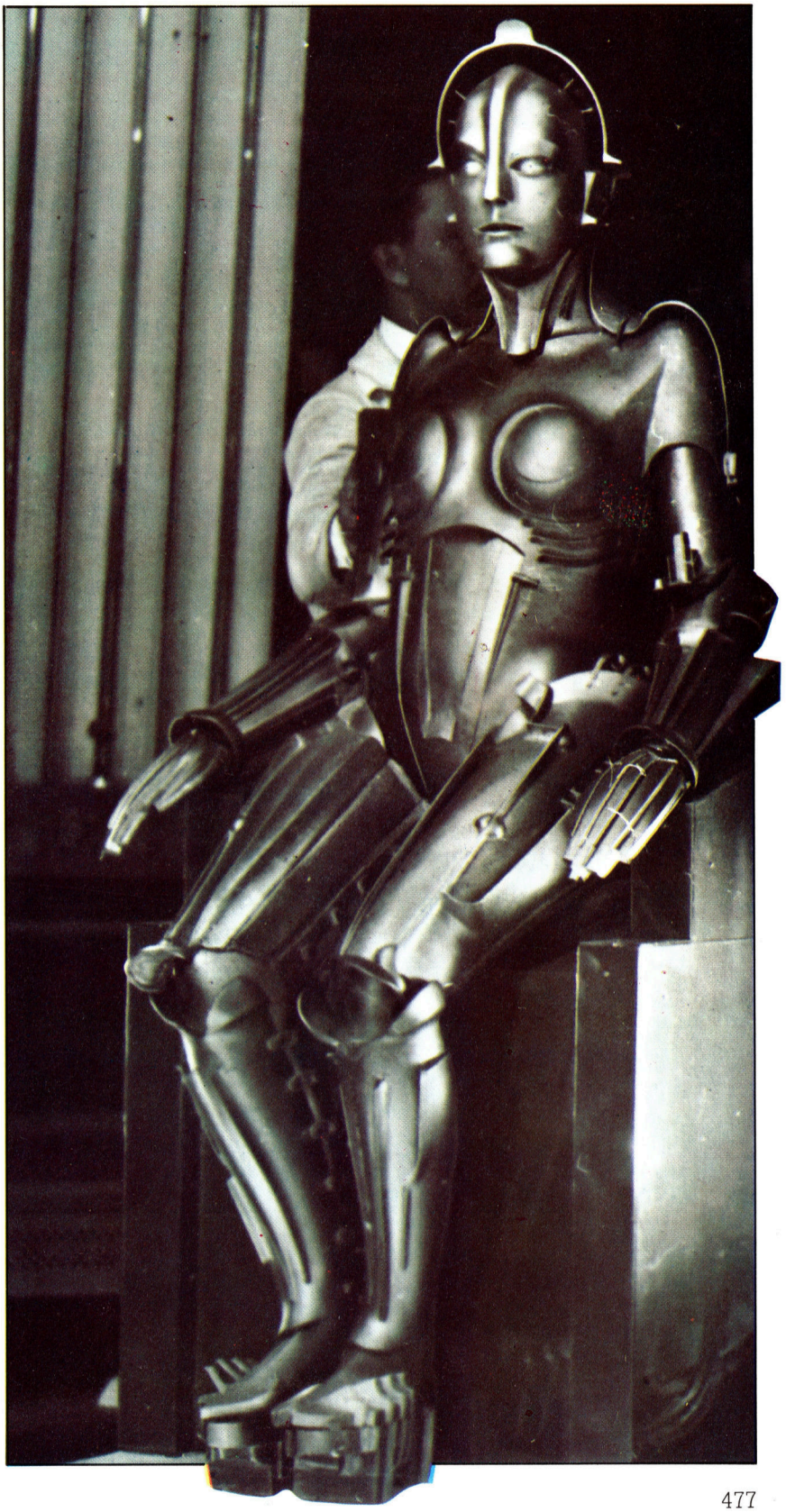
**Vertrieb:** Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1, Tel.: 040/23 40 85




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

# Enten und Androiden



Fritz Langs 1926 gedrehter Filmklassiker „Metropolis“ beeinflußte Filmschaffende und Zuschauer über Jahrzehnte. Nicht zuletzt, weil er mit „Die Maschine“ den ersten Roboter-Star des Kinos schuf.

**Mit diesem Beitrag werden Entwicklung und Hintergründe der Robotik aufgezeigt, beginnend mit den mechanischen Anfängen im 18. Jahrhundert bis hin zu den Industrierobotern von heute.**

Seit Hunderten von Jahren beschäftigen sich Menschen mit der Idee des mechanischen Menschen in dieser oder jener Form. Philosophen, Ingenieure und Erfinder versuchten, Maschinen mit menschlichen Verhaltensweisen zu schaffen. Wenngleich die heutigen Roboter kaum menschenähnlich zu nennen sind und nur für ganz bestimmte Aufgaben konstruiert wurden, gestaltete man die ersten künstlichen Menschen so „echt“ wie möglich, um sie so zu befähigen, jede menschliche Handlung vollbringen zu können.

Der erste mechanische Roboter hatte allerdings keine Menschengestalt. 1738 stellte der französische Ingenieur Jacques de Vaucanson (1709–1782) der Akademie der Künste in Paris eine mechanische Ente vor. Diese konnte die Flügel bewegen, quaken und Körner picken. Gegen Ende des 18. Jahrhunderts schuf der Schweizer Erfinder Pierre Jacquet-Droz (1721–1790) eine Reihe mechanischer Puppen, die in der Lage waren, verschiedene Dinge zu verrichten. Eine schrieb, die andere zeichnete Figuren, und eine dritte spielte Orgel. Ende des 19. Jahrhunderts gab es eine Vielzahl solcher Automaten, die durchweg mit Uhrwerken betrieben wurden.

## Der „Elektro“-Mann

In der Viktorianischen Zeit wurden bemerkenswert echte Automaten geschaffen, bei denen auch andere Antriebsmechanismen Anwendung fanden. So baute 1893 George Moore einen mechanischen Menschen, der mit Dampfkraft bewegt wurde. Interessanter Nebeneffekt dieses Betriebssystems: Der mechanische Mann konnte eine Zigarre rauchen und schien Rauch auszublasen.

Neue Technologien begünstigten die Entwicklung leistungsfähigerer Maschinen: vom einfachen Roboter, der aus Teilen eines „Meccano“-Baukastens montiert wird und laufen kann, bis hin zum klassischen „Elektro“-Mann, den das amerikanische Unternehmen Westinghouse bauen ließ. Der mechanische

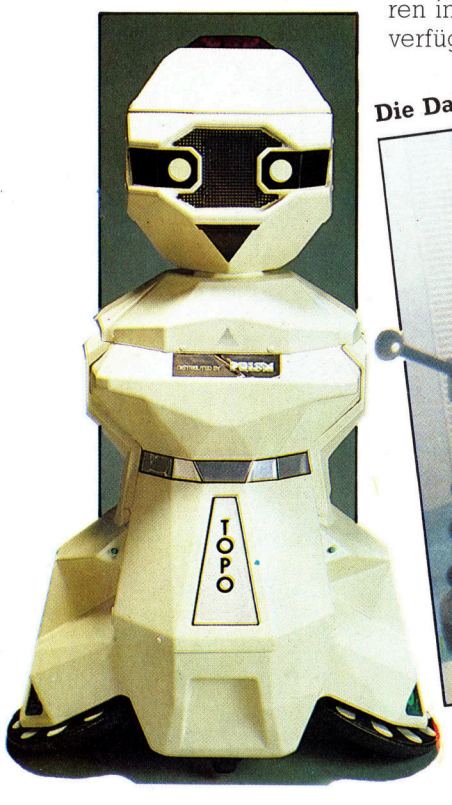
## Was ist ein Roboter?

In der nebenstehenden Übersicht wurde „Roboter" definiert als „eine Maschine, die bestimmte menschliche Tätigkeiten verrichten kann, obwohl sie nicht wie ein Mensch aussehen muß". Diese Definition ist weit gesteckt und könnte z. B. auch auf Computer übertragen werden (da diese ja menschliche „Rechenfunktionen" wahrnehmen). Nach dem allgemeinen Verständnis aber sollte ein Roboter über gewisse menschliche Qualitäten verfügen, wie etwa sich drehen oder gar laufen können. Er kann einen Arm haben, der mit einem menschlichen Arm identisch ist, oder auch optische und akustische Signale erkennen. Und er verfügt vielleicht sogar über eine gewisse Intelligenz.

Die genaue Form und die Fähigkeiten der Roboter hängen im wesentlichen von zwei Dingen ab: Was wir sie tun lassen wollen, und was man sie tun lassen kann! Ein schweißender Industrieroboter etwa kann sich nicht umherbewegen, nicht etwa, weil das technisch unmöglich wäre, sondern weil es aufgrund seiner Funktion nicht notwendig ist. Ein Haushaltsroboter dagegen mag zwar Tee kochen können, ist aber außerstande, die Treppe hochzusteigen und den Tee ans Bett zu bringen. – Vielleicht, weil es nicht möglich ist, einen Roboter zu bauen, der Treppen steigt, ohne den Tee zu verschütten!

Der Terminus „Roboter" ist zum Synonym für alle menschenähnlichen Maschinen geworden. Was sie sind und was sie können, hängt von denen ab, die sie konstruieren und bauen. Doch die Möglichkeiten nehmen täglich zu.

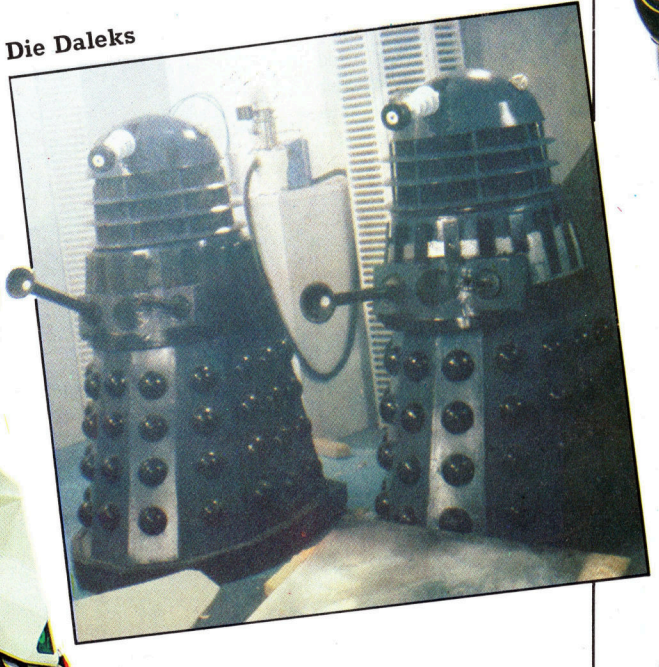
**Die berühmtesten Fernsehroboter sind die „Daleks", bewaffnete Ein-Personen-Fahrzeuge, die von ihren Erbauern gesteuert werden, die in ihnen sitzen, also keine „echten" Robots.**
**Der Roboter Robbie aus „Der Tag, an dem die Erde stillstand" verkörpert den sorgenden, einfühlsamen Roboter mit starker Menschenähnlichkeit.**
**Topo, der bis vor kurzem von Prism produzierte persönliche Roboter, war ein halb ernsthafter Versuch, den Roboter „ins Haus" zu bringen.**



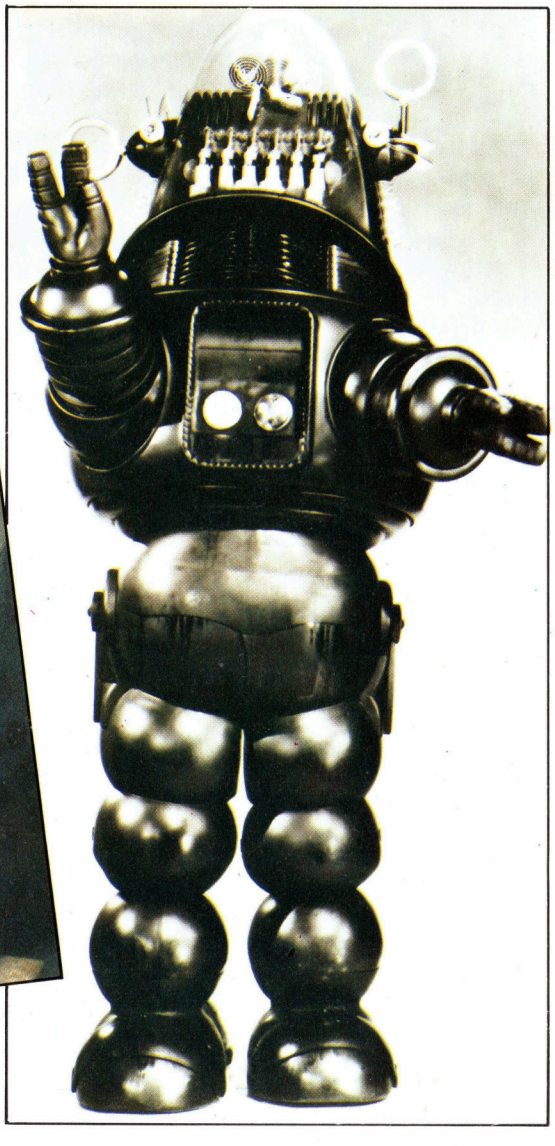

„Elektro"-Mensch war 2,15 Meter hoch, konnte 80 verschiedene Worte sprechen, zählen, gehen, grüßen und zwischen Farben unterscheiden. Elf Elektromotoren trieben das 117 Kilo schwere Geschöpf an. Das „Hirn", das den Giganten steuerte, setzte sich aus 82 verschiedenen Relais zusammen.

Doch all diese mechanischen Menschen waren in ihrem Tun eingeengt. Keiner von ihnen verfügte, abgesehen vom offensichtlichen Unterhaltungswert, über eine jener Eigenschaften, die man von einem idealen Roboter erwarten würde. Ein mechanischer Mensch, der Figuren zeichnet, kann eben nicht einkaufen, und ein mechanisches Wesen, das einen Raum durchqueren kann, wird nicht in den nächsten Laden gelangen, ohne zuvor gegen einen Laternenpfahl gestoßen zu sein. Denn es handelte sich bei diesen mechanischen „Menschen" um Maschinen, die für eine spezielle Funktion gebaut worden waren; Maschinen also, ohne jegliche Intelligenz.

Erfinder und Ingenieure steckten voller Ideen, ohne sie in die Realität umsetzen zu können. Solche kreativen Grenzen waren Schriftstellern nicht gesetzt. Gerade in der Science-Fiction entwickelte sich die Roboter-Idee weiter. Tatsache ist, daß das Wort „robot" das Ergebnis einer solchen Arbeit ist. Der tschechische Bühnenautor Karel Čapek (1879–1938) schrieb 1923 ein Stück mit dem Titel „R.U.R.", was eine Abkürzung für „Rossum's Universal Robots" war. Es ging darin um die Erfindung eines mechanischen Menschen, der so perfekt war, daß er jede Aufgabe erfüllen



**Die Daleks**



**Der Roboter Robbie**

## Roboter-Sprache

In der Literatur hat man den Robotern so viele verschiedene Namen gegeben, daß eine Übersicht der am häufigsten verwendeten sinnvoll scheint. Berücksichtigt werden sollte allerdings, daß es diese Roboter nicht unbedingt gibt, nur weil sie eine Bezeichnung haben.

**Android:** Ein Roboter, der in jeder Hinsicht wie ein Mensch aussieht.

**Anthropomorph:** Buchstäblich „menschlich". Ein Android ist in jeder Hinsicht menschlich. Viele Roboter sind aber nur teilweise menschenähnlich gestaltet. So können sie mit einem Arm ausgestattet sein, der wie ein menschlicher Arm aussieht.

**Automation:** Die automatische Kontrolle eines Herstellungsprozesses.

**Automat:** Eine Maschine mit begrenztem Aufgabenbereich für bestimmte Funktionen. Die ersten „mechanischen Menschen" waren Automaten. In Verbindung mit „Automaten-Theorie" hat das Wort eine mehr technische Bedeutung. Diese Theorie ist ein analytisches System, mit dessen Hilfe man jeden Gegenstand studieren und beschreiben kann – Roboter, Computer, aber auch ... Menschen.

**Cybert:** Die Fiktion eines rein mechanischen Humanoiden.

**Cybot:** Ebenfalls literarisch-fiktiv: ein Roboter mit geistig-menschlichen Fähigkeiten.

**Cyborg:** Ein CYBernetischer (kybernetischer) ORGanismus, der aus biologischen und mechanischen Teilen besteht.

**Doppelgänger:** Die exakte Nachbildung eines lebenden Menschen – gemeinhin ein Geist oder Gespenst.

**Droid:** Ein „guter" Roboter, der Asimovs „Drei Gesetze" befolgte.

**End effektor:** Heutiger Terminus für die „Hand" eines Roboters.

**Homunculi:** Kleine Menschen oder zwergenhafte Phantome.

**Kybernetik:** Das Studium von Kontroll- und Kommunikationssystemen. 1947 von Norbert Wiener entwickelt, besteht die Hauptaufgabe darin, biologische Systeme so zu studieren, als handle es sich um Maschinen.

**Manipulator:** Synonym für eine „Roboter-Hand".

**Mechanisierung:** Ersatz eines Arbeitsprozesses durch einen mechanischen Vorgang.

**Metal-Collar-Workers:** Industrieroboter. Menschliche Büroangestellte werden in englischsprachigen Ländern häufig „White Collar" (also „weißer Kragen") genannt, im Gegensatz zu den „Blue Collar"-Workers (wegen des blauen Arbeitszeuges der Arbeiter). Folglich sind die metallenen Industrieroboter „Metall-Kragen"-Arbeiter.

**Roboter:** Eine Maschine, die bestimmte menschliche Tätigkeiten verrichten kann, obwohl sie nicht wie ein Mensch aussehen muß.

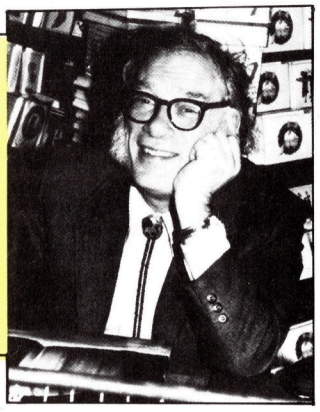
**Robotik:** Die Wissenschaft über Roboter.

konnte, die normale Menschen verrichteten. Schließlich entdeckten die Roboter, daß sie gar keine Menschen brauchten, was folglich für die Menschen Probleme aufwarf. Im Tschechischen bedeutet das Wort „robota" einfach „Arbeiter". Der Titel von Čapeks Stück hätte also mit „Rossums Universal-Arbeiter" übersetzt werden müssen, doch irgendwie blieb das Wort „Robot" erhalten und ist seitdem Inbegriff des künstlichen Menschen mit menschlichen Fähigkeiten.

Literarische Phantasien über künstliche, menschenähnliche Wesen haben lange Tradition. So schrieb Mary Shelley ihren berühmten Horror-Roman „Frankenstein" bereits 1818.

## Asimovs Gesetze der Robotik

**1.** Ein Robot darf keinen Menschen verletzen oder durch Untätigkeit zu Schaden kommen lassen.
**2.** Ein Robot muß den Befehlen des Menschen gehorchen – es sei denn, solche Befehle stehen im Widerspruch zum ersten Gesetz.
**3.** Ein Robot muß seine eigene Existenz schützen, solange dieser Schutz nicht dem ersten oder zweiten Gesetz widerspricht.



**Asimovs Robotik-Gesetze könnten Verhaltensgrundregeln schaffen, wenn Roboter einmal zur unabhängigen Funktion fähig wären. Die Roboter von heute können einen Menschen nicht erkennen, daher ist die Interaktion Roboter – Mensch irrelevant.**

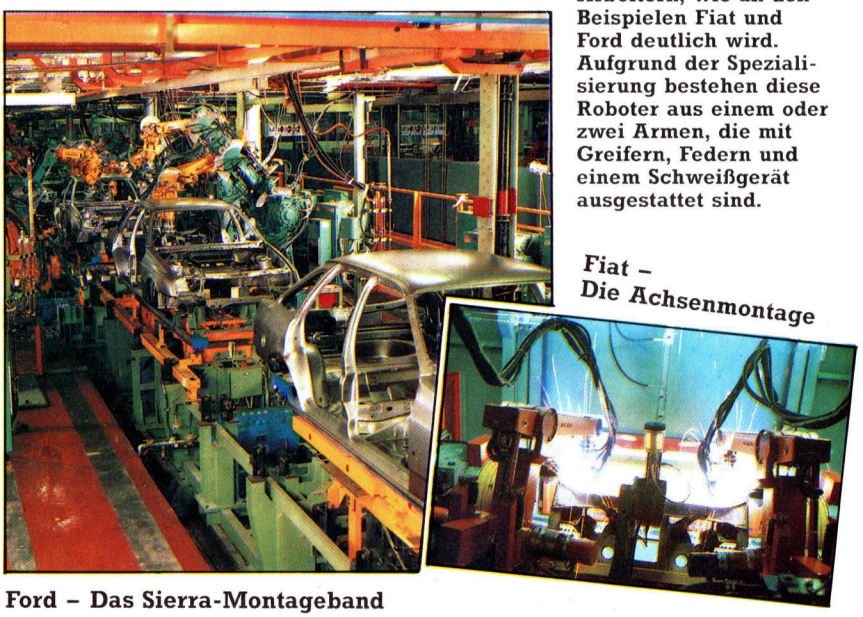
Wenngleich nicht mechanisch, so war dieses von Victor Frankenstein erschaffene Monster aus Teilen zusammengesetzt, die man sich durch Leichenfledderei beschafft hatte. Zumindest teilweise waren auch die Invasoren in H. G. Wells „Krieg der Welten" (1898) Roboter.

Die Autoren des 20. Jahrhunderts haben ungeheuer detailliert eine von Robotern bewohnte fiktive Welt geschaffen. Den wohl wichtigsten Beitrag dazu lieferte Isaac Asimov, der 1940 damit begann, Kurzgeschichten über Roboter und ihre imaginären Betriebs- bzw. Daseinsprobleme zu schreiben. Asimovs visionäre Roboter-Welt ist so komplex, daß er die „Drei Gesetze der Robotik" formulierte.

In Film und Fernsehen spielen Roboter eine ebenso bedeutende Rolle. Die Fernsehserie „Dr. Who" lebt von „Daleks" und „Cybermen", im „Krieg der Sterne" sind „C3PO" und „R2D2" ihren menschlichen Partnern ebenbürtig.

Verglichen mit diesen Phantasiegestalten mutet der Einsatz von Robotern heute recht banal an. Industrieroboter, wie sie bei der Automontage an Fließbändern Verwendung finden, spielen derzeit die wichtigste Rolle. Man schätzt, daß 1985 in Japan 25 000 Industrieroboter im Einsatz sind, 15 000 in den USA und 8000 in der Bundesrepublik. Indes rechnet man mit einer Expansion in Europa: 1990 werden mechanische Arbeiter im Wert von mehr als einer Milliarde Mark aufgestellt sein.

**Roboter werden vornehmlich an Fließbändern eingesetzt. Die Gesetze der Massenproduktion machen sie zu idealen Fließband-Arbeitern, wie an den Beispielen Fiat und Ford deutlich wird. Aufgrund der Spezialisierung bestehen diese Roboter aus einem oder zwei Armen, die mit Greifern, Federn und einem Schweißgerät ausgestattet sind.**



**Fiat – Die Achsenmontage**

**Ford – Das Sierra-Montageband**

# Ameisen greifen an

**Die Bedeutung von Quicksilvas „Ant Attack", einem dreidimensionalen Labyrinthspiel, das für den ZX-Spectrum mit 48K-RAM entwickelt wurde, liegt zum einen in der herausragenden Grafik-Qualität, zum anderen in der hervorragenden Umsetzung einer höchst komplizierten Programm-Idee.**

Sowohl Software-Autoren als auch Verleger sind mit der Urheberrecht-Gesetzgebung nicht zufrieden. Daraus resultieren die vielen verschiedenen Versuche, Programme gegen unerlaubtes Kopieren zu schützen. Der Autor dieses Spiels, Sandy White, versuchte sein Werk auf andere Art und Weise vor Plagiaten abzusichern, indem er ein Patent auf die darin verwandte Software-Technik anmeldete. Seit 1977 lehnt das britische Patentamt generell den Schutz von Computerprogrammen ab (mit dem Verweis, daß diese nicht als Erfindungen betrachtet werden können); man kam zu dem Ergebnis, da es sich bei dem fraglichen Patent um eine mathematische Formel bzw. einen Algorithmus handele.

Dies ist an sich interessant, da man für ein derartiges Spiel normalerweise keinen so komplexen Algorithmus benötigt. Was ist nun so Besonderes an „Ant Attack", daß man sich um einen völlig neuen Softwareschutz bemüht?

„Ant Attack" wirkt insofern ungewöhnlich, weil es kein Abklatsch eines Automatenspiels


**Zuweilen ist der Umstand, daß Ameisen keine Treppen steigen können, von Vorteil. Warum unser Held so hoch geklettert ist, sei dahingestellt. Das Erklettern von Gebäuden gibt dem Protagonisten die Chance, Granaten werfen zu können, ohne mit Gegenwehr rechnen zu müssen. Indes sollten Sie nicht vergessen, daß die Zeit unerbittlich gegen Sie läuft!**

**Im ersten Durchgang befindet sich das „Opfer" ständig in der Nähe zum Eingang der Stadt. Ein rascher Sprung über die schützende Mauer – und der männliche oder weibliche Protagonist wird mit den Worten „Mein Held – bring mich fort von hier" begrüßt.**

ist. Bekanntlich wurden die meisten populären Heimcomputer-Spiele aus Konzepten entwikkelt, die Atari, Taito und andere Automatenhersteller originär in speziellen Maschinen umsetzten. Mit „Ant Attack" verlieh der promovierte Student des Edinburgh College of Art seinem Protest gegen diesen Spieltyp Ausdruck. Zuvor hatte Sandy White noch nie Spiele-Software geschrieben. Seine Marketinguntersuchungen beschränken sich auf Umfragen bei Freunden, deren Einstellung zu solchen Spielen er in Erfahrung zu bringen versuchte.

Sein bemerkenswertes Programm leitete er zunächst Sinclair Research zu. Das Unternehmen vermochte „Ant Attack" nicht zu beurteilen, da – wie man sagte – im Hause kein Videorecorder zum Abspielen des Videobandes zur Verfügung stand.

Der Spieler oder die Spielerin kann sich durch Granatenwerfen gegen Monsterameisen verteidigen. Leider gibt es keine logische Abfolge beim Einsatz dieser Waffe. Ob es nun an einem irrtümlichen Zufallsfaktor liegt oder einfach am nachlässigen Programmieren, sei da-


hingestellt. Bewegt man den Protagonisten im entgegengesetzten Uhrzeigersinn um 90 Grad, muß man zuvor die Taste „M" des Spectrums und das Shift-Symbol drücken, wenn man in die andere Richtung will. Die Gummitastatur des Spectrum ist nicht sonderlich geeignet, um Kontrolle über die Spielfigur zu erlangen.

Man hat den Eindruck, als sei „Ant Attack" vor der Konstruktion des Sinclair Interface 2, an das man zwei Atari-kompatible Joysticks anschließen kann, entwickelt worden. Eine entsprechende Programmänderung, die eine Joystick-Steuerung vorsieht, sollte indes leicht möglich sein.

Ergänzend zu der Vorwärtsbewegung, dem Springen und dem Werfen von Granaten (überdies sind vier verschiedene Wurfentfernungen möglich), kann der Spieler zwischen vier Blickwinkeln wählen.

Eben dieser Programmteil, die Grafikgene-

ration, unterscheidet „Ant Attack" von anderen Programmen, die mit weniger als 48 K auskommen. Die Umsetzung erfolgt nahezu sofort und übertrifft damit das normale Tempo von 3-D-Grafik-Generatoren für den Spectrum bei weitem. Die Möglichkeit, den Blickwinkel verändern zu können, ist Grundvoraussetzung für das Spiel. Ohne sie bliebe ein wichtiger Teil des Spielfeldes verborgen.

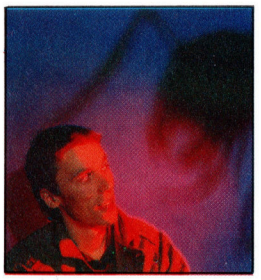
Verständlicherweise wollte der Autor nicht zuviel über seine Programmiertechnik verraten. Er deutete aber an, daß das Spielfeld nicht, wie man erwarten sollte, im Format 128 × 128 × 6 aufgebaut ist. Der Beweis wird vollzogen, wenn man die Spielfigur in die Wüste führt, statt die Stadt zu betreten. Nach einem kurzen Marsch gelangt er oder sie in eine andere Stadt, dann wieder in eine andere, und so weiter. Und letztendlich auch zum Ziel des Spieles selbst. Die Handlung findet in der Stadt Antescher statt (von den Spielautoren zu Ehren des holländischen Künstlers und Designers M. C. Escher so benannt, der geniale optisch-architektonische Täuschungen zeichnete). Vor den Toren der Stadt sind Hilferufe zu vernehmen. Man springt über eine niedrige Mauer und sucht nach dem verfolgten Opfer, springt über Hindernisse oder versucht auszuweichen. Die Stadt scheint isometrisch projiziert und ist nicht einmal ansatzweise nach den Regeln der Perspektive angelegt.

Da stets nur ein kleiner Ausschnitt der Stadt zu sehen ist, folgt der entsprechende Bildschirmausschnitt den Bewegungen des Spielers. Dieses „Scrollen", also „Rollen", ist ebenso hervorragend wie die witzige Animation der Gestalt.

Schon bald zeigt sich, daß die Stadt von riesigen Ameisen bevölkert wird, deren Biß zwar nicht sofort tödlich ist, aber nach entsprechender Anzahl eben doch das Ende des Spiels zur Folge hat. Bemerkt eine Ameise die Anwesenheit des Spielers, folgt sie ihm. Mit genügend Erfahrung kann man sie abhängen oder aber mittels Granate stoppen. Wirft man diese indes gegen die vor einem befindliche Wand, hat das üble Folgen: den eigenen „Tod".

Beim ersten Spieldurchgang befindet sich die zu rettende Gestalt in voller Größe gegenüber dem Eingangstor. Im folgenden Ablauf wird es immer schwieriger, sie zu finden, und noch schwerer, zu ihr zu gelangen, da sie ständig ihren Aufenthaltsort verändert. Da der Retter jeweils nur auf einer Ebene zu bewegen ist, treten verhängnisvolle Schwierigkeiten auf, etwa dann, wenn sich das Opfer eine Treppe höher befindet. Einzige Lösungsmöglichkeit: Den Angriff der Ameisen abzuwarten, eine zu paralysieren und dann auf ihren Rücken zu springen, um sie als Treppe zu mißbrauchen.

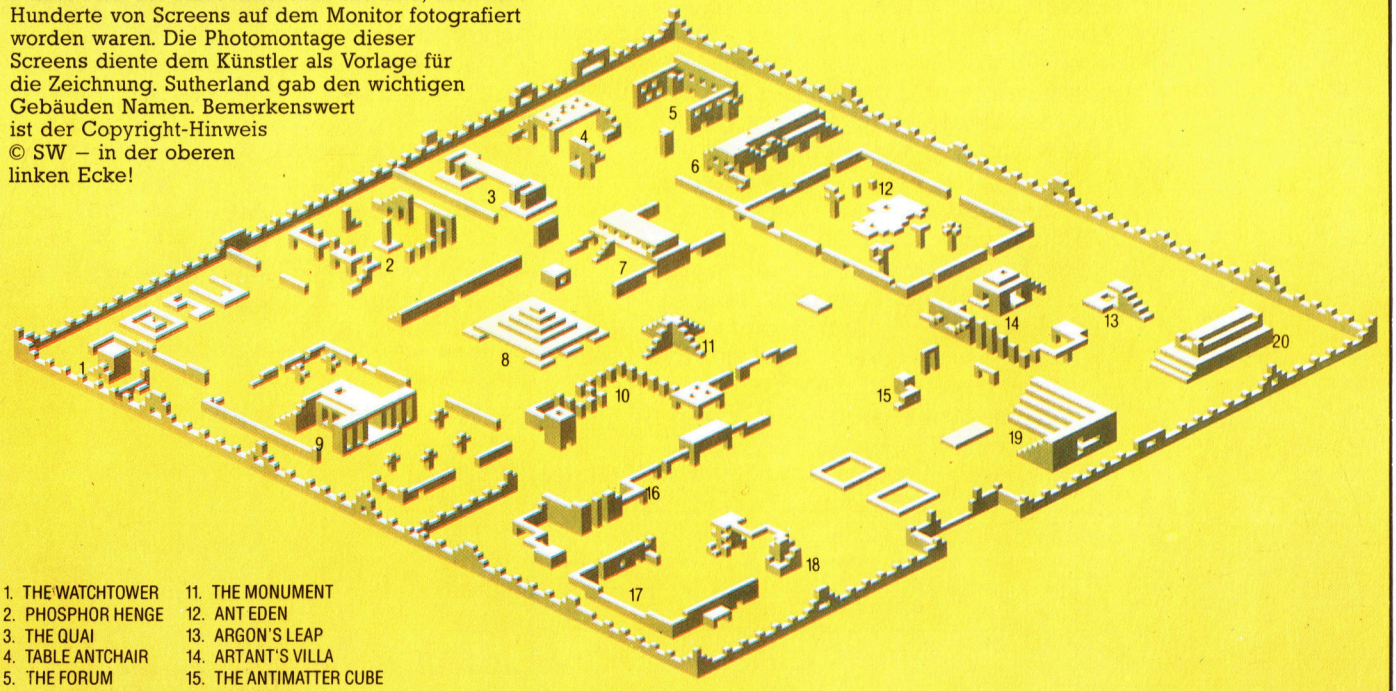
Schließlich kann der Retter einfach davonlaufen – die Ameisen greifen in diesem Fall nicht an. Sobald Held und Opfer außerhalb der Stadt sind, ist der Durchgang beendet.



**„Ant Attack" war der erste Versuch des Autors Sandy White, kommerzielle Software zu schreiben. Sandy, bei Veröffentlichung des Programms 23 Jahre alt, hatte sein Studium der Bildhauerei am Edinburgh College of Art abgeschlossen und wollte einfach ein Programm für Heimcomputer schreiben. Seine Freundin Angela Sutherland half ihm bei der Gestaltung der Stadt Antescher.**

## Rätsel im Sand

Dieser Plan der Stadt Antescher entstand, nachdem Hunderte von Screens auf dem Monitor fotografiert worden waren. Die Photomontage dieser Screens diente dem Künstler als Vorlage für die Zeichnung. Sutherland gab den wichtigen Gebäuden Namen. Bemerkenswert ist der Copyright-Hinweis © SW – in der oberen linken Ecke!




1. THE WATCHTOWER
2. PHOSPHOR HENGE
3. THE QUAI
4. TABLE ANTCHAIR
5. THE FORUM
6. THE ANTICHAMBER
7. SKAZ YANDOR
8. THE PYRAMID
9. THE ANCIENT
10. OXYMINE
11. THE MONUMENT
12. ANT EDEN
13. ARGON'S LEAP
14. ARTANT'S VILLA
15. THE ANTIMATTER CUBE
16. DROXTRAP
17. ADRIANT'S WALL
18. BONZAI WALK
19. THE SQUARENA
20. THE CRYPT

# Kollisionen

**In diesem Teil des LOGO-Kurses werden Sie die Sprite-Grafiken des Atari-LOGOs sowie die vielseitigen Möglichkeiten in bezug auf Farbauswahl und Kollisionsabfragen kennenlernen.**

Das Atari-LOGO arbeitet mit vier Sprites, die mit den Werten 0 bis 3 anzusprechen sind. Da diese Sprites im Handbuch stets als Turtles beschrieben sind, werden auch wir diesen Begriff verwenden.

Die Anweisung TELL 1 bestimmt, daß die Turtle mit der Nummer 1 die folgenden Befehle auszuführen hat. Dazu ein Beispiel:

```
TELL 1
FD 40
RT 90
TELL 2
BK 40
RT 90
TELL 3
RT 90
```

Daneben haben Sie auch die Möglichkeit, mehrere Turtles gleichzeitig anzusprechen:

```
TELL [1 2 3]
FD 50
```

So lange die Form der Figuren nicht verändert wird, stellen sich alle vier Turtles in der bekannten Schildkrötenform dar. Man kann jedoch mit Hilfe des Editors bis zu fünfzehn verschiedene Figuren entwerfen und diese den Turtles zuweisen. Die Art der Bewegung dieser selbstdefinierten Figuren unterscheidet sich jedoch von den originären Turtles.

Um etwa die Figur 1 neu zu definieren, geben Sie EDSH 1 ein. Auf dem Schirm erscheint ein Spriteraster, auf dem Sie die Cursorsteuerungstasten wie gewohnt benutzen können. Mit der Leertaste füllen bzw. löschen Sie die einzelnen Felder. Ist der Entwurf der neuen Figur fertiggestellt, drückt man die ESC-Taste, um diese zu definieren. Mit dem Befehl SETSH 1 wird die zuvor entworfene Form auf die Turtles 1, 2 und 3 übertragen.

Bei Verwendung der Anweisung ASK wird nur eine bestimmte Turtle angesprochen. Geben Sie

```
ASK 1 [FD 20]
```

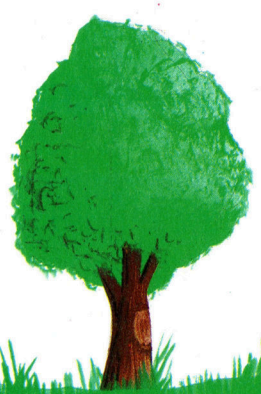
ein. Im Gegensatz zum Befehl FD 20, bei dem sich Turtle 1, 2 und 3 in Bewegung setzen, läuft bei ASK 1 nur die Turtle mit der Nummer 1 vorwärts.

Wie Sie bereits wissen, läßt sich neben Richtung und Position auch die Geschwindigkeit der Turtles verändern. SETSP 30 bewirkt, daß sich die angesprochene Turtle mit einer Geschwindigkeit von 30 Einheiten bewegt und zwar solange, bis ein neuer SETSP-Befehl eingegeben wird. Um die Bewegung der Turtles zu stoppen, geben Sie SETSP 0 ein.

Beim Atari-LOGO stehen 128 verschiedene Farbtöne zur Auswahl, aus denen Sie die Farben für den Hintergrund, den Stift und die Turtles bestimmen können. SETBG 92 zum Beispiel stellt einen grünen Hintergrund dar, und SETPC 0 23 veranlaßt, daß die Farbe des Stiftes 0 (Sie können zwischen drei Stiften wählen) auf Orange (23) wechselt. Die Anweisung SETC 7 bewirkt, daß die entsprechende Turtle in weißer Farbe erscheint.

Das Atari-LOGO enthält einige spezielle Effekte, die besonders bei Bewegungsspielen sehr nützlich sind. Die Tabelle auf Seite 145 des Handbuchs zeigt eine Aufstellung über 21 „Kollisionen und besondere Vorfälle" zwischen Turtles oder zwischen Turtles und Linien, die LOGO überprüfen und feststellen kann. Dabei bestimmt ein sogenannter „Demon" (der eine spezielle Prozedur beinhaltet), was im Falle eines Zusammenstoßes zwischen zwei Objekten passieren soll. Dazu wieder ein Beispiel: Der Zustand der Kollision ist gegeben, wenn Turtle 0 die von Stift 0 gezogene Linie kreuzt. Mit dem Befehl WHEN wird der „Demon", also die aus dem Zusammenstoß resultierende Handlung, definiert:

```
CS
TELL 0
PD
FD 50
PU
RT 90
FD 100
RT 90
FD 20
RT 90
```



Und so könnte der WHEN-Demon aussehen:

```
WHEN 0 [BK 50]
```

Sobald die Turtle nun die Linie erreicht hat, wird der WHEN-Demon aufgerufen, der sie nach der Kollision um 50 Einheiten zurückwirft. Experimentieren Sie mit diesem Beispiel und geben Sie eine andere Geschwindigkeit ein, zum Beispiel SETSP 30.

Die WHEN-Demon-Prozedur bleibt aktiviert, bis die Anweisung WHEN [] eingegeben wird. Sämtliche Demons werden durch CS, durch eine Fehlermeldung oder durch Aufruf des Editors gelöscht.

Hier noch zwei Befehle, mit denen sich Kollisionsbedingungen abfragen lassen: OVER <Turtlenumber> <Pennumber> gibt den Wert der Kollision zwischen der Turtle und der Linie, die mit dem definierten Stift gezogen wurde, aus. TOUCHING <Turtlenumber 1> <Turtlenumber 2> gibt die Demon-Nummer der Kollision zwischen den beiden Turtles aus.

Die folgenden Prozeduren „zwingen" die Turtle in einen Käfig. Sobald sie die eingrenzenden Linien berührt (WHEN OVER 0 0), wird die Demon-Prozedur aufgerufen, die die Turtle um 10 Einheiten zurücksetzt und sie anschließend zu einer RANDOM-Drehung veranlaßt:

```
TO TRAP
  DRAW. TRAP
  HOME
  WHEN OVER 0 0 [TURN]
  SETSP 50
END

TO DRAW. TRAP
  CS
  PU
  SETPOS [—50 —50]
  PD
  SQUARE
  PU
END

TO SQUARE
  REPEAT 4 [FD 100 RT 90]
END

TO TURN
  BK 10
  RT RANDOM 45
END
```

Ähnliche Demon-Prozeduren lassen sich auch in Verbindung mit dem Joystick einsetzen. So zum Beispiel bei Kollisions-Code 3, wenn der Auslöseknopf betätigt wurde, oder Code 15, sobald der Joystick in eine andere Richtung gedrückt wird. Der Befehl JOY 1 gibt einen Wert zwischen −1 und 7 aus, je nachdem in welcher Richtung der Joystick (Port 2) steht:

```
TO JOYH
  IF (JOY 1) <0 [STOP]
  ASK 0 [SETH 45 * JOY 1]
END
```

Setzen Sie nun die Geschwindigkeit auf SETSP 50 und geben Sie den WHEN-Demon ein:

```
WHEN 15 [JOYH]
```

Anhand dieser Prozedur läßt sich die Richtung der Turtle 0 mit dem Joystick steuern.

Das Atari-LOGO bietet ebenfalls die Möglichkeit, mehrere WHEN-Prozeduren zu definieren. Jedoch sind diese nie gleichzeitig aktiv, so daß bei einem eventuellen Zusammenstoß nicht alle Faktoren abgefragt und entsprechend behandelt werden können.

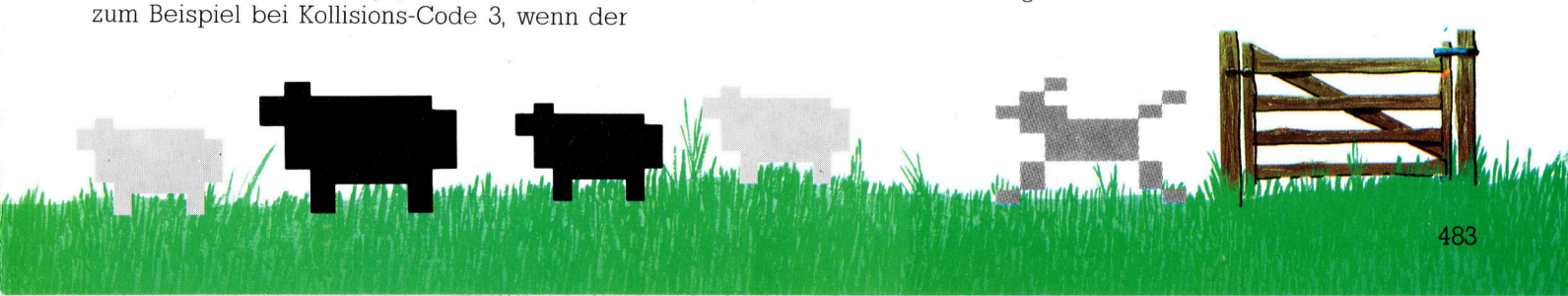
Ein Weg, dieses Problem zu umgehen, ist, die Geschwindigkeit der Turtles mit Hilfe einer Unterprozedur auf Null zu setzen und danach die folgende Prozedur aufzurufen. Diese Überwachungstechnik wird im nächsten Programm angewendet:

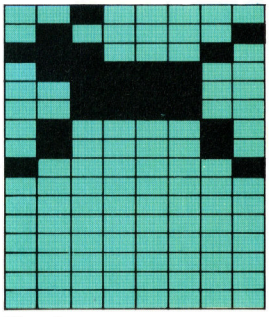
```
TO TRAP
  DRAW. TRAP
  HOME
  WHEN OVER 0 0 [SETSP 0]
  SETSP 50
  WATCH
END

TO WATCH
  IF :SPEED=0 [CHECK]
  WATCH
END
```

Die SPEED-Prozedur gibt die gegenwärtige Geschwindigkeit der Turtle an, und CHECK überprüft, unter welche Kategorie der Zusammenstoß fällt und ob die Geschwindigkeit geändert werden muß.

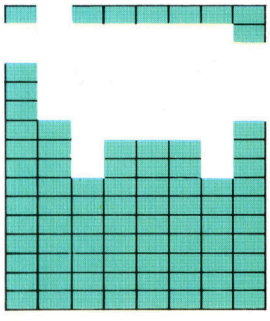
```
TO CHECK
  IF COND OVER 0 0 THEN [TURN]
  SETSP 50
END
```

Der Befehl COND in Verbindung mit einer Zahl ermittelt, ob eine Kollision erfolgt ist.





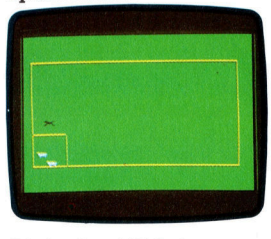
Hund



Schaf

Spielbeginn

Spielverlauf



Ab in den Käfig!

## Schafe hüten

In diesem Programm werden alle zuvor beschriebenen Techniken eingesetzt. Der Spieler steuert den Hund mit Hilfe des Joysticks. Sobald eines der Schafe den Zaun erreicht, wird es zurückgesetzt und anschließend gedreht. Fängt der Hund eines der Schafe, so dreht sich dieses um 90 Grad. Wird der Auslöseknopf des Joysticks gedrückt, erscheint ein Käfig in der linken unteren Bildschirmecke. Ziel ist es, die Schafe in den Käfig zu treiben.

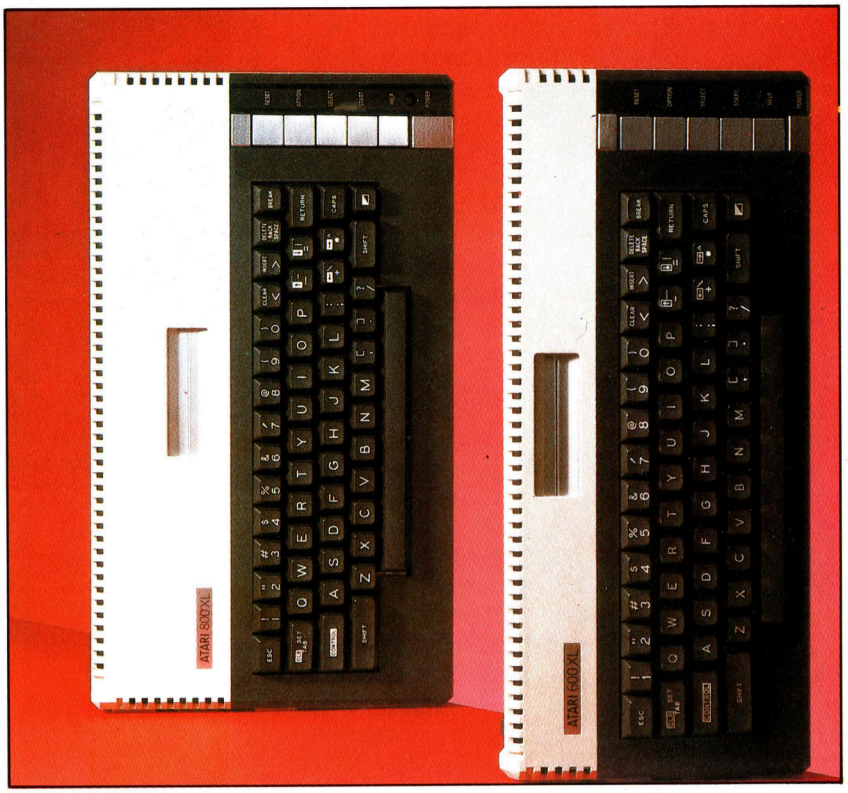
```
TO CHASE
   SET.VAR
   ASK :TURTLE [SET.SCREEN]
   SET.DEMONS
START
   WATCH
END

TO SET.VAR
   MAKE "FENCE 0
   MAKE "TURTLE 0
   MAKE "SHEEP1 3
   MAKE "SHEEP2 2
   MAKE "DOG 1
   MAKE "GREEN 92
   MAKE "ORANGE 23
   MAKE "BLACK 0
   MAKE "WHITE 7
END

TO SET. SCREEN
   CS
   FS
   SETBG :GREEN
   HT
   PU
   SETPOS [-150 -80]
   PD
   SETPC 0 :BROWN
   RECT 160 300
   PU
END

TO RECT :SIDE1 :SIDE2
   REPEAT 2 [FD :SIDE1 RT 90 FD :SIDE2 RT 90]
END

TO SET.DEMONS
   WHEN OVER :SHEEP1 :FENCE [SETSP 0]
   WHEN OVER :SHEEP2 :FENCE [SETSP 0]
   WHEN TOUCHING :SHEEP1 :SHEEP2 [SETSP 0]
   WHEN TOUCHING :DOG :SHEEP1 [SETSP 0]
   WHEN TOUCHING :DOG :SHEEP2 [SETSP 0]
   WHEN 3 [SETSP 0]
   WHEN 15 [JOYH]
END

TO JOYH
   IF ( JOY 1 ) < 0 [STOP]
   ASK :DOG [SETH 45 * JOY 1]
END

TO START
   SET :SHEEP1 1 [-150 20] 45 :WHITE
   SET :SHEEP2 1 [150 20] 315 :WHITE
   SET :DOG 2 [0 0] 0 :BLACK
   SET.SPEEDS
END

TO SET :NO :SHAPE :POS :HEAD :COLOR
   TELL :NO
   PU
   SETSH :SHAPE
   SETC :COLOR
   ST
   SETPOS :POS
   SETH :HEAD
END

TO SET.SPEEDS
   ASK :SHEEP1 [SETSP 10]
   ASK :SHEEP2 [SETSP 10]
   ASK :DOG [SETSP 60]
END

TO WATCH
   IF SPEED = 0 [CHECK]
   WATCH
END

TO CHECK
   IF COND OVER :SHEEP1 :FENCE [ASK :SHEEP1
   [BK 20 RT 90]]
   IF COND OVER :SHEEP2 :FENCE [ASK :SHEEP2
   [BK 20 RT 90]]
   IF COND TOUCHING :SHEEP1 :SHEEP2 [BUMP]
   IF COND TOUCHING :DOG :SHEEP1 [ASK
   :SHEEP1 [RT 90]]
   IF COND TOUCHING :DOG :SHEEP2 [ASK
   :SHEEP2 [RT 90]]
   IF COND 3 [ASK :TURTLE [DRAW.CAGE]]
   SET.SPEEDS
END

TO BUMP
   ASK :SHEEP1 [SETH RANDOM 360]
   ASK :SHEEP2 [SETH RANDOM 360]
END

TO DRAW.CAGE
   PU
   SETPOS [-150 -30]
   PX
   SETH 90
   REPEAT 2 [FD 50 RT 90]
   PU
END
```

### Übungen

1. Ändern Sie das Programm so, daß die Steuerung des Hundes über die Tastatur eingegeben werden kann.
2. Schreiben Sie ein Programm, in dem Sie ein Raumschiff steuern müssen. Ihre Flugbahn wird von Meteoriten gekreuzt, denen Sie natürlich ausweichen müssen. Definieren Sie ein Sprite für das Schiff und die anderen für die Meteoriten. Setzen Sie die WHEN-Demon-Prozeduren ein, um eventuelle Kollisionen zu überprüfen.

# Ataris XL-Serie

**Nach den Modellen 400 und 800 hat Atari die Geräte überarbeitet und die XL-Serie herausgebracht – zu günstigeren Preisen und mit einigen sehr interessanten Verfeinerungen.**




Die Tastaturen der beiden neuen Maschinen sind identisch. Sie verfügen jeweils über 62 Tasten, wobei sich auf der rechten Seite fünf Funktionstasten befinden: HELP, START, SELECT, OPTION und RESET. Neben den 29 festdefinierten Grafikzeichen bieten die Ataris den vollen ASCII-Zeichensatz. Die Cursorsteuerung erfolgt – wie bei den älteren Geräten – über das gleichzeitige Drücken der Control- sowie der jeweiligen Pfeiltaste.

Der Atari 400 verfügte über einen Arbeitsspeicher von 16 KByte, der Atari 800 über 48 KByte. Bei beiden Computern befinden sich die Erweiterungssteckleisten auf der Systemplatine, die nur nach Entfernung des Gehäusedeckels zugänglich ist. Bei den XL-Modellen sind die Erweiterungsleisten von außen bequem zu erreichen. Der einzige wesentliche Unterschied zwischen den XL-Maschinen ist die Größe ihres Arbeitsspeichers. Der 600XL verfügt über 16 KByte, kann aber mit einem Erweiterungspaket auf 64 KByte aufgerüstet werden, während der 800XL bereits mit 64 KByte geliefert wird.

Beide XL-Maschinen haben ein eingebautes Interface, den „Expander". Über diese busartige Steckleiste kann eine ganze Reihe von Peripheriegeräten und Erweiterungsmodulen angeschlossen werden. Unmittelbar hinter der Tastatur befindet sich ein Steckplatz für ROM-Cartridges, über den Spiele und andere Software problemlos eingesetzt werden können. Der Atari 800 hatte zwei Cartridgeslots, wobei es für den zweiten Schacht jedoch so gut wie keine Software gab. Dieser zweite Slot wurde bei den XL-Modellen ebenso wie zwei der ursprünglichen vier Joystickports weggelassen.

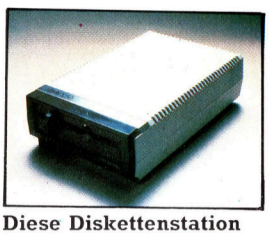
## Eingebautes Interface

Alle Ataricomputer lassen sich direkt an ein Fernsehgerät anschließen und (mit Ausnahme des 400er) auch an einen Monitor. Doch selbst bei Fernsehgeräten ist die Bildschirmdarstellung gut: Die Zeichen sind leicht lesbar, und der Kontrast zwischen Text und Hintergrund ist scharf. Es kann eine große Anzahl von Farben eingesetzt werden. Die normale Textdarstellung erscheint in weißen Buchstaben auf blauem Hintergrund. Die Computer von Atari zeigen maximal 24 Zeilen zu je 40 Zeichen an.

## Grafikmöglichkeiten

Atari war einer der ersten Hersteller, der dem Anwender Spritegrafik zur Verfügung stellte. Die Sprite-Funktion heißt „Player-Missile"-Grafik und wird von einem Spezialchip – dem GTIA – gesteuert. „Players" (Spieler) sind Objekte, die aus Pixeln aufgebaut sind. Ist die Form eines „Players" definiert, können die Werte der einzelnen Pixel mit POKE in einem Bereich des Speichers abgelegt werden, der „Shape Table" (Formentabelle) genannt wird. Es lassen sich vier „Player" einsetzen, von denen jeder einer bestimmten Missile-Komponente entspricht. Auf dem Bildschirm läßt sie sich über die Veränderung der Werte im „Shape Table" bewegen. Mit Hilfe der Player-Missile-Grafik lassen sich erstaunliche Bildschirmdarstellungen erzeugen. Der Einsatz dieser Grafikelemente setzt jedoch gute Pro-

**Der 600XL und der 800 XL sind fast identisch. Der 600XL hat jedoch nur 16K Arbeitsspeicher, während der 800XL über 64K verfügt. Beide Maschinen besitzen eine Schreibmaschinentastatur und gute Farbgrafik. Da sie verbesserte Versionen der früheren Atari-Computer sind, steht ihnen bereits eine breite Palette an Software zur Verfügung.**



**Diese Diskettenstation ist eine praktische Erweiterung. Die Speicherkapazität beträgt 127 KByte. Die Standardversion des 600XL hat nicht genug Speicherplatz, um die Diskettenstation nutzen zu können. Durch Erweiterung auf 64 K kann er jedoch auch mit dieser Floppy arbeiten.**

grammierkenntnisse voraus.

Die XL-Maschinen verfügen über 11 Grafikarten und bis zu 256 Farben (d. h. 16 Farben, die bis zu 16 verschiedene Schattierungen annehmen können). Die Anzahl der Farben, die gleichzeitig auf dem Schirm dargestellt werden kann, hängt allerdings von dem Auflösungsgrad der Grafik ab, da die Kapazität des Bildschirmspeichers begrenzt ist: Je höher der Grad der Auflösung, desto weniger Farben lassen sich gleichzeitig darstellen. Die maximale grafische Auflösung des 600XL und des 800XL beträgt 320 × 192 Pixel.

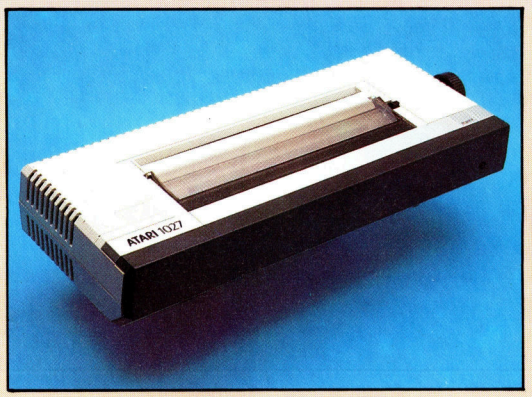
## Großer Tonumfang

Auch die Klangeigenschaften der Ataris werden von einem Spezialchip gesteuert. Es gibt vier unabhängige Stimmen, von denen jede einen Tonumfang von 3 ½ Oktaven hat. Die einzelnen Stimmen lassen sich vom BASIC aus über den SOUND-Befehl steuern, oder über POKE durch die Belegung von bestimmten Speicherstellen. Im Aufbau eines Tones lassen sich Frequenz, Tonhöhe, Verzerrung und Lautstärke genau festlegen. Über den Befehl SOUND kann man nur jeweils eine Stimme ansteuern, so daß bei Akkorden jede Stimme nacheinander und einzeln angegeben werden muß. Die dadurch entstehenden Zeitverzögerungen lassen sich durch den Einsatz von Maschinencode vermeiden oder durch den Befehl POKE.

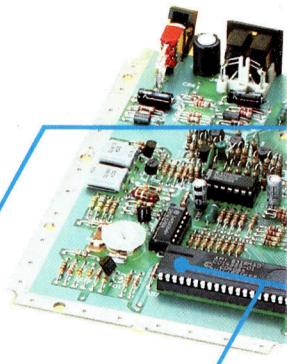
Zusätzlich zu den neuen Computermodellen hat Atari die bereits existierenden Peripheriegeräte umgestaltet und zudem weitere Geräte auf den Markt gebracht. Das vielleicht nützlichste Peripheriegerät ist eine Zusatzbox, die sich in den Expander stecken läßt. Sie hat acht Erweiterungssteckleisten und kann Schnittstellenkarten für mehrere Peripheriegeräte, zwei serielle Anschlüsse (RS232) und einen parallelen Bus aufnehmen.

Der 600XL und der 800XL zeichnen sich durch guten Aufbau, Solidität der Konstruktion und hervorragende Grafikfähigkeiten aus. Für die Geräte gibt es außerdem noch eine große Anzahl Programme auf Cartridges, Cassetten und Disketten. Computerbegeisterten sind diese beiden Maschinen sehr zu empfehlen.

Inzwischen stellte Atari bereits wieder neue Computer-Modelle vor: die XE- und die ST-Serie. Die XE-Geräte wurden, wie die älteren Atari-Computer, mit einem 6502-Prozessor ausgerüstet, so daß auch für diese Geräte die vorhandene umfangreiche Software-Palette zur Verfügung steht. Die ST-Serie zeichnet sich durch den 68000-Prozessor sowie durch Macintosh-ähnliche Grafikfähigkeiten aus. Als externe Speicher lassen sich an den Atari 130ST und 520ST 3,5-Zoll-Floppies bzw. Hard Disks anschließen. Bei der ST-Serie wurde ferner die Maus-Technologie angewendet.




**Atari-Drucker 1027**
Der Drucker ist mit einer Typentrommel ausgerüstet. Die Druckqualität steht herkömmlichen elektrischen Schreibmaschinen nicht nach, die Druckgeschwindigkeit ist allerdings sehr niedrig.
Atari bietet noch weitere Drucker an, zum Beispiel den Farbplotter, der mit Minen arbeitet, sowie einen Matrixdrucker. Andere können über eine externe Schnittstelle angeschlossen werden.

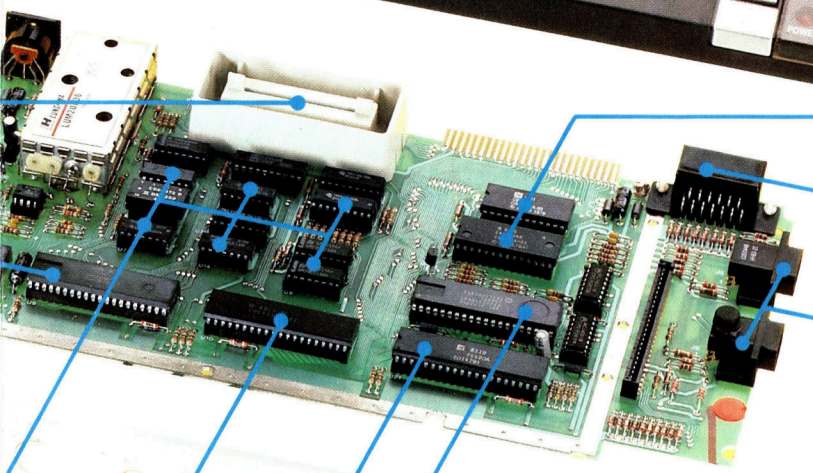



**Cartridge-Slot**
Die XL-Reihe hat nur einen Steckplatz für Cartridges.

**Spezialchips**
Der GTIA-Clip ist für die Grafik, ANTIC dagegen für die Bildschirmsteuerung sowie Ein- und Ausgabefunktionen verantwortlich.

**RAM**
Diese Chips enthalten 16 K RAM.


**Trackball und Joystick**
Der Atari bietet verschiedene Möglichkeiten der Spielsteuerung. So z. B. per Joystick, wobei der von Atari (rechts) zu einer Art Industriestandard wurde. Der Atari-Trackball besteht aus einer Kugel, die sich im Gehäuse drehen läßt.



**ROM**
Diese zwei ROMs des Atari enthalten den BASIC-Interpreter.

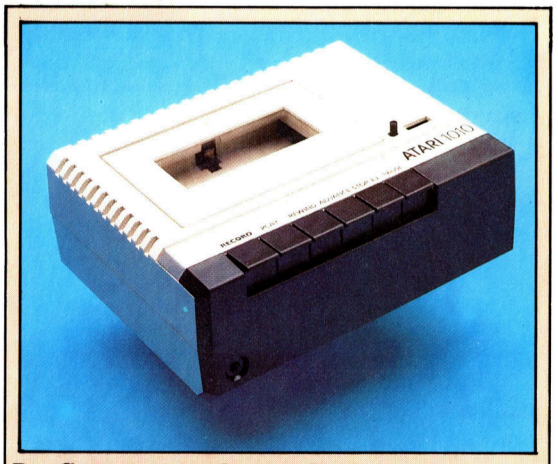
**Peripherieanschluß**
Über diese dreizehnpolige Steckleiste werden Peripheriegeräte, wie z. B. Diskettenstation, Drucker und der Atari-Cassettenrecorder, angeschlossen.

**Anschluß für Joysticks**

**Tonerzeugung**
Ein Spezialchip namens POKEY kontrolliert die Klangerzeugung.

**Chip für Ein- und Ausgabe**
Ein 6520 steuert die Schnittstellen für Ein- und Ausgabe.

**Zentraleinheit**
Der Atari ist mit dem bekannten 6502-Prozessor ausgerüstet.



**Der Cassettenrecorder von Atari**
Als Cassettenspeicher läßt sich nur der Atari-eigene Recorder einsetzen. Er nutzt zwei Spuren zur Aufzeichnung: Eine Spur speichert Programme, während die zweite Spur für Tonaufnahmen eingesetzt werden kann. Bei Lernprogrammen für Sprachen können damit Worte und Satzteile abgerufen werden.

## Atari 600XL / 800XL

**PREIS**
600XL: ca. 250 DM
800XL: ca. 450 DM

**GRÖSSE**
600XL: 380 × 170 × 40 mm
800XL: 380 × 220 × 40 mm

**CPU**
6501, 1,79 MHz

**SPEICHERKAPAZITÄT**
16–64 KByte RAM, 24 KByte ROM

**BILDSCHIRM-DARSTELLUNG**
24 Zeilen mit je 40 Zeichen Text, Grafik bis zu 320 × 192 Pixel mit Sprites und 16 Farben in je 16 Helligkeitsstufen.

**SCHNITTSTELLEN**
Joysticks(2), Peripherieanschluß, Erweiterungssteckleiste, Cartridgeslot

**VERFÜGBARE PROGRAMMIERSPRACHEN**
BASIC, FORTH, LOGO, PILOT, 6502-Assemblersprache

**TASTATUR**
Schreibmaschinentastatur mit insgesamt 62 Tasten, darin enthalten sind Cursorsteuerungstasten und Funktionstasten für Programmsteuerung wie SELECT und START.

**DOKUMENTATION**
Ausführliche Handbücher waren noch nie die Stärke von Atari. Es gibt jedoch eine große Anzahl ausgezeichneter Bücher und Zeitschriften, die unabhängig von Atari entstanden sind, aber auch extra bezahlt werden müssen.

**STÄRKEN**
Atari setzt seine Tradition von ausgezeichneten Computern mit hervorragender Grafik, Klangerzeugung und einer breiten Palette an bestehender Software fort. Die neuen Maschinen verfügen auch über exzellente Erweiterungsmöglichkeiten.

# Klanggebäude

**Die Untersuchung des Befehls ENVELOPE auf dem Acorn B wird mit der „Hüllkurve der Lautstärke" fortgesetzt.**

In der letzten Folge dieser Serie haben wir den Befehl ENVELOPE auf dem Acorn B vorgestellt. Zusammen mit dem Befehl SOUND gibt er dem BASIC-Programmierer eine Vielzahl von Steuerungsmöglichkeiten. In den folgenden Beispielen geht es um die „Hüllkurve der Lautstärke".

In der nachstehenden Programmzeile steuern die Parameter N bis NS3 die Hüllkurve der Tonhöhe (siehe letzte Folge).

ENVELOPE N, T, PS1, PS2, PS3, NS1, NS2, NS3, AR, DR, SR, RR, FAL, FDL

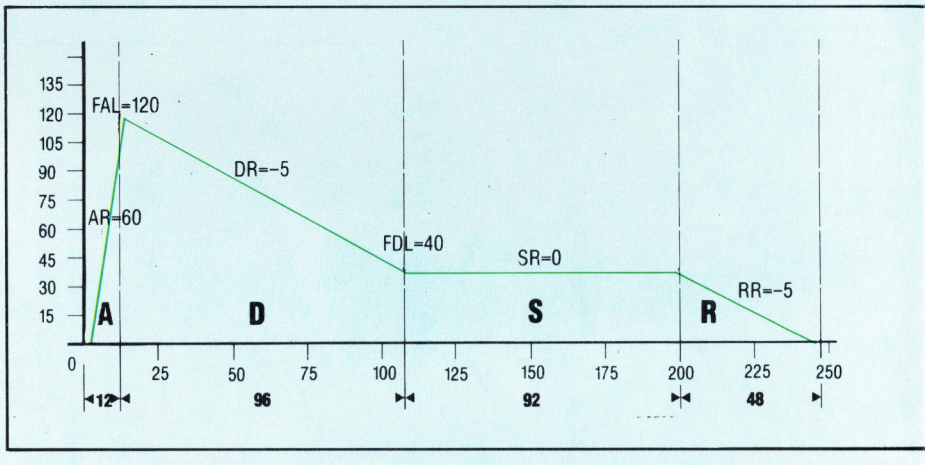
Die verbleibenden Parameter beziehen sich alle auf die Hüllkurve der Lautstärke.

AR & DR (—127 bis 127) + FAL & FDL (0 bis 126)

Mit AR wird die Attack-Rate (Anfangslautstärke) eines Tones bestimmt. Die Software erlaubt zwar auch die Angabe eines negativen Wertes. Es läßt sich aber nur der Bereich von 1 bis 127 verwenden. Der Wert von AR bezieht sich auf die Anzahl der Veränderungen pro Zeitabschnitt und läßt die Lautstärke kontinuierlich ansteigen, bis der Wert von FAL (Final Attack Level = Ebene der höchsten Anfangslautstärke) erreicht ist und die Decay-Phase (Abklingen) beginnt. Das Decay wird mit der Variablen DR auf die gleiche Weise gesteuert, wobei DR normalerweise mit einer negativen Zahl festlegt, wie schnell die Lautstärke bis auf den Wert von FDL — Final Decay Level (endgültige Abklinglautstärke) abfallen soll. Obwohl die Angabe eines Wertes zwischen 0 und 126 möglich ist, läßt die Hardware derzeit nur Zahlen zwischen 0 und 16 zu. Ein FAL-Wert von z. B. 50 wird dabei automatisch heruntergesetzt und auf 6 gerundet.

SR & RR (—127 bis 0)

Die Werte für Sustain (SR) und Release (RR) beziehen sich ebenfalls auf die Veränderung der Lautstärke pro Zeitabschnitt, können jedoch nur negative Zahlen annehmen. Das Sustain hält so lange an, bis die im SOUND-Befehl angegebene Tondauer beendet ist. Ist die Zeit für Attack und Decay länger oder gleich der über SOUND gesetzten Tondauer, dann wird die Sustain-Phase unterdrückt, selbst wenn sie programmiert wurde. Der Releaseabschnitt beginnt, sobald die in SOUND festgelegte Tondauer beendet ist. Dabei fällt die Lautstärke entsprechend der in RR gesetzten Geschwindigkeit bis auf Null ab, wenn nicht inzwischen auf dem gleichen Oszillator ein neuer Ton begonnen wurde. Der Releaseabschnitt wird also abgebrochen, wenn „H" nicht von einem neuen SOUND & -Befehl auf „1" gesetzt wird.




Unser Diagramm zeigt die Hüllkurve für die Nachahmung eines Klaviertones. Die dazugehörigen Variablenwerte sehen folgendermaßen aus:

| | | | |
|---|---|---|---|
| T=6 | AR=60 | SR=0 | FAL=120 |
| | DR=—5 | RR=—5 | FDL=40 |
| SOUND | Tondauer = 40 (2 Sekunden) | | |

Und der entsprechende Befehl:

ENVELOPE 1,6,0,0,0,0,0,0,60,—5,0,—5,120,40

Das folgende Programm verwendet alle Tonbefehle des BBC-BASIC. Es spielt eine Tonfolge mit einer Hüllkurve für Klavier und setzt über den letzten Akkord eine kurze dreieckförmige Hüllkurve für Tonhöhen, die mehrfach wiederholt werden.

```
10 REM**COSMIC**
20 ENVELOPE 1,6,0,0,0,0,0,0,60,—5,0,—5,
   120,40
30 ENVELOPE 2,6,1,—1,1,1,2,1,60,—5,0,
   —5,120,40
40 FOR I=1TO4:READ N
50 SOUND 1,1,N,20:REM**SPIEL A B G G**
60 SOUND &1001,0,0,5:NEXT I
70 SOUND &201,2,77,40:REM**LETZTER**
80 SOUND &202,2,89,40:REM**D-DUR**
90 SOUND &203,2,109,40:REM**AKKORD**
100 DATA 137,145,129,85:REM**ABGG**
```

# Lichtwellen

**Die Grafikfähigkeiten der Ataris haben einen Trend gesetzt, dem auch andere Hersteller folgen.**

Eine der interessantesten Fähigkeiten des Atari ist seine spriteähnliche Grafik, die unter dem Namen „Player Missile-(PM) Grafik" bekannt wurde und vom BASIC aus das Schreiben von schnellen und interessanten Spielen möglich macht. Für die PM-Grafik gibt es in BASIC jedoch keine speziellen Befehle, und alle Abläufe müssen über PEEK und POKE direkt an die entsprechenden Speicherstellen im RAM gesetzt werden.

Die Darstellungsarten 0, 1 und 2 sind für Text reserviert. Beim Einschalten der Maschine wird automatisch Mode 0 aktiviert und der Bildschirm auf 24 Zeilen mit je 40 Zeichen eingestellt, wobei die Zeichen das normale ASCII-Format von acht mal acht Bildpunkten haben. In der Darstellungsart 1 sind die Zeichen doppelt so breit wie bei Mode 0, haben aber die gleiche Höhe. Mode 2 dagegen dehnt die Zeichen sowohl in der Höhe als auch in der Breite um das Doppelte des Mode 0 aus.

Mit Ausnahme von 0 haben alle Darstellungsarten einen unterteilten Bildschirm, wobei am unteren Rand einige Zeilen für Systemmeldungen und Fehleranzeigen reserviert sind. Bei der Darstellungsart 1 und 2 muß eine Kanalnummer angegeben werden, wenn mit PRINT ein Zeichen im Hauptfeld ausgegeben werden soll. So kann Text z. B. über PRINT #6 in den Grafikbereich des Bildschirms gedruckt werden. Modi 3 bis 8 sind für Grafik reserviert. In ihnen können Punkte und Linien mit unterschiedlichen Auflösungsgraden in mehreren Farben auf den Schirm gebracht werden. Die abgebildete Tabelle gibt einen Überblick über die verschiedenen Möglichkeiten in den Grafik- und Textmodi.

Die Wahl der Darstellungsart hängt von dem zur Verfügung stehenden freien Speicherplatz ab. Mode 5 benötigt mit vier Farben fast doppelt soviele Speicherstellen wie Mode 4 mit nur zwei Farben.

```
SETCOLOR a,b,c
```

Fünf Register steuern die Bildschirmfarben, wobei nicht jede Darstellungsart alle Register benötigt. SETCOLOR wählt die Farben aus, die über diese fünf Register eingesetzt werden. Dabei ist a die Nummer des Farbregisters (0–4), b die Farbnummer (0–15) und c eine von acht Helligkeitsstufen, die über eine gerade Zahl zwischen 0 und 14 bestimmt wird.

```
COLOR n
```

Dieser Befehl funktioniert auf zwei Arten und ist abhängig davon, ob ein Text- oder Grafikmode gewählt wurde. In den Darstellungsarten 0, 1 und 2 ist n eine Zahl zwischen 0 und 255.

In den Grafikarten kann n einen Wert zwischen 0 und 3 annehmen und bestimmt damit das Farbregister, das bei dem PLOTten eines

| Mode | Art | Zeilen | Spalten | Farben |
|---|---|---|---|---|
| 0 | Text | 24 | 40 | 2 |
| 1 | Text | 20 | 20 | 5 |
| 2 | Text | 10 | 20 | 5 |
| 3 | Grafik | 20 | 40 | 4 |
| 4 | Grafik | 40 | 80 | 2 |
| 5 | Grafik | 40 | 80 | 4 |
| 6 | Grafik | 80 | 160 | 2 |
| 7 | Grafik | 80 | 160 | 4 |
| 8 | Grafik | 160 | 320 | 1 |

Punktes angesprochen wird.

```
PLOT x, y
```

Der Nullpunkt des Atari-Bildschirms befindet sich in der linken oberen Ecke. PLOT spricht den durch die x-y-Koordinaten bestimmten Punkt an. Der Befehl POSITION funktioniert nach dem gleichen Schema:

```
POSITION x, y
```

setzt einen unsichtbaren Cursor an den Punkt (x, y) des Bildschirms.

```
DRAWTO x, y
```

zeichnet eine gerade Linie von der alten Cursorposition zu dem angegebenen Punkt (x, y). Schließlich setzt

```
X10 18, #6, 0,0,"S:"
```

den Ein- und Ausgabebefehl X10 von Atari ein, der einen auf den Bildschirm gezeichneten Umriß mit Farbe ausfüllt. Der Befehl ist recht kompliziert, erzielt aber gute Ergebnisse. Ist ein geschlossener Umriß einmal auf dem Bildschirm gezeichnet, braucht der Cursor nur an dessen linke untere Ecke gesetzt werden. Die Einfärbung beginnt am oberen Rand des Umrisses und setzt sich über die gesamte Figur fort, bis sie den Cursor am unteren Rand erreicht. Die Farbe wird mit POKE 765, C gesetzt, wobei C wie bei COLOR Werte zwischen 0 und 3 annehmen kann.

**Atari-Grafik ist nicht einfach zu programmieren. Das Fehlen von Standardbefehlen für hohe Auflösung wie z. B. CIRCLE bedeutet, daß der Programmierer recht hart arbeiten muß, um gute Ergebnisse erzielen zu können. Von Vorteil ist die Wahl zwischen mehreren Textarten. Das folgende Programm zeigt den Einsatz von Zeichen in doppelter Größe, die in Verbindung mit dem Befehl POSITION einen Text auf den Schirm bringen:**

```
10 REM GROSSBUCH-
   STABEN
20 GRAPHICS 2+16
30 SETCOLOR 0,3,6
40 FOR X=19 TO 8
   STEP -1
50 POSITION X,1
60 FOR J=1 TO 100:
   NEXT J
70 PRINT#6;" DAS IST "
80 NEXT X
90 FOR X=19 TO 6
   STEP -1
100 POSITION X,3
110 FOR J=1 TO
    100:NEXT J
120 PRINT#6;"IHR "
130 NEXT X
140 FOR X=13 TO 12
    STEP -1
150 POSITION X,9
160 FOR J=1 TO
    100:NEXT J
170 PRINT #6; "COM-
    PUTER "
180 NEXT X
190 SETCOLOR 0,5,5
200 FOR Y=9 TO 5
    STEP -1
210 POSITION 7,Y
220 PRINT #6; "KURS"
230 NEXT Y
240 GOTO 240
```

**Addiert man beim gewählten Mode 16 hinzu (Zeile 20), erscheint kein Textfenster.**

# Zukunfts-musik

**Was werden die nächsten fünf Jahre der Computerentwicklung bringen – decken sich unsere Prognosen mit Ihren Erwartungen?**

Wie sieht der Heimcomputer der neunziger Jahre aus, und was wird er können? Auf diese Fragen soll hier anhand einer Betrachtung des Konzepts und wesentlicher Komponenten des Rechners von morgen eine Antwort versucht werden. Viele der Prognosen stützen sich auf Technologien, die gerade auf den Markt kommen, während sich andere auf denkbare zukünftige Entwicklungen beziehen.

Mit das Wichtigste bei der vorgestellten Hypothese ist der modulare Aufbau. Der Benutzer wird künftig ein umfassendes Angebot zur Erweiterung der Grundeinheit vorfinden. Er wird seinen Rechner weitestgehend selbst konzipieren können, indem er sich ein bestimmtes Grafik- oder Soundsystem aussucht.


### 1. Tastatur-Anzeige

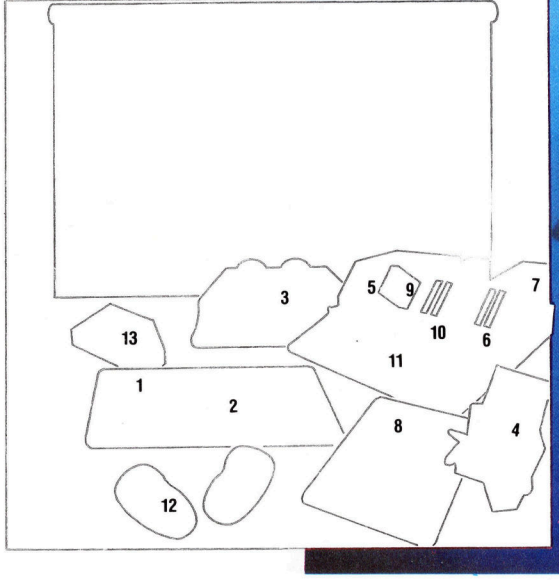
Die 32-Bit-Microprozessoren können gleichzeitig mehrere Anzeigefelder bedienen. Der große Bildschirm kann z. B. die Aussicht vom Pilotensitz eines Raumschiffs wiedergeben, während auf einer zusätzlichen Anzeigetafel über dem Tastenfeld die Informationen der Cockpit-Instrumente erscheinen.

### 2. Tastenfeld

Trotz der bekannten Schwächen der Schreibmaschinentastatur ist es unwahrscheinlich, daß ernsthaft versucht wird, sie durch eine Alternative zu ersetzen. Gut gefederte Schreibmaschinentasten finden noch immer den größten Anklang, obwohl sie durch „Halleffekt"-Tasten verdrängt werden könnten, die kontaktlos durch Verschiebung von Magneten über Halbleiterplättchen schalten. Denkbar wäre auch eine optoelektronische Tastatur, bei der die betätigten Tasten mit Hilfe von Laserstrahlen abgelesen werden.

### 3. Monitor

Projektions-Fernsehgeräte gibt es zwar seit Anfang der 80er Jahre, sie sind aber wegen der begrenzten Leistungsfähigkeit der Kathodenstrahlröhre nur eingeschränkt verwendbar. Fortschritte in der Röhren-Technologie werden eine raumfüllende Projektion auf flachen Bildschirmen ermöglichen.




### 4. Zusatz-Prozessoren

Der Rechner der 90er Jahre wird neben der 32-Bit-Zentraleinheit vermutlich weitere Prozessoren in Form von Steckmodulen aufnehmen können. Teilaufgaben – z. B. die Versorgung bestimmter Peripheriegeräte oder die Darstellung von Dateien – würden dann von der Zentraleinheit an den geeigneten Zusatzprozessor delegiert.

### 5. Arbeitsspeicher

Der 32-Bit-Prozessor kann ein RAM mit über vier Milliarden Byte adressieren – kein Vergleich mit den 65536 Byte, die die Grenze für die meisten zur Zeit üblichen Heimcomputer mit dem 8-Bit-Prozessor bedeuten.

### 6. Kommunikationstechnik

In den 90er Jahren werden Reflektorantennen für den Satellitenempfang wohl ebenso üblich sein wie digitale Telefonkanäle. Dabei wird man für die Steuerung von Sendung und Empfang spezielle Kommunikations-Steuereinheiten benötigen, die auch die Funktion der heutigen MODEMs übernehmen.

### 7. Netzteil

Der erhöhte Stromverbrauch und die Vielzahl der Anschlußgeräte erfordern stärkere Netzteile als bisher üblich. Die Spannungsregelung mit aufladbaren Bufferbatterien wird dazugehören, so daß Netzspannungsschwankungen oder Stromausfall nicht zur Zerstörung von Register- und Speicherinhalten führen können.

### 8. Kabelloser Bildschirm

Ein Flachbildschirm – vermutlich in Form einer schnell reagierenden Flüssigkristall-Matrix und vielleicht über Infrarot oder sogar UHF mit der Zentraleinheit gekoppelt – könnte für die Wiedergabe von Texten und Grafiken verwendet werden. Eine wegweisende Alternative zum Monitor.

### 9. CD-ROM

Das Compact-Disc-ROM, bei dem mit einem Laser optisch gespeicherte Information ausgelesen wird, dürfte wegen seiner weit höheren Kapazität die herkömmlichen ROM-Steckmodule verdrängen.

### 10. Diskettenlaufwerke

Bis zum Ende dieses Jahrzehnts können hochentwickelte Disketten hinsichtlich Geschwindigkeit und Aufzeichnungsdichte wahrscheinlich mit Winchesterplatten konkurrieren. Dabei könnte ihr Durchmesser unter dem derzeitigen Minimum von 3 Zoll liegen.

### 11. Schalterkonsole

Bei den ersten Rechnern mußte man – bevor höhere Sprachen und Tastaturen üblich wurden – die Befehle über ein Bedienungspult binär eingeben. Dieses bestand aus einer Reihe von Schaltern und Lämpchen, durch die man Zugang zu jedem Einzelbit des Adreß-, Daten- und Steuer-Busses hatte.

### 12. Infrarot-Mäuse

Beim PC-Junior von IBM erfolgt die Datenübertragung von der Tastatur zum Rechner bereits kabellos durch Infrarotstrahlen. Diese Technologie der drahtlosen Steuerung und Dateneingabe mit Hilfe der Infrarot-„Mäuse“ befreit den Anwender von dem leidigen Kabel-Wirrwarr.

### 13. 32-Bit-Microprozessor

Die ersten Heimcomputer mit 32-Bit-Prozessor kamen 1983 auf den Markt. Sie konnten den Prozessor aber nicht voll nutzen, weil sie mit einer Datenbus-Breite von 16 oder nur 8 Bit arbeiteten, um die Kompatibilität mit den übrigen Speicher- und Peripherie-Bausteinen zu gewährleisten. Mit der Einführung von Bausteinen wie dem Motorola 68032 mit 32-Bit-Datentransfer wird diese Leistungsfähigkeit den Standard prägen.

# Montage der Module

**Nachdem die elementaren Teile des Programms fertiggestellt sind, kann jetzt nach Möglichkeiten gesucht werden, um das „elektronische Adreßbuch" anwenderfreundlicher zu gestalten.**

Obwohl noch viele Details des Adreßbuch-Programms ausgearbeitet werden müssen, sollte die allgemeine Struktur jetzt klar werden. An diesem Punkt der Entwicklung eines Programms, ganz gleich welcher Größe, ist es ratsam, ein Block-Diagramm zu zeichnen und sich Gedanken über die einzelnen Aktivitäten innerhalb des Programms zu machen.

Die Tabelle zeigt die bisher erstellten Hauptblöcke des Programms. Um einen einheitlichen Standard festzulegen, enthalten Namen für Unterroutinen sechs Buchstaben, String-Felder (inkl. $) sieben, einfache numerische Variablen (inkl. $) fünf Buchstaben. Lokalen Variablen (beispielsweise in Schleifen) sind einzelne Buchstaben zugeordnet.

**HAUPTPROGRAMM-BLÖCKE**

| | | |
|---|---|---|
| | CREARR | (definiert Datenfelder und initialisiert Variablen) |
| INITIL | LSINDT | (liest Datei von Cassette oder Diskette) |
| | SETFLG | (setzt Flag und modifiziert Variablen) |
| BGRUES | | (druckt Begrüßung aus) |
| AUWAHL | CHMENU | (druckt Options-Menü) |
| | INWAHL | (weist Option WAHL zu) |
| | FNDVER | (lokalisiert und druckt ein Verzeichnis) |
| | FNDNMS | (lokalisiert Namen aus Teileingabe) |
| | FNDSTD | (lokalisiert Verzeichnis nach Stadt) |
| | FNDINT | (lokalisiert Namen nach Initialen) |
| AUSFUH | LSTVER | (listet alle Verzeichnisse) |
| | ADDVER | (fügt neues Verzeichnis hinzu) |
| | MODVER | (modifiziert bestehendes Verzeichnis) |
| | LOEVER | (löscht ein Verzeichnis) |
| | ENPROG | (sichert Datei und beendet Programm) |

Jeder der Hauptprogramm-Blöcke in der zweiten Spalte muß noch in Untereinheiten aufgeteilt werden. Diese Subroutinen müssen dann noch weiter verbessert werden, bevor das Programm in BASIC geschrieben werden kann.

Vorausgesetzt, daß die meisten Programm-Module bereits ausgearbeitet, in BASIC geschrieben und getestet wurden, wie können sie dann zu einem kompletten Programm zusammengefügt werden? Die beste Lösung ist, jedes Modul unter dem ihm gegebenen Namen auf Cassette oder Diskette abzuspeichern. Wenn man ein Programm von Cassette oder Diskette einlädt, verwendet man normalerweise den LOAD-Befehl, gefolgt vom Dateinamen (z. B. LOAD „ADDVER"). Durch diesen Befehl wird jedoch der gesamte Speicher gelöscht. Hätte man also ADDVER im Speicher und würde MODVER einladen, würde ADDVER gelöscht.

Doch für dieses Problem gibt es eine einfache Lösung, den MERGE-Befehl. Dieser Befehl lädt ein Programm von Cassette oder Diskette, ohne den Speicher zu löschen. Ein wichtiger Punkt muß aber trotzdem noch beachtet werden. Sollten Programmzeilen der Routine MODVER mit Zeilen von ADDVER übereinstimmen, so werden die alten Zeilen durch die neuen ersetzt. Das Resultat: Chaos. Mit BASIC-Versionen, die den Befehl RENUM kennen, kann dies leicht umgangen werden, indem man die einzelnen Routinen vor dem Abspeichern neu durchnumeriert.

## Auf Zeilennummern achten

Leider kennen jedoch die meisten BASIC-Versionen von Heimcomputern den Befehl RENUM nicht. Daher ist es ratsam, die Verwendung der Zeilennummern von Anfang an so zu planen, daß problemlos Erweiterungen integriert werden können. Wenn Ihre BASIC-Version nicht über den MERGE-Befehl verfügt, müssen Sie die einzelnen Routinen neu eintippen und gemeinsam abspeichern.

Die Programm-Module in unserer Tabelle sind testweise zusammengefaßt worden (siehe Listing). Wir wollen Ihnen damit die „Fallgruben" aufzeigen, die bei dieser Methode auftreten. Zugegebenerweise verleitet BASIC dazu, immer erst einmal alles kurz auszuprobieren, anstatt richtig zu planen. Wenn Sie noch alle Routinen aus den vorangegangenen Teilen gespeichert haben und Ihre BASIC-Version über

den RENUM-Befehl verfügt, können Sie versuchen, die Routinen neu durchzunumerieren und mit MERGE zusammenzufügen.

Der erste Block des Hauptprogramms ist INITIL, der zur Initialisierung von Variablen, Dimensionierung von Bereichen, Einlesen von Dateien, Zuordnen der Daten in die Felder und Setzen von Flags zuständig ist.

Die *INITIL*-Unterroutine ist in die Routinen *CREARR* (zum Definieren von Datenfeldern), *LSINDT* (zum Einlesen der Dateien und Zuordnen der Daten) und *SETFLG* (zum Setzen von Flags) unterteilt.

Wenn all dies ausgeführt wurde, fährt das Programm mit der Routine *BEGRUES*, einer Unterroutine zum Drucken einer Begrüßungsmeldung, fort.

Das Programm geht dann weiter zu *AUWAHL*, sobald der Anwender die Leertaste betätigt. Diese Routine umfaßt zwei Teile: Der erste stellt ein Menü von verfügbaren Optionen des Programms zur Verfügung; der zweite ist für die Eingabe über die Tastatur des Rechners und für das Zuordnen des (numerischen) Wertes zu einer Variablen mit dem Namen WAHL zuständig.

## Programmblock AUSFUH

Der Wert dieser Variablen wird vom nächsten Programmblock, AUSFUH, verwendet, um einen der neun weiteren Programmblöcke aufzurufen. Alle, ausgenommen ENPROG, müssen nach Ausführung wieder zu *AUWAHL* zurückkehren, so daß der Anwender eine neue Auswahl treffen kann. Bei Auswahl von 9 (ENPROG) ist dies nicht nötig, da mit dieser Routine das Programm beendet wird.

Das Hauptproblem des Programms in dieser Form ist, daß der Ablauf noch nicht korrekt ist. INITIL versucht, von einem Speichermedium eine Datei einzulesen, ohne zu überprüfen, ob überhaupt eine Datei existiert. Wenn das Programm das erste Mal gestartet wird, sind noch keine Daten eingegeben und somit ist auch noch keine Datei auf Cassette oder Diskette gespeichert worden. Jeder Versuch, eine Datei zu öffnen und zu lesen, führt daher zu einer Fehlermeldung.

Vernünftiger ist es, die *LSINDT*-Routine nur von einem der AUSFUH-Module aufzurufen, und zwar nur einmal pro Programmlauf. Das bedeutet, daß ein INDT-Flag gesetzt werden muß. Originalwert ist 0, bis er auf 1 gesetzt wird, wenn eine Datei eingelesen wurde. Wenn der Wert 1 ist, kann keine Datei mehr eingelesen werden. ADDVER sucht dann in den Bereichen nach dem ersten freien Element und legt die Daten dort ab.

Hinzufügen, Löschen oder Modifizieren eines Verzeichnisses bedeutet, daß die Reihenfolge der Verzeichnisse nicht mehr in Ordnung ist. Die Routine FNDVER sollte also jeweils vor der Ausführung dieses Flag überprü-

```
10 REM 'HAUPTPROGRAMM'
20 REM * INITIL *
30 GOSUB 1000
40 REM * BGRUES *
50 GOSUB 3000
60 REM * AUWAHL *
70 GOSUB 3500
80 REM * AUSFUH *
90 GOSUB 4000
100 END
1000 REM * INITIL * UNTERROUTINE
1010 GOSUB 1100: REM *CREARR* (DEFINIERE BEREICHE) UNTERROUTINE
1020 GOSUB 1300: REM *LSINDT* (LESE DATEI EIN) UNTERROUTINE
1030 GOSUB 1320: REM *SETFLG* (SETZE FLAGS) UNTERROUTINE
1040 REM
1050 REM
1060 REM
1070 REM
1080 REM
1090 RETURN
1100 REM *CREARR* (DEFINIERE BEREICHE) UNTERROUTINE
1110 DIM NAMFLD$(50)
1120 DIM MODFLD$(50)
1130 DIM STDFLD$(50)
1140 DIM STAFLD$(50)
1150 DIM TELFLD$(50)
1160 DIM INXFLD$(50)
1170 REM
1180 REM
1190 REM
1200 REM
1210 LET GROS=0
1220 LET VMOD=0
1230 LET SVED=0
1240 LET CURR=0
1250 REM
1260 REM
1270 REM
1280 REM
1290 RETURN
1300 REM *LSINDT* UNTERROUTINE — SIEHE BASIC 'DIALEKTE'
1310 ON ERROR GOTO 1370
1320 OPEN "I",#1, "ADBK.DAT"
1330 FOR L=1 TO 50
1340 INPUT #1 NAMFLD$(L), STRFLD$(L),STDFLD$(L), STAFLD$(L),TELFLD$(L)
1350 NEXT L
1360 CLOSE #1
1370 RETURN
1380 REM TESTWEISE *SEFFLG* ROUTINE
1390 RETURN
3000 REM * BGRUES * UNTERROUTINE
3010 PRINT
3020 PRINT
3030 PRINT
3040 PRINT
3050 PRINT TAB(13);"*WILLKOMMEN ZUM*"
3060 PRINT TAB(11);"ADRESSBUCH-PROGRAMM*"
3070 PRINT TAB(8);"DES COMPUTER-KURSES*"
3080 PRINT
3090 PRINT TAB(1);"(DRUECKE DIE LEERTASTE, UM FORTZUFAHREN)"
3100 FOR L=1 TO 1
3110 IF INKEY$ <> " " THEN L=0
3120 NEXT L
3130 PRINT CHR$(12)
3140 RETURN
3500 REM * AUWAHL * UNTERROUTINE
3510 REM
3520 REM 'CHMENU'
3530 PRINT CHR$(12)
3540 PRINT "WOLLEN SIE"
3550 PRINT
3560 PRINT
3570 PRINT
3580 PRINT "1. VERZEICHNIS DURCH NAMEN SUCHEN"
3590 PRINT "2. NAMEN DURCH TEIL EINES NAMENS SUCHEN"
3600 PRINT "3. VERZEICHNISSE NACH STADTANGABEN SUCHEN"
3610 PRINT "4. VERZEICHNISLISTE DURCH INITIALEN"
3620 PRINT "5. LISTE ALLER VERZEICHNISSE"
3630 PRINT "6. HINZUFUEGEN EINER ADRESSE"
3640 PRINT "7. AENDERN EINER ADRESSE"
3650 PRINT "8. LOESCHEN EINER ADRESSE"
3660 PRINT "9. PROGRAMM BEENDEN UND DATEN SPEICHERN"
3670 PRINT
3680 PRINT
3690 REM 'INWAHL'
3700 REM
3710 LET L=0
3720 LET I=0
3730 FOR L=1 TO 1
3740 PRINT "WAEHLEN SIE (1—9)"
3750 FOR I=1 TO 1
3760 LET A$=INKEY$
3770 IF A$=" " THEN I=0
3780 NEXT I
3790 LET WAHL=VAL(A$)
3800 IF WAHL < 1 THEN L=0
3810 IF WAHL > 9 THEN L=0
3820 NEXT L
3830 RETURN
4000 REM * AUSFUH * UNTERROUTINE — SIEHE 'BASIC-DIALEKTE'
4010 ON WAHL GOSUB 310,330,350,370,390,410,430,450,470
4020 RETURN
9000 REM * ADDVER * UNTERROUTINE
9010 PRINT CHR$(12)
9020 INPUT "GIB NAMEN EIN";NAMFLD$(GROS)
9030 INPUT "GIB STRASSE EIN"; STRFLD$(GROS)
9040 INPUT "GIB STADT EIN";STDFLD$(GROS)
9050 INPUT "GIB STAAT EIN"; STAFLD$(GROS)
9060 INPUT "GIB TELEFONNUMMER EIN";TELFLD$(GROS)
9070 LET VMOD=1
9080 LET INXFLD$(GROS)=STR$(GROS)
9090 GOSUB * MODNAM *
9100 RETURN
```

fen. Sind Änderungen vorgenommen worden, kann entweder vor dem Suchen eine Sortierung vorgenommen oder aber trotzdem der ganze „Haufen" durchsucht werden. ENPROG überprüft VMOD automatisch und ruft (wenn der Wert 1 ist) die Sortier-Routine auf, bevor die Daten auf Cassette oder Diskette gespeichert werden.

Die bereits angesprochenen Aspekte zur Programmoptimierung lassen sich in die folgenden Kategorien einteilen:

User Interface
User Image
Fehlerbeseitigung
Sicherheit
Adaptions-Möglichkeiten

„User Interface" bezieht sich auf die Art und Weise, in der der Anwender mit dem Programm kommuniziert. Wir haben uns für die Verwendung von Menüs entschieden (im Gegensatz zu Befehlen).

In der momentanen Form ist das Programm in bezug auf Anwenderfreundlichkeit noch nicht gut. Die Begrüßungsmeldung wird durch Drücken der Leertaste fortgesetzt; das Auswahl-Menü wird durch Drücken einer Zahlentaste von 1 bis 9 automatisch verlassen; und letztlich wird die Dateneingabe durch Drücken von RETURN (für jedes Feld) abgeschlossen.

„User Image" bezieht sich darauf, wie der Anwender die Arbeit mit dem Programm empfindet. Dies wird wesentlich durch das „User Interface" beeinflußt. Die meisten Vorgänge innerhalb eines Computers sind für den Anwender nicht sichtbar. Die einzigen Rückschlüsse kann er aus den visuellen Darstellungen auf dem Bildschirm ziehen, die wiederum Resultat seiner Eingaben über die Tastatur sind. Das „User Image", das wir für das Adreßbuchprogramm erreichen wollen, soll dem eines „normalen" Adreßbuches entsprechen.

## Übersichtliche Darstellung

Ähnlich sollte das „User Image" eines Textverarbeitungsprogramms einem Blatt Papier entsprechen, auf dem man schreiben kann.

Das „User Image" eines Programms ist dann am besten, wenn alle Daten klar und übersichtlich dargestellt werden (z. B. in Form eines Blattes Papier oder einer Karteikarte), anstatt mit abstrakten „Unter-Dateien", „Buffern" oder ähnlichem zu arbeiten. Viele kommerzielle Datenverwaltungsprogramme schenken diesem Aspekt nur wenig Bedeutung; der Anwender muß sich merken, welche Teilinformationen in welchen Unter-Dateien oder temporären Feldern „versteckt" sind.

Die Fehlerbeseitigung ist ein ebenso wichtiger Aspekt. Was passiert, wenn Sie gerade den Namen einer Person eingegeben haben und dann bemerken, daß Sie einen Tippfehler gemacht haben? Müssen Sie die Eingaben vollenden und dann in das Menü zurückkehren, um MODVER zur Korrektur aufzurufen, oder gestattet das Programm eine direkte Korrektur? Die meisten BASIC-Versionen geben Fehlermeldungen bei der Eingabe eines Programms, entweder bei der Eingabe der fehlerhaften Zeile oder wenn das Programm gestartet wird. Doch dies ist kein Bestandteil des „User Interfaces". BASIC beinhaltet eine Anzahl von Meldungen, die dem Anwender eine fehlerhafte Eingabe aufzeigen (beispielsweise die Meldung RED0, wenn bei einer INPUT-Anweisung eine falsche Eingabe vorgenommen wurde).

Die Handhabung von Fehlern umfaßt zwei Fakten – Fehlermeldungen und Fehlerbeseitigung. Ein bekanntes Textverarbeitungsprogramm beispielsweise verfügt über ausführliche Fehlermeldungen, jedoch eine schlechte Fehlerbeseitigung. Wenn man ein sehr großes Dokument erstellt und dieses auf einer nahezu vollen Diskette abspeichern will, erhält man die hilfreiche Meldung „DISK SPACE EXHAUSTED" (Diskette voll). Unglücklicherweise wird dem Anwender jedoch nicht ermöglicht, eine neue Diskette zu formatieren, ohne den eventuell in vielen Stunden eingegebenen Text aus

## BASIC-Dialekte

SPECTRUM

Betrachten Sie die Zeilen 1300 bis 1370 des Programmlistings. Beim Spectrum ist zu beachten, daß die Schleife zwischen den Zeilen 3750 und 3780 zwar läuft, jedoch aufgrund der Tastaturwiederholfunktion eventuell zu Problemen führt. Im Handbuch wird angegeben, daß dieses Problem wie folgt behoben werden kann:

```
3755 IF INKEY$ <> ""
     THEN GOTO 3755
```

Die Funktion VAL (A$) ist auf dem Spectrum verfügbar, doch führt sie zu einem Programmabsturz, wenn das erste eingegebene Zeichen alphanumerisch ist. In diesem Programm kann das Problem wie folgt behoben werden:

```
3790 LET WAHL=CODE A$-48
```

Dies ist jedoch keine komplette Lösung. Sie funktioniert nur, wenn A$ nur ein einziges Zeichen enthält (wie in diesem Programm vorgesehen). Der Spectrum verfügt nicht über die Anweisung ON...GOSUB, doch gestattet er Ihnen, GOSUB (Berechnung) und GOSUB (Zeilennummer) zu verwenden. Zeile 4010 kann somit wie folgt umgestellt werden:

```
4010 GOSUB (290+WAHL*20)
```

Der Spectrum verfügt nicht über die RENUMBER-Funktion.

dem Arbeitsspeicher zu löschen.

Jede Operation, die durch den Anwender vorgenommen wird und zu Datenverlust führen könnte (z. B. bei MODVER), sollte immer vor der Ausführung verifiziert werden. Es sollten immer Meldungen wie z. B. „DAS VERZEICHNIS WIRD GELOESCHT. SIND SIE SICHER? (J/N)" erscheinen. Bei einem Textverarbeitungsprogramm lautet eine Meldung: „DER ‚SICHERN'-BEFEHL LÖSCHT DIE ALTE VERSION DES TEXTES. IST DAS OK? (J/N)".

## Auf Fehlersuche

Das Handling von Fehlern sollte bei der Entwicklung eines Programms ständig berücksichtigt werden. Ganz besonders wichtig ist dies bei Routinen, in denen fehlerhafte Daten, Menüauswahlen oder Befehle eingegeben werden können, sowie dann, wenn Daten modifiziert oder gespeichert werden oder wenn alte Daten überschrieben werden.

Sie müssen auch auf die Sicherheit des Programms achten. Was passiert mit dem Programm oder den Daten, wenn ein fataler Fehler auftritt (z. B. bei einem Stromausfall)? Der Programmentwickler sollte bestimmen, wie viele Daten maximal verlorengehen dürfen und welche Methoden zur Wiederherstellung oder Rekonstruktion der übrigen Daten möglich sind. Ein sehr hochentwickeltes Textverarbeitungsprogramm umfaßt ein Programm mit Namen RECOVER. Durch diese Vorrichtung geht selbst bei einem Stromausfall oder dem versehentlichen Ausschalten des Computers vor Sicherung der Texte so gut wie nichts verloren. Derart fortgeschrittene Programmiertechniken übersteigen leider den Rahmen dieses Kurses. Trotzdem sollten Sie versuchen, Ihre Programme so sicher wie möglich zu gestalten und alle erdenklichen Fehler möglichst von vornherein auszuschließen.

Die Adaptions-Möglichkeiten sind ebenso wichtig. Wir haben uns mit diesem Fakt bereits befaßt. Im einfachsten Fall lassen Sie möglichst viel Abstand zwischen den Zeilennummern (in BASIC) und fügen leere REMs ein, die später gegebenenfalls durch weitere Anweisungen ersetzt werden können. Wenn Sie Datenfelder definieren, integrieren Sie mindestens einen zusätzlichen Bereich für zukünftige Erweiterungen. Es ist eine Kardinalregel in der Programmierung, daß zukünftige Bedürfnisse nicht vorhersehbar sind. Das einzig Sichere ist, daß ein gutes Programm immer noch verbessert werden kann.

**MERGE**

Diese Anweisung ist bei einigen Computern wie Oric, Commodore 64, VC 20, Acorn B usw. nicht vorhanden. Trotzdem kann der MERGE-Befehl oftmals simuliert werden. Der Lynx verfügt über den Befehl APPEND. Dieser Befehl gestattet Ihnen, ein Programm an das Ende eines bereits im Speicher befindlichen Programmes zu laden. Vorausgesetzt, die erste Zeilennummer des einzuladenden Programms ist höher als die letzte Zeilennummer des bereits im Speicher befindlichen Programms.

**RENUMBER**

Dieser Befehl wird weder vom Spectrum noch von den Commodore-Rechnern oder dem Oric angeboten. Bei diesen Maschinen kann eine neue Durchnumerierung nur per Hand durchgeführt werden.

**INKEY$**

Siehe vorangegangene BASIC-Dialekte.

**OPEN CLOSE**

Siehe vorangegangene BASIC-Dialekte.

**PRINT CHR$(12)**

Ersetzen Sie diese Anweisung gegebenenfalls durch CLS (außer beim Commodore 64 und beim VC 20). Bei diesen Computern tippen Sie PRINT "SHIFT-Taste+CLR-Taste", worauf ein invertiertes Herz oder ein großes S zwischen den Anführungszeichen erscheinen sollte.

**ON... GOSUB**

In Zeile 4010 bewirkt ON...GOSUB den Sprung zu nicht existierenden Zeilen – 310, 330, 350, usw. –, wodurch das Programm abstürzen würde. Diese Zeilen werden später noch für die Menü-Unterroutinen eingefügt. In diesem Stadium ersetzen Sie diese Zeilen testweise durch:

```
310 RETURN
330 RETURN
usw.
```

**STR$**

Der Lynx kennt diese Anweisung nicht. Ersetzen Sie daher Zeile 9080 durch:

```
9075 N=GROS
9077 GOSUB 9500
9080 INDFLD$(GROS)=N$
```

Fügen Sie außerdem die folgende Unterroutine ein:

```
9500 REM KONVERTIERUNG VON
     N IN N$, WOBEI N GANZZAH-
     LIG SEIN MUSS
9510 LET N$=" "
9520 IF N < 0 THEN LET N$=""
9530 N=ABS(N)
9540 X=10
9550 I=1
9560 FOR K=1 TO 8
9570 I=I*X
9580 R=K
9590 IF I > N THEN K=8
9600 NEXT K
9610 FOR K=1 TO R
9620 I=I/X
9630 Z=INT(N/I)
9640 LET N=N-Z*I
9650 N$=N$+CHR$(48 + INT(Z))
9660 NEXT K
9670 RETURN
```

# Komplett-System

**Hier werden einige Erweiterungs-Möglichkeiten des Spectrum gezeigt, die von verschiedenen Herstellern angeboten werden.**

**Akustikkoppler**
Der hier gezeigte Micro-Myte 60 macht es möglich, daß ein Computer mit dem anderen kommuniziert.

**Microdrive**
Dieses Steckmodul enthält ein Endlosband, das innerhalb von sieben Sekunden umläuft. Die Datenübertragung findet mit einer Rate von 6 KByte je Sekunde statt, das ist viermal schneller als bei einem normalen Cassettenrecorder. Bis zu acht Microdrives können aneinandergesteckt werden und ergeben dann eine Gesamtkapazität von 700 KByte oder mehr.

**Tastatur**
Die FDS-Tastatur von Fuller ist mit zusätzlichen Funktionstasten ausgestattet.

**Joysticks**
Mit dem Sinclair-Interface 2 kann jeder Joystick, der eine Atari-Schnittstelle hat, verwendet werden, gleichgültig, nach welchem Prinzip er arbeitet. Gleichzeitig können zwei Joysticks angeschlossen werden.

**RAM-Modul**
Der kleinere Spectrum mit 16 KByte kann durch ein RAM-Modul um 32 KByte erweitert werden.

Der Spectrum von Sinclair wurde 1982 als „echter Durchbruch" auf dem Heimcomputersektor auf den Markt gebracht. Ein Jahr später hielt der Spectrum mit 600 000 verkauften Einheiten einen Marktanteil von mehr als der Hälfte aller Heimcomputer in England – eine Tatsache, die selbst die Hersteller überraschte. Mit 16 oder 48 KByte RAM als Standard, acht Farben für den Rahmen, den Hintergrund und die Texte und einer begrenzten hochauflösenden Grafikfähigkeit, der verbesserten Tastatur sowie der Fähigkeit, einfache Töne zu erzeugen, stellte der Spectrum eine beachtliche Verbesserung gegenüber dem früheren Modell von Sinclair, dem ZX81, dar. All diese Eigenschaften konnten jedoch andere unabhängige Hersteller nicht davon abhalten, eine Vielzahl von Zubehör zu produzieren. Sinclair selbst beeilte sich, einen Massenspeicher, Schnittstellen für steckbare ROM-Module und Joysticks zu entwickeln.


**Bildschirmgerät**
Da der Spectrum keinen Monitoranschluß besitzt, ist die präzise Grafikdarstellung schwierig. Hier der Profeel-Bildschirm von Sony, dessen Empfängerteil vom Monitor getrennt ist.

**Serieller Drucker**
Neben dem ZX-Drucker kann auch ein herkömmlicher Matrixdrucker oder Typenraddrucker angeschlossen werden. Mit der Sinclair-Schnittstelle 1 sind serielle Peripheriegeräte ansteuerbar.

**Musiksynthese**
Der Trichord-Musiksynthesizer von Petron ist ein gutes Beispiel für handelsübliches Zubehör. Trichord ist eine Dreierakkordeinheit, mit der bis zu 6134 aus drei Noten bestehende Akkorde auf dem 48er Spectrum erzeugt werden können.

**Cassettenrecorder**
Selbst mit angeschlossenem Microdrive kann der Spectrum mit einem handelsüblichen Cassettenrecorder Daten, etwa Software, austauschen.

**Sprachsynthese**
Der Cheetah-Sweet-Talker ist ein Synthesizer, der nach dem vereinfachten „allophonischen" System Sprache dadurch erzeugt, daß Wörter aus ihren Silbenbestandteilen aufgebaut werden. Es stehen 63 Allophone und vier unterschiedliche Zwischenraumlängen zur Verfügung.

**Interface 2**
Verschiedene Hersteller liefern programmierbare Joystick-Schnittstellen. Aber die von Sinclair selbst gelieferte Schnittstelle enthält eine ROM-Buchse, so daß Spiele und Alternativsprachen leichter zu installieren und zu benutzen sind.

**ZX-Drucker**
Für Heimcomputer-Anwender, die den Drucker hauptsächlich zum Testen von Programmen benötigen, bietet Sinclair diesen Niedrigpreis-Drukker, der mit einer Art „Funkenerosion" auf aluminisiertes Papier druckt.

**Tastatur**
Die Standard-Tastatur des Spectrum stellt zweifellos eine beachtliche Verbesserung gegenüber der Z80- und Z81-Tastatur dar, eignet sich allerdings nicht zum Schnellschreiben.

**Interface 1**
Diese Schnittstelle ist zwar speziell für den Microdrive ausgelegt, enthält aber auch Anschlußmöglichkeiten für andere Geräte und ermöglicht die Vernetzung mehrerer Spectrum-Computer.

# Vannevar Bush

**Sein elektromechanischer Computer konnte auch schwierige Differentialgleichungen lösen.**

**Der Differential-Analysator**

Diese Maschine wurde für die Lösung einer wichtigen Gruppe mathematischer Probleme gebaut, die in Technik und Wissenschaft häufig vorkommen – die Differentialgleichungen zweiter Ordnung. Die zuerst von Lord Kelvin vorgeschlagene Methode dazu bestand in der Eingabe der Ergebnisse eines „Integrierers" (Gerät, das die Fläche unter einer Kurve bestimmt) in einen zweiten. Vor der Erfindung von Verstärkern war dies jedoch wegen der zu schwachen Ausgangssignale eines Integrierers nicht möglich. Bushs 1931 entwickelte Maschine war vollständig mechanisch aus komplizierten Getrieben, Achsen und Elektromotoren aufgebaut. Ein- und Ausgabe wurden durch Rotation bewerkstelligt, das Problem der Rückkopplung durch einen Drehmomentverstärker beseitigt. In den 40er Jahren wurde ein verbesserter Differential-Analysator mit elektrischen Komponenten gebaut, der jedoch mehr als hundert Tonnen wog. Die Ergebnisse erschienen in digitaler Form und waren auf 5 Stellen genau. Die Maschinensteuerung ging mit Lochstreifen vonstatten. Während des Zweiten Weltkriegs wurde das Gerät für die Verschlüsselung und außerdem für ballistische Berechnungen eingesetzt.

Von vielen Fachleuten wird Vannevar Bush als „Vater des Computers" bezeichnet. Er hatte schon im Jahre 1931 mit dem Differential-Analysator ein mechanisches Gerät entwickelt, das die Forschung auf den richtigen Weg zu den heutigen digitalen Rechnern brachte.

Bush wurde am 11. März 1890 in Boston, Massachusetts, geboren. Wie schon sein Vater, studierte auch der junge Bush die Ingenieurwissenschaften und erhielt 1913 sein Diplom. Nach einer kurzen Beschäftigung bei General Electric wurde er Lehrbeauftragter an seiner ehemaligen Hochschule, bildete sich aber später in Harvard und am Massachusetts Institute of Technology (MIT) weiter. Während des Ersten Weltkrieges war Bush an der Entwicklung von U-Boot-Aufspürgeräten für die US-Marine beteiligt.

Schon als Student machte Bush seine erste Erfindung: ein Gerät zur Landvermessung. Der Mechanismus war in einem Fahrradrahmen aufgehängt und zeichnete während der Fahrt ein Höhenprofil des Geländes. Die neuartige „Vermessungsmaschine" enthielt einen sogenannten „Integrierer", da für die Höhenbestimmung an jedem Punkt alle vorher registrierten Werte berücksichtigt werden mußten.

Bush wurde Dozent für elektrische Energieübertragung am MIT und beschäftigte sich dort intensiv mit einem schwerwiegenden Problem der Energieversorgung: Stromausfälle durch unvorhergesehene, plötzliche Verbrauchsspitzen sollten verhindert werden. Die mathematischen Grundlagen für die Behandlung dieses Problems hatte der schottische Wissenschaftler C. Maxwell mit den nach ihm benannten Gleichungen schon im 19. Jahrhundert gelegt. Der Rechenaufwand für die Vielzahl der zu lösenden Gleichungsaufgaben war jedoch für reine „Handarbeit" zu groß, so daß Bush sich an die Entwicklung einer Maschine für diesen Zweck machte. Dabei konnte er sich auf die Arbeiten von Lord Kelvin stützen, der schon eine Vielzweck-Maschine für die mathematische Vorausberechnung von Ebbe und Flut entworfen hatte.

## Der „Produkt-Telegraf"

Erster Erfolg der Forschung war der „Produkt-Telegraf". Mittels eines Potentiometers wurde eine Wellenform in elektrische Spannung umgewandelt, die einem eigens konstruierten Wattmeter (Stromzähler) zugeführt werden konnte. Dieses wiederum „berechnete" ein „Produkt" aus Spannung und Strom.

Der Erfolg des Produkt-Telegrafen bei der Lösung von Gleichungen spornte den Forscher an, sich an ein Gerät für die Bewältigung von Differentialgleichungen zweiter Ordnung zu wagen. 1931 entstand der erste Differential-Analysator, der in Europa und England mehrfach nachgebaut wurde. Sogar die amerikanische „Moore School of Electrical Engineering" – an der später der ENIAC-Computer gebaut wurde – bestellte ein Exemplar. Die Rechengenauigkeit des Produkt-Telegrafen lag nur bei 2%, mit dem neuen Differential-Analysator konnten aber Ergebnisse auf 0,05% exakt ermittelt werden. Die um nicht einmal zwei ganze Stellen verbesserte Genauigkeit verhundertfachte allerdings bei mechanischen Geräten deren Preis – eine ähnliche Verbesserung eines heutigen Digitalrechners ist höchstens doppelt so teuer.

Bush wurde Dekan der Ingenieurschule und 1939 auch Vizepräsident des Carnegie-Institutes. Als Chef der mit Finanzmitteln reichlich ausgestatteten Militärforschung war Bush auch maßgeblich am Manhattan-Projekt beteiligt – der Entwicklung der amerikanischen Atombombe. 1955 zog es ihn aus dem Beruf ins Privatleben. Vannevar Bush starb 1974.

# Lehren und Lernen

**Computer spielen im Schulsystem zwei unterschiedliche Rollen. Sie werden sowohl zum Studium der Computertechnologie selbst benutzt als auch zum Lernen eingesetzt, etwa als interaktives Lehrbuch.**

Im Jahre 1980 wurde in England ein Lehrplan veröffentlicht, mit dem das Computerverständnis an Grund- und weiterbildenden Schulen verbessert werden sollte. Unter dem Namen „Microcomputers in Education Project" (Microcomputerprojekt in der Erziehung) oder kurz MEP sah man einen Gesamtzeitraum von sechs Jahren vor und stellte dazu 21 Millionen Pfund zur Verfügung. Es wäre unrealistisch, diese Summe durch 25 000 zu teilen (was der Gesamtzahl aller englischen Schulen entspräche) und daraus die Verteilung der Geldmittel abzuleiten. Das vorrangig empfohlene Gerät war der Acorn B mit einem Preis von umgerechnet 1300 Mark. Das Fünffache hätte für das andere in Frage kommende System, Research Machine 380 Z, angesetzt werden müssen. Die zu geringe finanzielle Ausstattung des Projekts wurde offensichtlich, womit die Schulen gezwungen wurden, auf Eigenmittel zurückzugreifen.

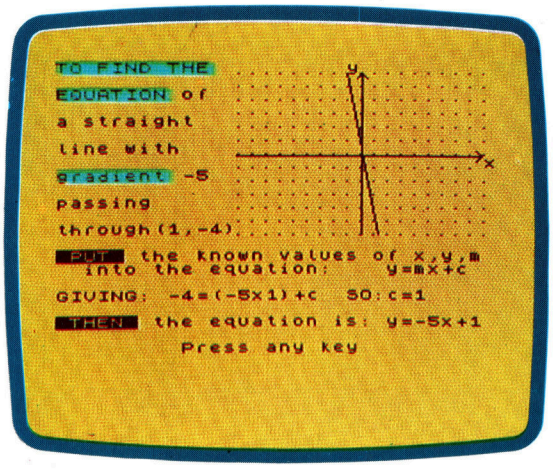
## Kostspielige Ausrüstung

1983 verkündete das Ministerium für Informationstechnologie, daß alle Sekundarschulen (4553 in England) und mehr als die Hälfte der Grundschulen mit einem Computer ausgestattet seien. Grundlage für diesen Erfolg waren im wesentlichen die Bemühungen von Elternvereinigungen, Wohlfahrtsverbänden und schließlich der Schulkinder selber, die beachtliche Spendenbeträge sammelten.

Die Initiative der britischen Regierung scheint die Probleme eher verschlimmert zu haben, trotz der Bemühungen, den Lehrplan den technischen Gegebenheiten anzupassen. Aufgrund des Bedarfs an kostspieliger Ausrüstung ist der Unmut von Lehrern wie Schülern gewachsen. Der immer größer werdende „Vorsprung" von Kindern aus wohlhabenden Familien denen weniger begüterter Herkunft gegenüber wird offensichtlich; denn erstere können mehr Geld zur Realisierung solcher Projekte aufbringen. Wegen der geringen Anzahl von Microcomputern an den einzelnen Schulen stehen dem „durchschnittlichen" Schüler die Rechner lediglich 15 Minuten pro Woche zur Verfügung. Zu wenig, um das angestrebte Computerverständnis zu erlangen, ge-


**Viele Heimcomputer-Besitzer erwerben Lernsoftware für ihre Kinder. Für kleine Kinder gibt es Programme, die den Lernprozeß einleiten; ebenso Software für ältere Schüler als Vorbereitung bei Klassenarbeiten.**

**Programme, die dem Schüler beim Rekapitulieren helfen, sind in ihrer Darstellungsbreite begrenzt. Mit solchen Programmen werden Einzelthemen der verschiedenen Fächer detailliert behandelt. Die Darstellung rechts zeigt mathematische Gleichungen, die auf dem Spectrum dargestellt werden. Rechts daneben sehen Sie die Bildschirmanzeige eines Geometrie-Programms. In den anderen Beispielen werden das Ohmsche Gesetz und die Konstruktion eines Verstärkers veranschaulicht.**




**„Equations And Inequalities", Rose Software**

schweige denn, die Vorteile interaktiver Lernsoftware nutzen zu können.

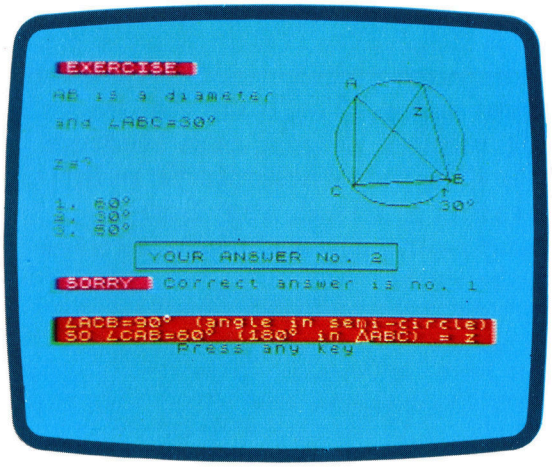
Eben diese Unzulänglichkeit mag Ursache dafür sein, daß so viele Heimcomputer für den privaten Haushalt erworben wurden – nämlich um Kindern die Möglichkeit des Lernens mit dem Computer zu geben.

Vom Hardware-Problem einmal abgesehen, steht die Rolle der Lernsoftware als Lehrhilfe außer Frage. Die Erstellung solcher Software ist vergleichsweise einfach, da man sich dabei derselben Methoden bedient, die auch beim Schreiben von Spielprogrammen angewendet werden. Die Bezeichnung „interaktive Lehrbücher" wurde gewählt, da sie viele Gemeinsamkeiten mit den Lehrmitteln in gedruckter Form haben. Der Ruf des Autors ist ein ebenso wichtiges Kaufargument wie beim herkömmlichen Lehrbuch, und das im Buchhandel übliche informative Verkaufsgespräch findet auch bei diesem neuen Medium statt, eben durch überzeugungsfähige Vertreter.

Der wesentliche Unterschied aber besteht in der Einführung des Konzepts der Interaktion. Traditionell hat der Schüler zwei Wissensquellen: den Lehrer und das Lehrbuch. Die Beziehung zu einem Lehrer ist nur bis zu einem bestimmten Grad interaktiv. Von wenigen Ausnahmen abgesehen sind 30 und mehr Schüler in einer Klasse. Basierend auf zwei Unterrichtsstunden pro Woche ist jedem Kind die ungeteilte Aufmerksamkeit des Lehrers bestenfalls für vier Minuten sicher. Deshalb ist es nicht überraschend, daß Erzieher nach effektiveren Lehrmethoden suchten und Lehrmaterial entwickeln, das Interaktion erlaubt.

Erste Versuche in dieser Hinsicht waren die „Sprachlabore" der sechziger Jahre, die vorrangig der Vermittlung von Fremdsprachen dienten. Jedem Schüler standen ein Cassettenrecorder und ein aufgezeichneter Text zur Verfügung, der durchgearbeitet werden mußte. Der Lehrer konnte sich in den Audiokanal jedes einzelnen Schülers bei Bedarf einschalten. Ziel jedoch war es, diesen Eingriff weitmöglichst zu reduzieren.

Als logische Schlußfolgerung bot sich der computer-gestützte Unterricht, der auf diese Eingriffsmöglichkeit noch mehr verzichtet. Das setzt eine perfekte Lehrmethode voraus, ein System, in dem es keine Ungereimtheiten gibt; Lehrstoff, der so gestaltet ist, daß der Lernende auf natürliche Weise weitergeführt wird. Das heißt, daß der Anwender nicht über Computerkenntnisse – sei es als Operator oder




**„Geometry", Rose Software**

Programmierer – verfügen muß, um damit umgehen zu können.

Bei der Betrachtung der Software ist zwischen derjenigen naturwissenschaftlicher und derjenigen geisteswissenschaftlicher Art zu unterscheiden. In der Naturwissenschaft geht es um bekannte und quantifizierbare Fakten, in der Geisteswissenschaft werden dieselben Fakten subjektiv analysiert. Dieser Unterschied wird beim computergestützten Unterricht noch deutlicher und größer. Naturwissenschaftliche Fächer werden stark illustriert vermittelt – selbst mit der Einschränkung, daß die eingesetzten Microcomputer eventuell nur über relativ geringe Grafikmöglichkeiten verfügen. Dagegen sind geisteswissenschaftliche Fächer weitestgehend textorientiert.

Ergänzend zu fachbezogenem Lehr- und Übungsmaterial ist reichhaltiger „Stoff" für den Grundschulbereich entwickelt worden – etwa Grundrechenarten, Lesen und Schreiben.

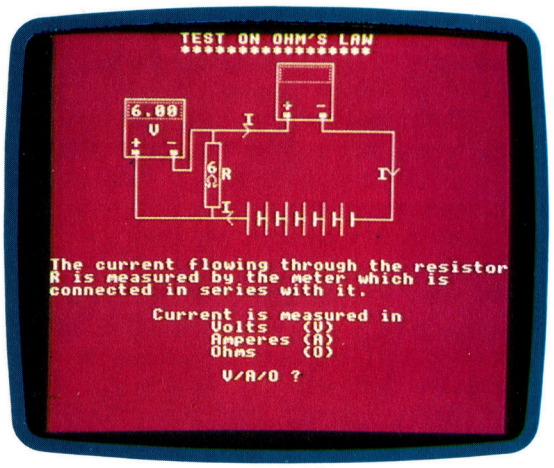
Lernsoftware kann didaktisch oder unter dem Aspekt „Forschung" angelegt sein. Man präsentiert Fakten auf bestimmte einprägsame Art oder vermittelt sie, indem man die Wahl zwischen mehreren Antworten bietet. Ein dritter Software-Typ findet mehr in der Schule als zu Hause Anwendung: experimentelle Simulations-Software. Die dabei angewandten Programmiertechniken sind weitaus komplexer als etwa bei Mathematik-Lehrprogrammen. Denn es geht um die Darstellung physikalischer Gesetzmäßigkeiten, die verständlich werden sollen. Die Anwendung solcher Simu-

lationsmethoden im Klassenzimmer ist für Lehrer naturwissenschaftlicher Fächer ebenso interessant wie für Fachleute in der Industrie. Der Grund: So wird die Möglichkeit des ungefährlichen Experimentierens für wenig Geld gegeben, was etwa beim Chemieunterricht ein unschätzbarer Vorteil ist.

Lernsoftware wird für drei Altersgruppen angeboten, um die speziellen Programme gezielt einsetzen zu können. Die jeweiligen Unterschiede und Ähnlichkeiten dieser Software-Pakete werden nachstehend aufgezeigt.

## Acht und jünger

Programme für kleinere Kinder dienen der Entwicklung und Vermittlung von Grundwissen und Mustererkennung. Bei vielen Programmen werden einfache arithmetische oder sprachliche Aufgaben gestellt, um so mit der Grundsymbolik vertraut zu machen. Die Methodik hat




**„D. C.", SciCAL Software**

häufig einen spielerischen Charakter, damit die Aufmerksamkeit des Kindes erhalten bleibt. Bei manchen Programmen wird die Präsenz eines Erwachsenen oder älteren Kindes vorausgesetzt.

Übliche Programme für diese Altersgruppe lehren das Kind, die Uhrzeit zu lesen, zu zählen, zu addieren und zu subtrahieren (dargestellt mit Hilfe einer Waage), auch der Aufbau kurzer Sätze und das Buchstabieren werden dabei gelernt.

## Neun bis vierzehn

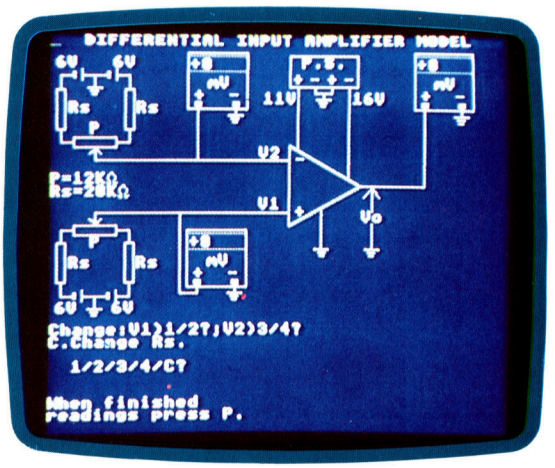
In dieser Altersgruppe wird auf spielerische Elemente in den Programmen verzichtet. Sie gehen disziplinierter vor und sind so den Erfahrungen des Kindes in der Schule angepaßt. Software dieser Art will das Kind zu seiner Benutzung motivieren, was mittels „Belohnung" geschieht. Diese Methode besteht beispielsweise darin, nach erfolgreichem Abschluß eines Lernabschnittes innerhalb einer bestimmten Zeitspanne ein Spiel freizugeben. Es ist natürlich im Programm enthalten.

Rechnen sowie Rechtschreibung und Sprache generell sind Haupteinsatzgebiete von Software in dieser Altersgruppe. Ergänzend dazu gibt es Geschichts- und Erdkundeprogramme, ferner Musikgrundlagen und einfache Simulationsprogramme.

## Fünfzehn und älter

In diesem Alter stehen Prüfungen und Abschlüsse an. Entsprechend komplexer ist die Lernsoftware gestaltet. Es gibt Programme, mit denen Grundwissen wiederholt wird, doch die meisten Anbieter richten ihre Programme auf spezielle Belange innerhalb der Fächer aus. Dabei geht es gelegentlich um elektronische Tests, die den Prüfungsvorschriften für Klassenarbeiten entsprechen. Andererseits vertiefen spezielle Programme das erlernte Grundwissen, verdeutlichen Theorie und Methodik des Stoffes unter anderen Gesichtspunkten und fragen dann den Stoff nach dem Prinzip der „Multiple Choice" ab. Dabei werden mehrere mögliche Antworten vorgegeben, von denen die Schüler die richtigen auswählen sollen. Und bei Fächern wie Literatur wird sogar eine Analyse von Stil und Inhalt durchgeführt, genauso, wie es ein Lehrer auch tun würde.

In diesem Stadium erwartet man von den Schülern einen bestimmten Grad von Eigen-




**„Ampli.", SciCAL Software**

motivation. Folglich verzichtet man darauf, sie mit spielerischen Mitteln an Themen zu interessieren. Was allerdings nicht bedeutet, daß der Stoff langweilig dargeboten wird. Vielmehr bedient man sich stehender wie bewegter Grafiken und benutzt auch Soundeffekte zur Wissensvermittlung. Es gibt eine Reihe kompakter Lernprogramme für diese Altersgruppe. Und die richtigen für die betreffenden Fächer auszuwählen, sollte man einen entsprechenden Fachlehrer zu Rate ziehen, der ihren „Lehrwert" beurteilen kann.

# Interne Funktionsabläufe

**Mit der Maschinensprache lassen sich alle Funktionen des Microprozessors direkt steuern. Sie ist der Schlüssel zur wahren Beherrschung eines Computers. Hier beginnt ein umfassender Kurs, in dem wir schrittweise die Maschinensprache der Microprozessoren 6502 und Z80 erklären.**

Der Maschinencode kann folgende Formen annehmen:

```
INSTK: SBC $D9FA,X;Outport flag value
```

oder so:

```
DE23 FD FA D9
```

oder aber:

```
11011110 00100011 11111101 11111010
11011001
```

Manchmal auch so:

```
1240 LET ACC=ACC-FLAG(X)
```

oder so:

```
PERFORM FLAG-ADJUST THROUGH LOOP1
```

Es gibt viele Möglichkeiten, dasselbe auszudrücken. Alle haben direkten Einfluß auf die Funktionen des Computers und werden deshalb Maschinensprache oder Maschinencode genannt. Auf der anderen Seite ist es der Maschine gleich, welche Form der Code annimmt – für sie ist er immer eine Folge von elektronischen Impulsen.

## Lösungswege

Mit dem Ausdruck „Maschinencode" ist meistens die Assemblersprache gemeint. Das erste Beispiel stellt eine Befehlszeile für den 6502 dar. Alle anderen Beispiele wurden angeführt, um zu zeigen, daß es keinen Standard der Maschinensprache gibt, sondern nur eine Anzahl unterschiedlicher Methoden, die Folgen von elektrischen Impulsen mehr oder weniger leicht lesbar darzustellen. Wir wollen daher den Maschinencode (oder Assembler – wir gehen im Augenblick noch nicht auf Unterschiede ein) zunächst nur als eine normale Programmiersprache ansehen. Wichtig ist jedoch, daß dabei die Programmierung immer im Vordergrund steht: Ganz gleich, ob Sie in IBM-Assembler oder in Atari-BASIC schreiben, bevor Sie auch nur ein Zeichen eingeben, müssen Sie eine Vorstellung des Lösungsweges haben.

Wozu braucht man den Maschinencode, und was bedeutet der Name? Zunächst: Der Name wurde gewählt, weil dieser Befehlssatz mit den grundlegenden Funktionen der Microprozessoren übereinstimmt. Der Maschinencode wird eingesetzt, wenn die Abläufe im Microprozessor Schritt für Schritt überwacht werden sollen, statt die Steuerung an den Interpreter einer Programmsprache zu übergeben, der sich nur indirekt kontrollieren läßt.

Generell wird im Maschinencode programmiert, wenn höhere Ablaufgeschwindigkeiten notwendig sind. Bei der direkten Arbeit mit dem Prozessor ersparen Sie sich den relativ langsamen Vorgang der Programmübersetzung. Genauer gesagt: Sie sparen Programmablaufzeit, aber die Eingabe, das Testen, die Fehlersuche, Änderungen und die Dokumentation nehmen im Maschinencode weitaus mehr Zeit in Anspruch als bei einer Hochsprache. Die Unbeweglichkeit und Komplexität der Maschinensprache hat nicht umsonst die Entwicklung von Hochsprachen wie COBOL und BASIC beeinflußt.

Wenn der Befehlssatz des Maschinencodes und die Funktionen des Microprozessors übereinstimmen, wie sehen dann diese Abläufe aus und wie arbeitet der Prozessor? In einfachen Worten beschrieben ist die Zentraleinheit eines Computers ein Schalter, der den Datenaustausch zwischen einzelnen Teilen des Systems steuert. Die Komponenten sind der Arbeitsspeicher, die Einheit für Arithmetik und Logik und die Ein- und Ausgabegeräte. Wenn Sie auf der Tastatur ein Zeichen eingeben wollen und eine Taste drücken, erzeugen Sie auf Maschinenebene eine Folge von elektrischen Impulsen. Die CPU leitet diese Folge in einen bestimmten Teil des Speichers, ruft aus einem anderen Speicherbereich eine entsprechende Impulsfolge ab und sendet diese zum Bildschirm, wo wiederum elektronische Impulse das Zeichen darstellen. Der Vorgang mag wie die Bedienung einer Schreibmaschine ausse-

hen. Bei ihr besteht zwischen der Taste und dem Druck des Zeichens eine mechanische Verbindung, während in einem Computer diese Verbindung von der CPU hergestellt wird, die die entsprechenden Impulsfolgen von einem Geräteteil zum anderen leitet. In manchen Fällen erscheint dabei noch nicht einmal ein Zeichen auf dem Bildschirm: Der Tastendruck kann zum Beispiel einen Asteroiden zerstören, ein Programm speichern oder eine Datei löschen, – die ausgeführte Funktion hängt davon ab, wohin die CPU die Impulsfolgen leitet.

In diesem vereinfachten Modell stellt die CPU das Systemherz dar, das alle Informationen (oder elektrische Impulse) passieren müssen, um von einem Teil des Systems in einen anderen gelangen zu können. In Wirklichkeit sieht die Arbeitsweise der CPU und des Systems jedoch wesentlich komplizierter aus. Stellen Sie sich die Zentraleinheit am besten als eine Hauptsteuerung vor, die verschieden gestaffelte Hierarchien von Schaltern betätigen kann, mit denen sie den Fluß der Information indirekt leitet.

## Impulsfolgen

Die Schaltoperationen der CPU lassen sich in bestimmte Bereiche aufteilen, die sich deutlich voneinander unterscheiden. Es gibt arithmetische, logische und speicherbezogene Abläufe wie auch Steuervorgänge. All diese Aktivitäten werden von Impulsfolgen ausgelöst, die auf unterschiedlichen Wegen durch das System geleitet wurden, wobei die CPU selbst die unterschiedlichsten Vorgänge immer auf die gleiche Art zu steuern scheint.

Die Steuerung arithmetischer Abläufe ist mit Abstand die wichtigste Funktion der Maschine. Die CPU kann zwei Zahlen addieren oder subtrahieren, wobei die Subtraktion eine Zahl in ihren negativen Wert verkehrt und dann zu der zweiten Zahl addiert. 7+5=12 bedeutet:

(plus 7) addiert mit (plus 5) ergibt (plus 12).

7–5=2 bedeutet:

(plus 7) addiert mit (minus 5) ergibt (plus 2).

Multiplikation und Division lassen sich als wiederholte Addition und Subtraktion ausdrücken und können dadurch von der Zentraleinheit ebenfalls ausgeführt werden. Mit den vier Grundrechenarten aber kann die CPU jeden mathematischen Vorgang bewältigen, wobei ihr gesamtes mathematisches Potential nur auf der einfachen Fähigkeit beruht, zwei Zahlen addieren zu können.

Logische Vorgänge wollen wir im Augenblick als die Fähigkeit beschreiben, zwei Zahlen miteinander vergleichen zu können. Damit sind aber nicht nur einfache Größenvergleiche gemeint, sondern auch die Untersuchung der Zahlenmuster einzelner Stellen. Die Tatsache, daß sieben größer als fünf ist, läßt sich leicht feststellen, wenn von sieben fünf abgezogen wird und das Ergebnis positiv ist. Die CPU verfügt über diese Fähigkeit, kann aber auch z. B. die Zahlen 189 und 102 miteinander vergleichen und feststellen, daß beide Zahlen in den Hundertern die gleiche Ziffer haben.

Die CPU kann grundsätzlich zwei Speichervorgänge ausführen: Sie kann die Informationen einer Speicherstelle in ihren internen Speicher kopieren und von diesem an eine andere Speicherstelle setzen. Werden beide Vorgänge nacheinander ausgeführt, können damit Informationen aus jedem Speicherbereich in jeden anderen Speicherbereich kopiert werden. Mit diesen beiden Fähigkeiten läßt sich das gesamte Speichermanagement vollständig bewältigen.

Steuervorgänge sind Entscheidungen der CPU, in welcher Reihenfolge sie die oben beschriebenen Vorgänge ausführt. Die CPU kann mathematische Berechnungen ausführen, Zahlen miteinander vergleichen, Informationen im Speicher bewegen und Entscheidungen über die internen Funktionsabläufe fällen. Beherrscht eine Zentraleinheit diese vier Abläufe dann kann sie alle nur denkbaren Aufgaben ausführen, die für Computer geeignet sind. Die Abläufe müssen dafür nur in die richtige Reihenfolge gestellt werden, wobei die Reihenfolge durch das Programm definiert wird, das für die Lösung einer speziellen Aufgabe analysiert wurde.

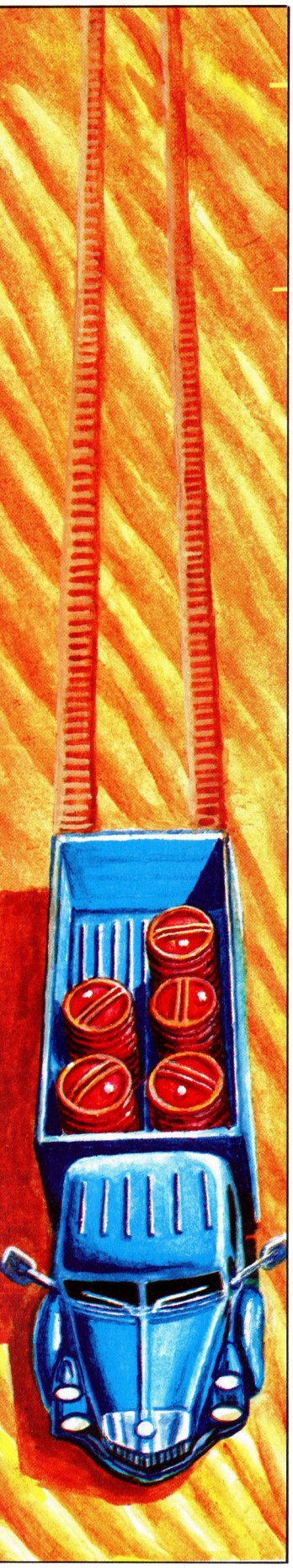
Diese vier Funktionsarten reichen für die Beschreibung eines Computerkonzeptes aus.

## Funktionsblöcke

Sehen wir uns einmal die Funktionsblöcke eines Programms in BASIC an. Welche Teile kann man darin unterscheiden? Jedes Programm hat Variablen, die nichts anderes sind als die Namen der Speicherbereiche, in denen bestimmte Informationen abgelegt sind. Die meisten Programme führen mit diesen Variablen mathematische Operationen durch. Nach der Berechnung vergleicht ein Programm in vielen Fällen zwei Informationen und entscheidet sich auf der Grundlage des Ergebnisses, ob es eine bestimmte Gruppe von Befehlen ausführt. Informationen erhält ein Programm üblicherweise über die Tastatur.

Abgesehen von der Ein- und Ausgabe enthält diese Beschreibung nicht mehr als die vier Grundfunktionen der CPU – nur in anderen Worten beschrieben. Und wenn Sie zunächst akzeptieren, daß die CPU Ein- und Ausgabegeräte nur als besondere Speicherbereiche ansieht, dann ist diese Beschreibung der Funktionsabläufe sogar komplett. Die Ausführung eines Programms läßt sich als Steuerung eines Informationsflusses beschreiben.

**Dieser Beitrag über die internen Funktionsabläufe wird im nächsten Heft fortgesetzt.**



# Durch die Wüste

**Computerspiele müssen nicht immer nur aus dem Schießen auf Feinde bestehen. Manchmal ist das logische Denken wichtiger.**

Im hier vorgestellten Spiel wird das Fahren mit einem Lastwagen durch die Wüste dadurch erschwert, daß der Spieler nicht genügend Benzin für eine Non-Stop-Tour mitnehmen kann. Das Spiel findet in einer 1000 Quadratkilometer großen Wüste statt. Etwa alle 100 Kilometer gibt es ein Lager, in dem Benzinkanister untergebracht werden können. Im Basislager befinden sich genügend Kanister, von denen jeder für die Strecke von einem Stützpunkt zum nächsten ausreicht. Soweit ganz einfach, der Lastwagen kann aber nur acht Kanister gleichzeitig aufladen. Um die Strecke zu schaffen, müssen Sie also hin- und herfahren und Benzindepots anlegen.

## Veränderte Variable

Das Wichtigste ist natürlich, daß man nie mit leerem Tank in der Einöde steht – das Basislager ist weit, und in der Wüste ist es alles andere als gemütlich. Außerdem müssen Sie versuchen, mit möglichst wenig Benzin und einer kurzen Fahrstrecke auszukommen. Mit einer Beladungskapazität von acht Kanistern ist das noch relativ einfach.

Sie können die Fahrt aber auch schwieriger gestalten: Wie läßt sich die Fahrt an, wenn nur vier oder sechs Kanister aufgeladen werden dürfen? Um das herauszufinden, wird die Variable M in Zeile 60 verändert. Zwar bleibt die grundsätzliche Strategie dabei gleich, aber Sie müssen öfter hin- und herfahren, und die Abstände zwischen den Benzinlagern werden verändert. Können Sie einen Lösungsweg finden, der Sie in jedem Fall sicher aus der Wüste herausführt? Ein Algorithmus könnte die Basis für ein Programm sein, das dieses Problem löst.

Anhand des Programms werden die grundlegenden Techniken aufgezeigt, die man für die spezifische Problemlösung braucht: Zuerst wird mit der gegebenen Information experimentiert, Beispiele werden ausprobiert. Dabei ergeben sich die festen Regeln. Aus diesem wiederum entwickelt sich der Algorithmus, den man zur Erstellung des Programms braucht.

Natürlich läßt sich das „Wüstenspiel" auch noch verfeinern: Entwickeln Sie eine passende Grafik, oder bauen Sie weitere Schwierigkeiten ein: So könnten zum Beispiel Bedingungen wie die Mitnahme von nötigem Werkzeug, Trinkwasser und Lebensmitteln eingegeben werden. Die Programmzeilen dienen also auch als Grundlage für ein komplexeres Spiel.

## BASIC-Dialekte

Das Programm ist in Microsoft-BASIC geschrieben. Bei Computern, die mit einem anderen BASIC arbeiten, sind entsprechende Änderungen einzufügen. Beim Spectrum muß vor jeder Variablen-Definition LET eingegeben werden.

CHR$(26): Beim Spectrum, Oric-1, Atmos, Dragon und Acorn B durch CLS ersetzen. Beim Commodore 64 und VC 20 CHR$(147) verwenden.

MID$(STR$(A(1),2)): Beim Spectrum und allen anderen Rechnern, bei denen der Befehl PRINT LEN(STR$(2)) eine 1 ergibt, durch STR$(A(1)) ersetzen.

THEN 1260 & THEN 1300: Beim Spectrum durch THEN GOTO 1260 & THEN GOTO 1300 ersetzen.

```
10 REM****DURCH DIE WUESTE****
30 DIM A(10)
40 A(1)=80 REM KANISTER BEIM START
50 S=1 REM POSITION DES LASTWAGENS
60 M=8: REM MAX. ANZ. DER KANISTER
70 N=0 REM ANZ. DER BENOETIGTEN KANISTER
100 REM NEUE RUNDE
110 PRINT CHR$(26): REM SCHIRM LOESCHEN
120 PRINT "LAGER  1 2 3 4 5 6 7 8 9 10"
130 PRINT "KANISTER  ";
140 FOR I=1 TO 10
145 A$=MID$(STR$(A(I)),2)
147 IF LEN(A$)<2 THEN LET A$=" "+A$ GOTO 147
150 PRINT A$;" ";: NEXT I: PRINT
160 X=9+(S-1)*3: PRINT TAB (X),"T" PRINT TAB (X)," "
    PRINT TAB(X-2), "LASTWAGEN" PRINT TAB (X-1),T
175 IF S=10 THEN PRINT: PRINT "GESCHAFFT!!!" GOTO1400
180 IF T=0 AND A(S)=0 THEN PRINT  PRINT "SIE MUESSEN
    LEIDER LAUFEN" GOTO 1400
190 PRINT PRINT "IHRE MOEGLICHKEITEN ": PRINT
200 IF T>1 THEN PRINT "A KANISTER ABLADEN"
210 IF A(S)>0 THEN PRINT "Z KANISTER ZULADEN"
220 IF T>0 THEN PRINT "F FOLGENDES LAGER"
230 IF T>0 AND S>1 THEN PRINT "L LETZTES LAGER"
240 PRINT: PRINT "IHRE WAHL"; : INPUT A$
250 IF T>1 AND (A$="A" OR A$="a") THEN 1260
260 IF A(S)>0 AND (A$="Z" OR A$="z") THEN 1300
270 IF T>0 AND (A$="F" OR A$="f") THEN
    S=S+1: T=T-1: N=N+1: GOTO 110
280 IF T>0 AND S>1 AND (A$="L" OR A$="1") THEN
    S=S-1: T=T-1: N=N+1: GOTO 110
290 GOTO 110
1250 REM KANISTER ABLADEN
1260 PRINT: INPUT "WIEVIELE ";A
1270 IF A>T OR A<>INT(A) OR A<0 THEN PRINT: PRINT
    "VERSUCHEN SIE ES NOCH EINMAL": GOTO 1260
1280 A(S)=A(S)+A: T=T-A: GOTO 110
1290 REM KANISTER AUFLADEN
1300 PRINT: INPUT "WIEVIELE "; A
1310 IF A>A(S) OR A<>INT(A) OR A<0 THEN PRINT: PRINT
    "VERSUCHEN SIE ES NOCH EINMAL": GOTO 1300
1320 IF A+T>M THEN PRINT: PRINT "ES IST NUR PLATZ
    FUER ";M;" KANISTER" GOTO 1300
1330 T=T+A: A(S)=A(S)-A
1380 GOTO 110
1390 REM SPEILENDE
1400 PRINT: PRINT "SIE HABEN ";N; " KANISTER GEBRAUCHT. ",
1410 PRINT "BRACHTEN ";T
1420 A=0: FOR I=2 TO 9: A=A+A(I): NEXT I
1430 PRINT "MIT UND HABEN ";A; "IN DER WUESTE GELASSEN"
1440 PRINT: PRINT "RUN FUER NEUEN START EINGEBEN"
1450 END
```

# Fachwörter von A bis Z

**ASCII = ASCII**
Der „American Standard Code for Information Interchange" ist von den meisten Hard- und Softwareanbietern für die Binär-Darstellung alphanumerischer Zeichen übernommen worden. Es gibt zwar keinen besonderen Grund, den Buchstaben A ausgerechnet als „0100 0001" (8-Bit-ASCII-Norm) zu codieren, aber die Vereinbarung einer Norm bietet den Vorteil, daß Programme auf unterschiedlichen Maschinen verwendbar werden. So wird eine problemlose Datenübertragung von einem zum anderen Rechner möglich. Kommerzielle Software-Pakete für Kleinrechner bieten oft die Möglichkeit, ASCII-Dateien zu erzeugen, deren Sinn am besten ein Beispiel erläutert: Bei der Textverarbeitung enthalten die abgespeicherten Schriftsatz-Dateien meist eine Anzahl spezieller Steuerzeichen, etwa um die Zentrierung der Überschrift auf einer Seite zu kennzeichnen. Macht der Rechner daraus eine ASCII-Datei, so setzt er überall statt der Steuerzeichen ASCII-Symbole ein, hier also entsprechend viele Leerzeichen. Die ASCII-Datei kann daher auch von einem zweiten Textverarbeitungssystem mit einem anderen Steuerzeichen-Satz gelesen werden.

**Assembly Language = Assemblersprache**
Der Assemblercode ist im wesentlichen nur eine lesbarere Schreibweise der Maschinensprache, mit „symbolischen" (alphanumerischen) anstelle hexadezimaler Adressen und mnemotechnischem Befehlscode statt eines Zahlenschlüssels. Im Gegensatz zu anderen höheren Sprachen werden die Daten bei der Assembler-Übersetzung Byte für Byte in den Maschinencode übertragen. Daher sind Assemblerprogramme ebenso effizient wie Maschinenprogramme. Das vom Benutzer geschriebene Quellprogramm, auch Source Code genannt, wird durch einen Übersetzer (der selbst „Assembler" heißt) im Rechner in ein Objektprogramm im Maschinencode umgewandelt.

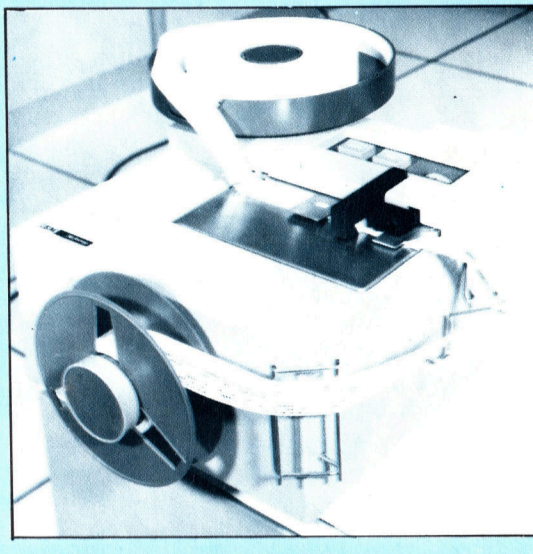
**Hier werden einzelne Fachausdrücke eingehend behandelt. Da bei der Kommunikation mit dem Computer meist die englische Sprache verwendet wird, werden hier zunächst die englischen Begriffe genannt, dann die deutsche Übersetzung. In den Gesamtindex werden sowohl deutsche als auch englische Stichwörter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

**Asynchronous = Asynchron**
Daten können synchron oder asynchron übertragen werden. Meist werden diese Begriffe verwendet, wenn es um die serielle Datenfernübertragung geht, zum Beispiel bei Telefonleitungen. Die asynchrone Übertragung ist einfacher durchführbar und über die gängigen Schnittstellen, wie RS 232, zu bewerkstelligen. Die synchrone Übertragung erlaubt jedoch höhere Datenraten und wird daher bei allen großen Rechnersystemen eingesetzt.

Beim Asynchron-Verfahren beginnt die Übertragung, wann immer das Sendegerät Daten ausgeben will. Jedem Byte wird ein zusätzliches „Start-Bit" vorangestellt, mit dem das empfangende Gerät darauf hingewiesen wird, daß gleich acht Informationsbits eintreffen, und zwar mit der Baudrate, auf die Sender und Empfänger eingestellt sind. Mit dem „Stop-Bit" wird die Übertragung des Bytes abgeschlossen. In dieser Abfolge wird ein Byte nach dem anderen übertragen.

Beim Synchronverfahren haben Sender und Empfänger präzise synchronisierte Taktgeber, und die Übertragung erfolgt in einem festgelegten Rhythmus. Es ist etwa so, als ob Sie sagen: „Ich warte zu jeder vollen Stunde fünf Minuten lang am Telefon auf deinen Anruf." – wobei der Rechner allerdings nur Mikrosekunden warten würde. Das Synchronverfahren benötigt keine Start- und Stop-Bits und arbeitet entsprechend schneller.

**Attribute = Attribut**
Die ersten Computer waren nur in der Lage, normale Zeichen darzustellen und allenfalls zwischen Klein- und Großbuchstaben zu unterscheiden. Heute können Sie schon bei einfachen Heimcomputern für jeden Buchstaben einzelne Farben festlegen, dazu noch die des Hintergrundes, außerdem Fett- oder Negativschrift wählen. Diese Zusatzangaben heißen „Attribute". Meist ist für jede Zeichenposition des Schirms ein Byte im RAM vorgesehen. Das Setzen der Attribute erfolgt entweder durch BASIC-Befehle wie BRIGHT, FLASH, INK, PAPER usw. oder durch unmittelbare Bitmanipulation im Maschinencode über AND/OR-Verknüpfungen.



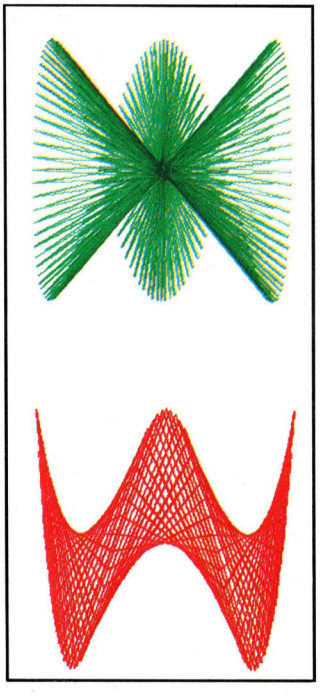
**Das System der Verschlüsselung läßt sich anhand eines Lochstreifengerätes veranschaulichen: Jedes ASCII-Zeichen wird durch eine Reihe von maximal sieben oder acht Löchern dargestellt; zwischen der dritten und der vierten Lochposition ist noch ein kleines Führungsloch für den Streifentransport.**

**Bildnachweis**
477: The Kobal Collection
480: Ian McKinnell, Soft
481: Kevin Jones, Your Spectrum
482, 483: Kevin Jones
484: Liz Dixon, Ian McKinnell
485, 500, 501: Ian McKinnell
490, 491, 495: Tony Lodge
499: Alan Adler
504: Adrian Morgan
U3: IBM

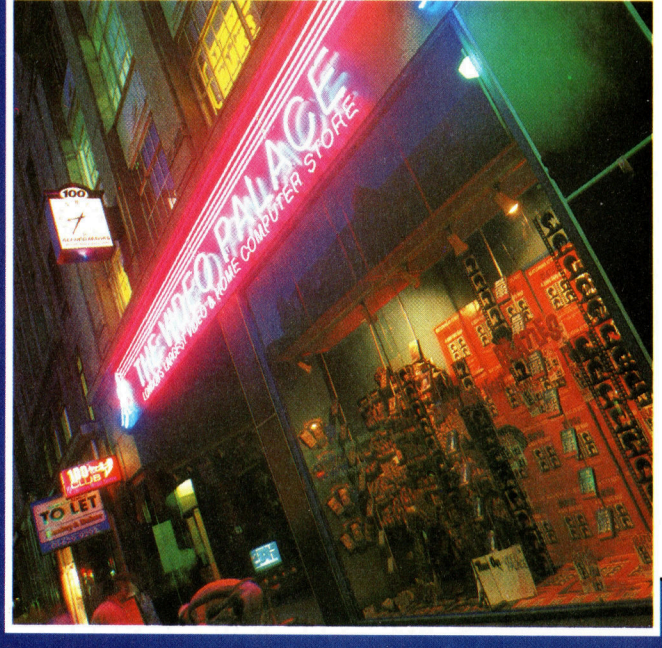
++ Vorschau +++ Vorschau +++ Vorschau +++

# computer kurs Heft 19


**Der IBM PC**
ist der Personal-Computer des größten Rechnerherstellers. Ein verläßliches Gerät, das durch Steckkarten vielfältig erweiterbar ist.



**LOGO: Zufallszahlen**
Dieser Teil unseres LOGO-Kurses zeigt, welche Möglichkeiten diese Sprache für die Bearbeitung numerischer Werte zu bieten hat.




**Programm-Verkauf**
Heimcomputer-Besitzer haben durch den Verkauf eigener Programme Geld verdient.

**+++ Das Büro im Computer +++ BASIC +++ Tips für die Praxis +++ Acorn Electron +++ Maschinencode +++ Vielseitige Printer +++ Fragen und Antworten +++ Die Welt der Roboter +++ Bill Gates**