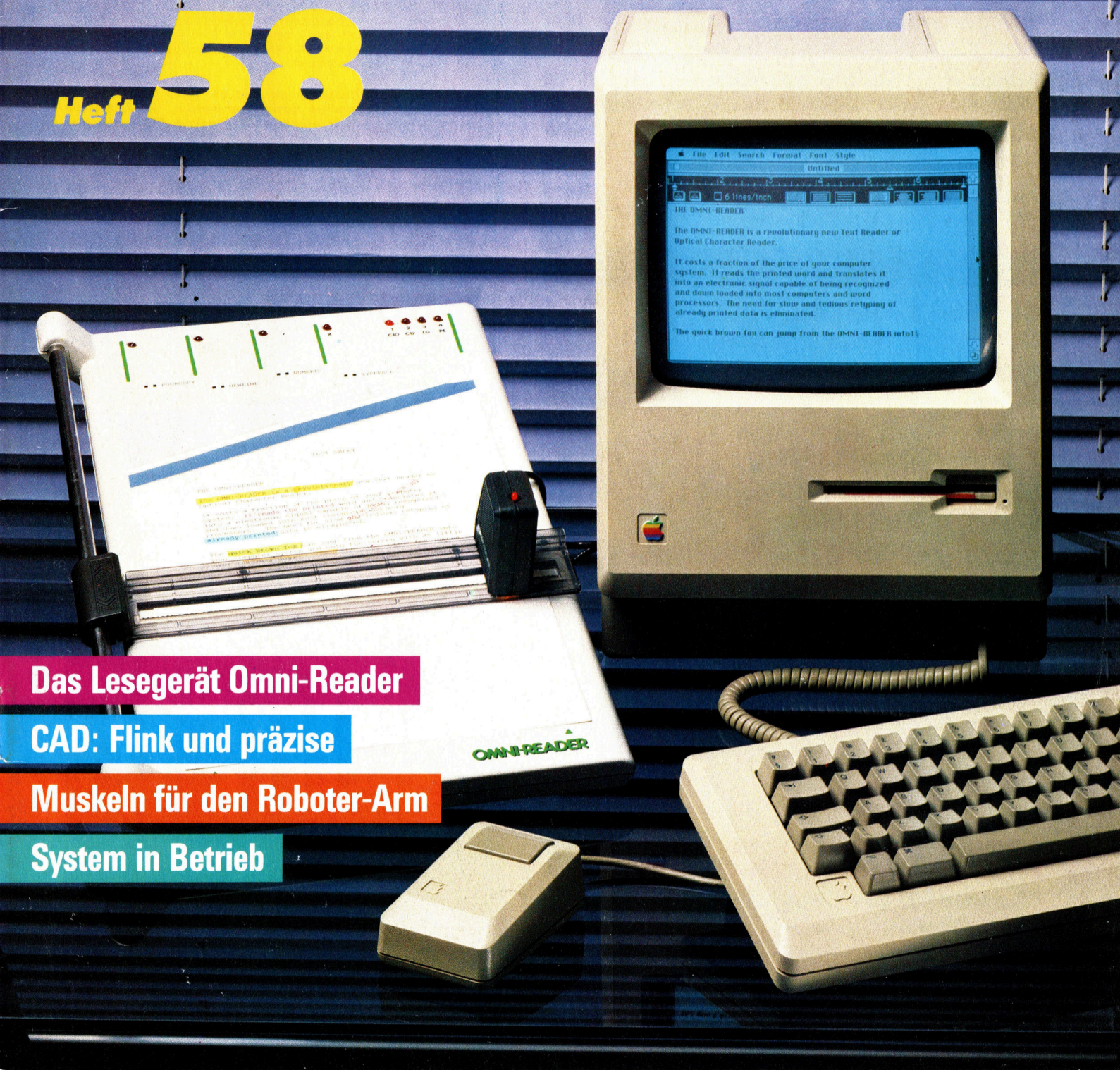
Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Ein wöchentliches Sammelwerk

Heft 58



Das Lesegerät Omni-Reader

CAD: Flink und präzise

Muskeln für den Roboter-Arm

System in Betrieb

computer kurs Heft 58

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

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

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

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

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

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

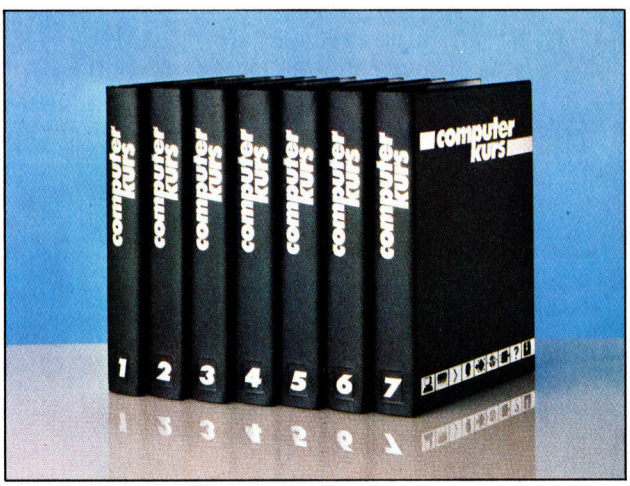
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Peter Aldick, Holger Neuhaus, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

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



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

Flink und präzise

Mit CAD, computergestütztem Zeichnen, können Produkte nicht nur entworfen, sondern bereits vor der Produktion auf ihre künftigen Eigenschaften hin untersucht werden.

CAD kann – im Unterschied zur gewöhnlichen Computergrafik – erheblich mehr, als nur beim Zeichnen zu helfen. Hochentwickelte CAD-Systeme „kennen" das Objekt in ihrem Speicher nicht allein dem Aussehen nach. Diese Systeme können das Objekt testen, Komponenten zusammenfügen und auch Belastungen und Abnutzungsstellen berechnen. Die Funktionssicherheit eines Werkstükkes kann damit bereits vor der Nullserie gründlich geprüft werden.

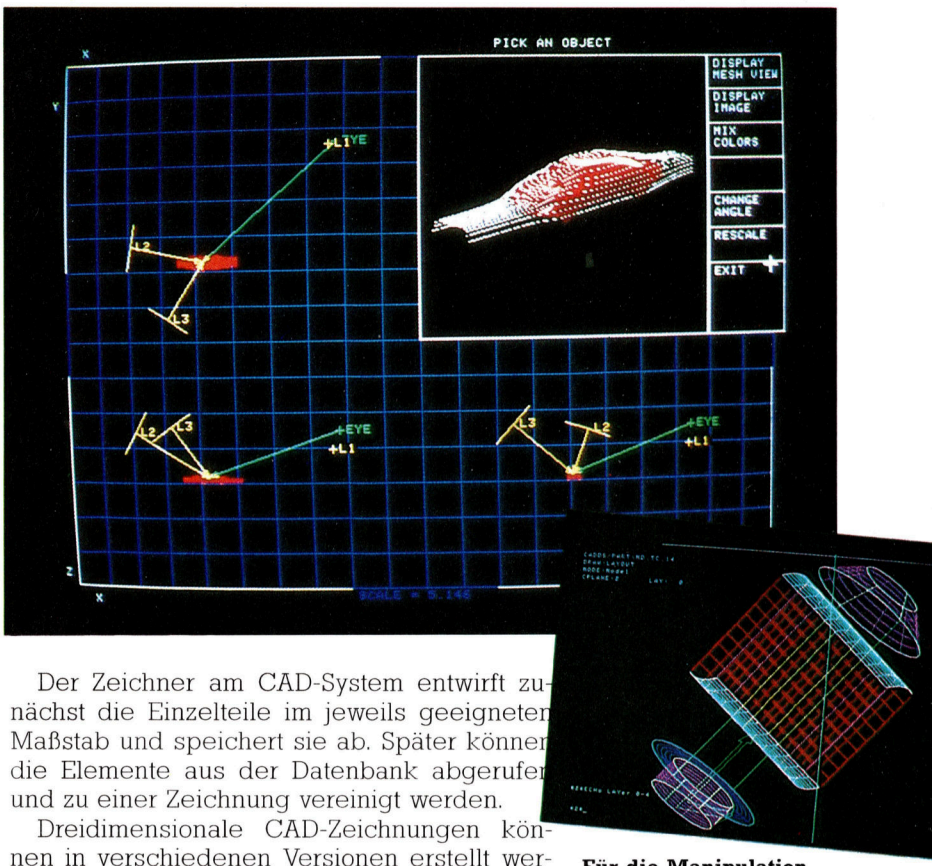
Bilder werden in CAD-Systemen anders als bei der üblichen Computergrafik erzeugt. Im konventionellen Verfahren werden Bilder aus Einzelpunkten (Pixeln) aufgebaut – nicht exakt genug für CAD, das mit einem vektoriellen Grafiksystem arbeitet. Das Bild wird dabei in mathematische Formeln umgewandelt, die bestimmte Bildinhalte verkörpern. Ein Strich in einer Pixelgrafik besteht aus einer Reihe von Punkten, bei CAD ist der Strich als kürzeste Verbindung zweier Punkte definiert. Vektorgrafik ist exakter als Pixelgrafik – ihre Auflösung kann das Darstellungsvermögen auch der besten Monitore übertreffen.

Einfache CAD-Systeme arbeiten nur zweidimensional – gelegentlich kann eine dritte Dimension zumindest in der Darstellung angedeutet sein – eine sogenannte $2\frac{1}{2}$-D-Zeichnung. Für einen Architekten reichen diese Möglichkeiten aus. Mit der zusätzlichen „halben" Dimension kann er sowohl Grundrisse als auch Seitenansichten auf dem Bildschirm darstellen.

3-D-Fähigkeiten

Zum Entwurf elektronischer Schaltungen werden heute ebenfalls zweidimensionale CAD-Systeme verwendet, die allerdings komplizierter aufgebaut sind: Sie speichern nicht nur das Bild allein, sondern auch Informationen über die Bildelemente. In der Elektronik sind CAD-Systeme meist mit Testprogrammen gekoppelt – nach dem Entwurf der Millionen von Schaltungen für einen Microchip simuliert und prüft das Testprogramm dessen Funktion.

Sehr komplex sind die Aufgaben des CAD beim Entwurf von Maschinenteilen. Dazu sind 3-D-Fähigkeiten, größere Speicher und eine höhere Verarbeitungsgeschwindigkeit nötig. Da die Zeichnungen direkt im Produktionsprozeß verwendet werden, müssen sie von höchster Genauigkeit sein.



Der Zeichner am CAD-System entwirft zunächst die Einzelteile im jeweils geeigneten Maßstab und speichert sie ab. Später können die Elemente aus der Datenbank abgerufen und zu einer Zeichnung vereinigt werden.

Dreidimensionale CAD-Zeichnungen können in verschiedenen Versionen erstellt werden. Am einfachsten sind Drahtmodelle, die sich zwar über alle Dimensionen erstrecken, in denen nur die Begrenzungslinien, nicht aber die Flächen dargestellt werden. Bei voller 3-D-Darstellung erscheinen Gegenstände fast naturgetreu, einschließlich aller Flächen.

Die besten CAD-Systeme erstellen sogar Zeichnungen mit bis zu 256 verschiedenen Farben. Auch Licht- und Schattenwirkungen werden in der Darstellung berücksichtigt und unterstützen so die räumliche Imagination.

Neben dem reinen Zeichnen steht dem CAD-Anwender eine Vielfalt von Funktionen zur Verfügung. Dazu gehören:

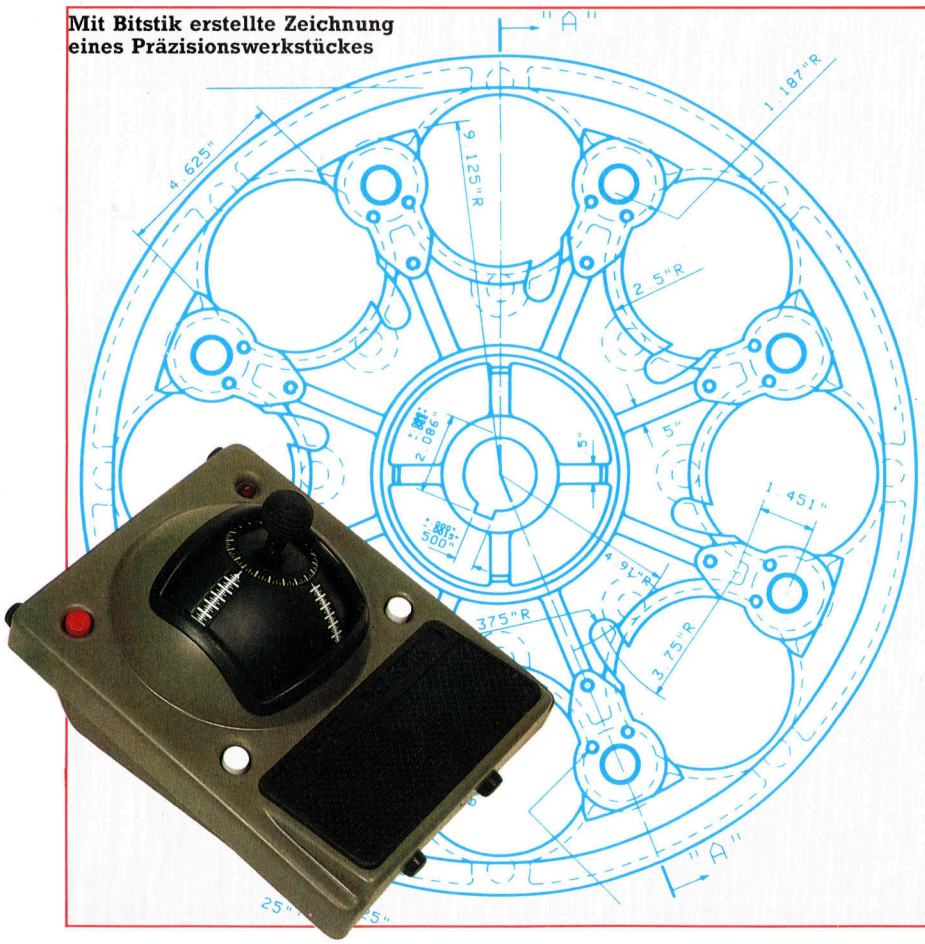
- **Bemaßung:** Eine Zeichnung kann nach der Fertigstellung automatisch mit allen notwendigen Maßangaben versehen werden. Das ist relativ unkompliziert, weil das System alle Werte bereits in numerischer Form gespeichert hat.
- **Schraffieren:** Teile eines Entwurfs können dadurch hervorgehoben werden. Bei CAD werden Schraffierungen auf Wunsch automatisch im entsprechend benötigten Rahmen und Bereich erzeugt.

Für die Manipulation von Bildern und Zeichnungen mit hoher Auflösung wird beim CAD die Vektorgrafik-Technologie eingesetzt. Das Bild oben zeigt links die CAD-Darstellung einer Schweißanordnung. Mit hochentwikkelten Systemen können die spezifischen Eigenschaften eines Werkstückes bereits am Bildschirm geprüft werden. Das obere Bild zeigt einen Cray-Supercomputer bei Aerodynamik-Tests an einem Automodell von General Motors.

● **Schichten:** Zeichnungen lassen sich aus einzelnen Schichten aufbauen. So kann etwa ein Architekt den Grundriß eines Gebäudes mit einer zweiten Zeichnung unterlegen, die nur die Zu- und Abflußleitungen darstellt. Beim Ausdruck können die Pläne dann sowohl einzeln als auch gemeinsam dargestellt werden.
● **Segmentierung:** Entwürfe werden dabei aus einzelnen Elementen zusammengestellt, die sich in einer Datenbank befinden und bei Bedarf aufgerufen werden können.
● **„Gummi-Modelle":** Zeichnungslinien erscheinen elastisch und können beim Entwurf wie ein Gummiband an den gewünschten Platz gezogen werden.
● **Skalierung:** Der Zeichner kann in jedem gewünschten Maßstab arbeiten, die Informationen speichern und später mit dem restlichen Entwurf zusammen in einem gemeinsamen Maßstab darstellen lassen.
● **Zoom:** Arbeitet wie in der Fotografie – ein bestimmter Teil des Entwurfs kann beliebig vergrößert bzw. verkleinert und dann bearbeitet werden.

Je nach der spezifischen Aufgabe gibt es verschiedene Eingabegeräte für CAD-Systeme. Am häufigsten werden Digitalisiertabletts, Lichtgriffel, Trackballs und die Maus verwendet. Bildprozessoren werden bei extrem komplizierten Zeichnungen und Fotografien eingesetzt: Die Vorlage wird fotografiert und vom Prozessor in einzelne Bildelemente zerlegt. Natürlich nutzt CAD auch die Tastatur – sie dient zur exakten Eingabe der Koordinaten einer Vektorgrafik.

Auch Joysticks können für CAD verwendet werden – sie müssen allerdings genauer arbeiten als die für Computerspiele benutzten Typen. Der „Bitstik" von der Firma Robocom etwa ist mit zwei Präzisionspotentiometern ausgestattet, die Bewegungen des Steuerknüppels auf sechs Dezimalstellen angeben. Der Joystick steuert den Cursor auf dem Bildschirm, und die Tasten rufen Menüfunktionen ab.



Mit Bitstik erstellte Zeichnung eines Präzisionswerkstückes

CAD mit dem Heimcomputer

CAD verbreitet sich zunehmend – kein Wunder, denn CAD ist ein sehr zuverlässiges und kostengünstiges Hilfsmittel sowohl für den Entwurf als auch für die Simulation und Prüfung von Werkstücken. Professionelle CAD-Systeme benötigen eine relativ aufwendige Geräteausstattung – dennoch gibt es CAD-Programme, die auf normalen Heimcomputern laufen.

Das HRX-System für den Memotech 512 kann die von einer Videokamera gelieferten Bilder mit dem Computer weiterverarbeiten. Einzelbilder werden in den Rechner gelesen und können in mehreren Stufen verändert werden. So läßt sich das Bild vergrößern, verkleinern oder drehen. Auch die verwendeten Farben sind beliebig veränderbar. Das HRX-Farbsystem arbeitet mit den Video-Grundfarben rot, grün und blau. Durch Kombination sind bis zu 16,5 Millionen Farbtöne möglich.

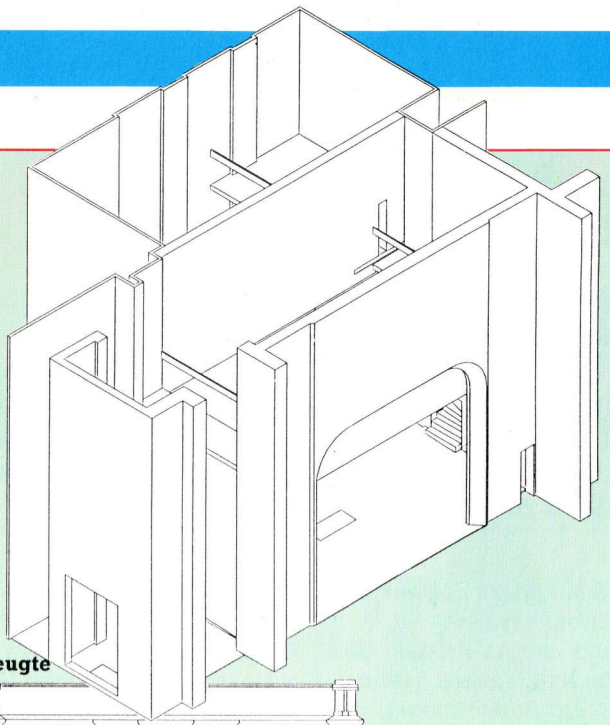
Diese enorme Informationsmenge erfordert viel Speicherplatz – zusätzliche Steuer- und Speicherkarten sind notwendig, bei Farbdarstellung sogar in mehrfacher Ausführung. Außerdem wird ein besonderer „flash"-A/D-Konverter (1-kanalig bei monochromer, 3-kanalig bei farbiger Darstellung) benötigt, der die Informationen aus dem MTX-Computer übernimmt. Das voll aufgerüstete System kann die Daten von der Videokamera mit 7,2 MByte pro Sekunde übernehmen. Allerdings haben Leistungen auch ihren Preis – das HRX-System kostet einige Tausender.

Etwas preisgünstiger ist das Bitstik-System für den Acorn B. Damit lassen sich technische Zeichnungen in professioneller Qualität erzeugen. Es umfaßt einen besonderen Präzisions-Joystick, die Systemdiskette, ein Acht-KByte-ROM und eine Diskette mit mehreren Normschriften und Symbolen. Allerdings muß der Acorn B mit einem 6502-Coprozessor und einem Doppellaufwerk ausgestattet sein. Um das System sinnvoll zu nutzen, kommen auch noch Farbmonitor und Farbplotter dazu.

Der Hersteller des Systems behauptet, daß auch ein Anwender ohne künstlerische Ambitionen damit professionelle Entwürfe anfertigen kann. Die Basis bilden die auf der Diskette gespeicherten Bildelemente und die im ROM vorhandenen Grundformen wie Kurven und Linien, die Zeichnung wird im Baukastensystem zusammengesetzt.

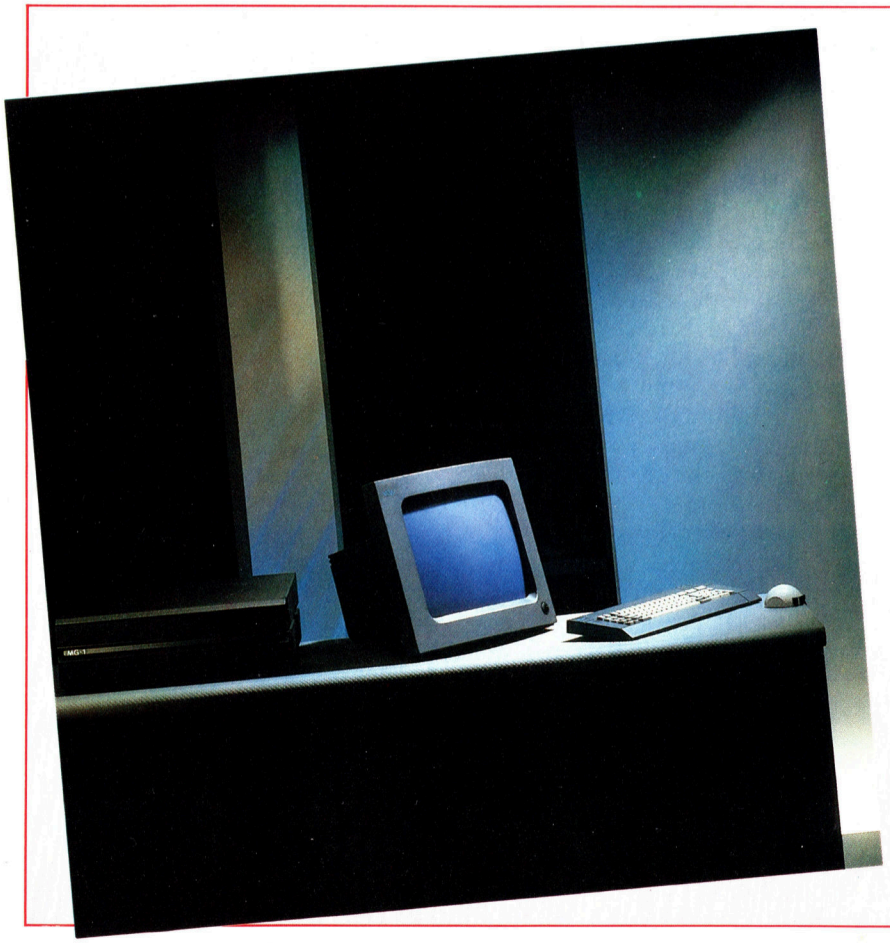
Gute Sicht auf jeder Bühne

Ein „Matchbox"-Bühnenbild-Entwurf



Mit „Matchbox" erzeugte architektonische Darstellung

Computer Aided Design wird meist mit der industriellen Produktion, etwa dem Entwurf neuer Automodelle, in Verbindung gebracht. Die Technik läßt sich jedoch an fast alle Bereiche anpassen, in denen Modelle eine Rolle spielen – auch an die Erfordernisse des Theaters: Bühnenbildnern bereitet es oft große Schwierigkeiten, ein Szenenbild so aufzubauen, daß die Sicht von allen Plätzen gleich gut ist. Soll ein Stück in verschiedenen Theatern laufen, wird es noch komplizierter – in jedem Schauspielhaus herrschen andere Sichtverhältnisse. Früher wurden maßstabgetreue Modelle der Szenerie gebaut, die auf einer ebenfalls verkleinerten Bühne so lange verschoben wurden, bis der optimale Platz gefunden war. Die Zusammenarbeit eines Theater-Teams mit der Software-Firma Robary Ltd. führte auf einen neuen Weg: Mit dem „Matchbox"-System kann das Bühnenbild am Computer erzeugt und aus jedem beliebigen Blickwinkel betrachtet werden. Beim Entwurf kann der Bühnenbildner seine Szenerie in jedes Theater versetzen und es von allen möglichen Standorten aus betrachten. Wenn Sichtprobleme auftreten, läßt sich das Bühnenbild rechtzeitig abwandeln. Die Requisiten werden dazu auf dem Bildschirm so lange umgestellt, bis die Sicht von jedem Platz gut ist. Matchbox erstellt auch exakte Pläne, nach denen Bühnentischler und Theatermaler arbeiten können.



CAD-Hardware

Nicht nur die Software wird speziell für CAD-Anwendungen entwickelt, es gibt auch besonders für diesen Einsatz geeignete Computer. Die „MG-1-Workstation" ist ein Rechner, der optimal auf CAD-Anwendung zugeschnitten ist. Der MG-1 arbeitet mit dem schnellen 16-Bit-Prozessor NS 32016 von National Semiconductor. Die Grafikauflösung beträgt 1024 × 800 Pixel, wobei Spezialausstattungen für ein flimmerfreies Bild sorgen. Dazu gehört ein direkter Speicher-Refresh, der Veränderungen in der Bildschirmdarstellung ohne Zwischenschritte möglich macht. Probleme mit ungenauer Kontrolle durch die Maus werden durch einen Extra-Prozessor für die Maus- und Tastatursteuerung vermieden.

Der MG-1 läuft unter dem Betriebssystem GENIX, einer National-Semiconductor-Version des verbreiteten Unix-Multitasking-Systems. Für den Rechner sind bereits mehrere Anwenderprogramme entwickelt worden. So hat etwa die englische Firma Payfec das professionelle Zeichenprogramm DOGS erstellt. Von Lattice Logic stammt ein Programm zur Entwicklung von Microprozessoren mit dem bezeichnenden Namen „Chipsmith" – der Chip-Schmied. Es ist ein interessantes Beispiel, wie Computer zur Weiterentwicklung der Computertechnologie eingesetzt werden.

MicroPen-Programm

An den Beispielen Archive, Card Box und dBase II hatten wir die Grundlagen der Datenbankverwaltung erläutert. In diesem Artikel untersuchen wir nun das auf den Schneider-Computer ausgerichtete, preisgünstige Datensystem MicroPen.

MicroPen wird von Schneiders Softwareabteilung als „Datenbanksystem für den CPC 464" angeboten und ist das Produkt des irischen Softwarehauses Intelligence Treland. MicroPen ist Teil einer Programmtrilogie, die eine Datenbank, eine Textverarbeitung und ein Kalkulationssystem umfaßt. Der Aufbau dieses Paketes hat Ähnlichkeiten mit Lotus 1–2–3. Die Datenbankverwaltung (DBV) enthält auch das Textsystem Penform, mit dem die Datensätze der DBV-Dateien auf dem Bildschirm angelegt werden. Die DBV selbst hat den Namen PEN.

Da die Schneider-Version des MicroPen unter CP/M-80 läuft, wird mindestens ein Diskettenlaufwerk benötigt. Das System verteuert sich dadurch zwar zusätzlich, gewinnt aber wesentlich an Geschwindigkeit und Kapazität.

Zur Anlage einer Datenbankdatei wird zunächst das Datensatzformat über Penform eingegeben. Obwohl vom Handbuch als Textverarbeitung bezeichnet, ist Penform jedoch kaum mehr als ein Bildschirmeditor: Die Zeichen werden per Bildschirm eingegeben und verändert, und der Cursor wird mit den Steuertasten bewegt. Einfache Editierbefehle wie „CTRL Y" zum Löschen einer Zeile oder „ESC E" zum Speichern auf Diskette unterstützen zwar die Cursortasten, doch lassen sich keine längeren Schriftstücke anlegen oder editieren.

Die Dateien von MicroPen können nur begrenzte Informationsmengen aufnehmen. So kann ein Datensatz (im Handbuch „Layout" genannt) bis zu 100 Felder enthalten, darf aber nicht länger als 1024 Zeichen lang sein. Und obwohl Dateien mit 32 750 Datensätzen möglich sind, sind Datenbanken dieser Größe auf einem Diskettensystem nicht realisierbar. Bei voller Auslegung der 32 750 Datensätze – jeder mit der maximalen Länge von 1024 Zeichen – würden allein die Rohdaten (ungepackt) 33 Megabyte belegen.

Jedes Datensatzfeld muß auf dem Bildschirm mit einer eindeutigen Kennung – der „Feldmarkierung" – versehen werden. Die Kennung kann jedes darstellbare Zeichen sein (außer der eckigen Klammer, die die Rolle des Feldbegrenzers übernimmt).

Wenn wir MicroPen als DBV für unsere Datei LAGER einsetzen, sieht das ein mit Penform entworfene „Layout" folgendermaßen aus:

```
Nummer des Herstellers: [A ]
Unsere Lagernummer: [B                ]
Preis: [C ] Lagermenge: [D            ]
Beschreibung: [E                      ]
Hersteller: [F                        ]
Telefon des Herstellers: [G           ]
Ansprechpartner: [H                   ]
```

Nach Erstellung dieser Maske wird die Struktur mit dem Dateinamen LAGER.INP auf der Diskette abgelegt (die Angabe .INP als Erweiterung des Dateinamens wird von dem MicroPen DBV-Programm benötigt).

Nach dem Aufruf zeigt Pen einen Prompt auf dem Bildschirm an und fragt nach einem Dateinamen. Für unsere Lagerdatei geben wir LAGER ein (hier ist kein .INP nötig). Pen zeigt nun sein „Hauptmenü" mit fünf Bearbeitungsmöglichkeiten:

```
Datenbank: LAGER, pro Satz 328, gespei-
cherte Sätze 0
0=Ende, 1=Eingabe, 2=Laden, 3=Laden
&Drucken, 4=Index, 5=Organisieren
```

Statuszeile

Über die Option 1 können Datensätze eingegeben werden. Die oberste Zeile – die Statuszeile – zeigt, daß jeder Datensatz 382 Zeichen enthalten kann, in der Datei jedoch (noch) keine Daten eingetragen wurden. Nach Aufruf der Option 1 wird das Datensatzformat mit einigen Anweisungen dargestellt. In unserem Beispiel steht dort folgendes:

```
{ESC} Drücken, wenn die Eingabe für Satz 1
beendet ist
^C löscht das aktuelle Feld
Nummer des Herstellers:
Unsere Lagernummer:
Preis:                Lagermenge:
Beschreibung:
Hersteller:
Telefon des Herstellers:
Ansprechpartner:
```

Die Daten werden normal eingegeben. ENTER beschließt die Feldeingabe und stellt den Cursor in das nächste Feld. Fehler lassen sich mit der Löschtaste korrigieren. Nach Eingabe

eines vollständigen Datensatzes ruft die ESC-Taste folgendes Untermenü auf:

0=Ende
1=Weiter
2=Datensatz in die Datei schreiben

Option 2 schreibt den Datensatz auf die Diskette und stellt eine Leermaske für die nächste Datensatzeingabe dar.

Über die Option 2 des Hauptmenüs kann auf Datensätze zugegriffen werden. Auch hier erscheint ein Untermenü:

0=Ende
1=Nummer des Datensatzes
2=Suchen
3=Alle Sätze der Datei anzeigen

Mit Option 1 können bestimme Datensätze angegeben und dargestellt werden – vorausgesetzt, Sie kennen die Satznummer. Die Option 2 läßt aber auch die Eingabe verschiedener Suchparameter zu, mit denen sich bestimmte Datensätze wiederfinden lassen. Hier lautet das Untermenü:

Escape drücken, wenn die Eingabe der Suchkriterien beendet ist. Such-Modus:
^Q=Enthalten, ^W=Nicht enthalten,
^E=Entspricht, ^R=Entspricht nicht,
^T=Größer, ^Y=Kleiner

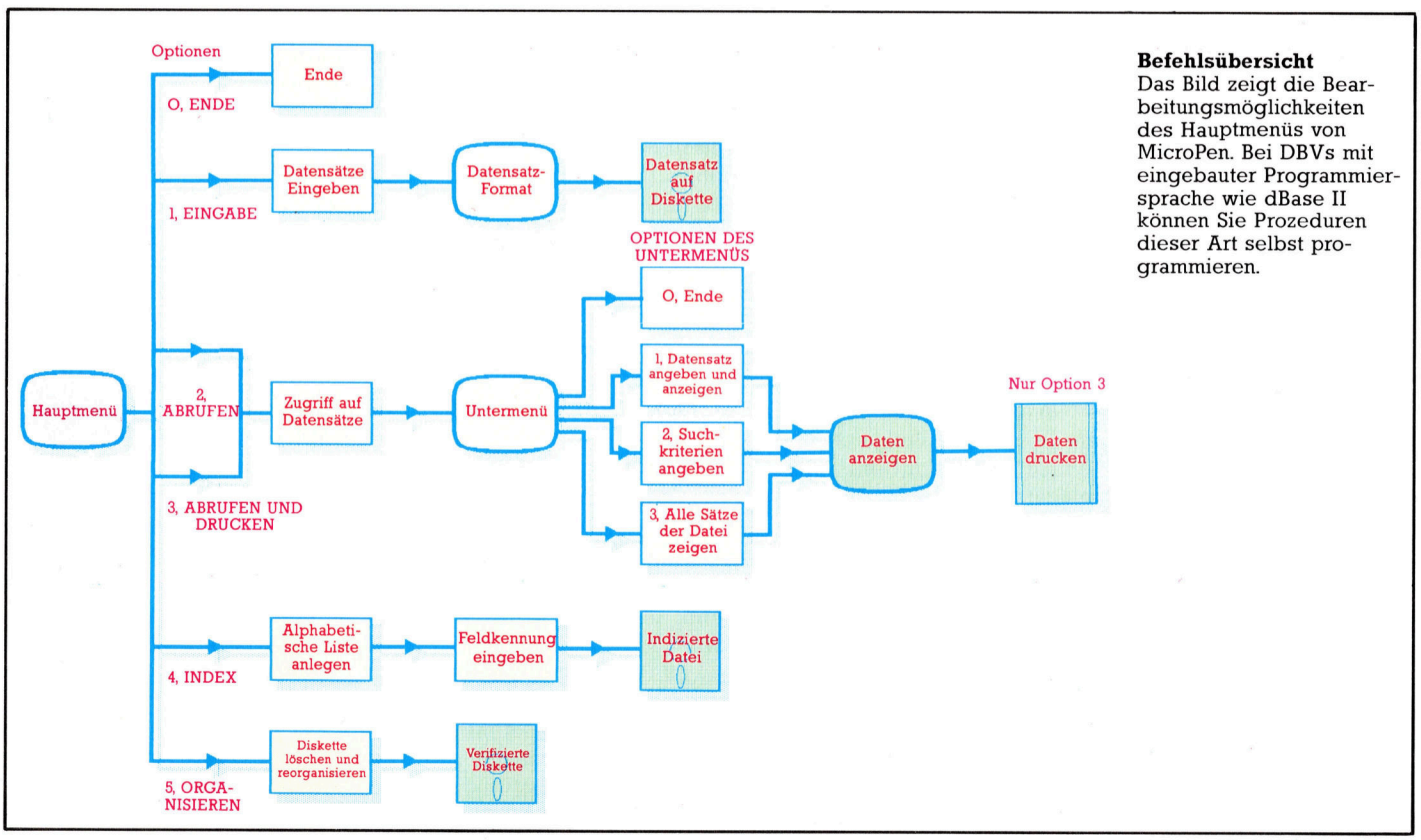
Sie haben sicher bereits bemerkt, daß die Befehle und Menüoptionen nicht einheitlich sind: Mal muß Escape gedrückt werden, mal ein CTRL-Zeichen und mal eine Menüzahl. Auch die Auswahl der CTRL-Zeichen wurde scheinbar willkürlich getroffen – ^E findet Datensätze, deren Felder übereinstimmen. ^Q läßt die Eingabe eines Kriteriums zu. Insgesamt sind die Suchfunktionen jedoch brauchbar.

Nach Eingabe der Suchparameter aktiviert ein CTRL-Zeichen die Suche. Dabei werden alle Datensätze angezeigt, die die gesuchten Daten enthalten; nicht enthalten; entsprechende Felddaten haben oder aber Felddaten enthalten, die niedriger sind als der angegebene Feldwert. Damit lassen sich beispielsweise leicht alle Artikel abrufen, deren Preis unter 37,50 Mark liegt.

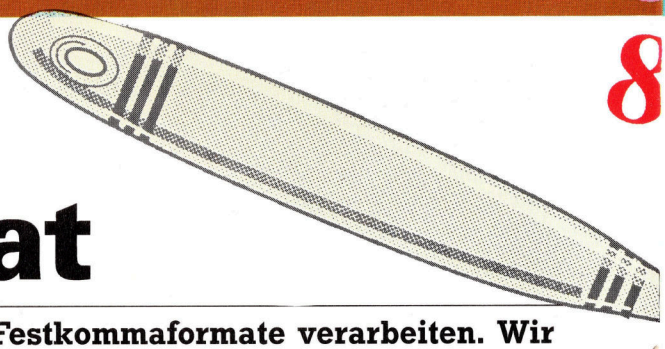
Feldweise Indizierung

MicroPen kann auch einen Index über die Dateieinträge aufbauen. Die Indizierung geschieht feldweise. Dabei muß die maximale Datensatzzahl bekannt sein. Die Felder werden über ihre Kennungen, nicht über den Feldnamen angesprochen. Um bei 500 Datensätzen den Index von „Unsere Lagernummer" zu erhalten, geben wir B=#500 an (B ist die Kennung für das Feld „unsere Lagernummer").

MicroPen bietet ausreichende Such- und Indexmöglichkeiten und ist zudem preisgünstig. Das Programm verfügt jedoch nicht wie Archive und dBase II über eine integrierte Programmiersprache. dBase II ist ein teures Paket, das für größere Geräte entwickelt wurde, wogegen Archive im Lieferumfang des Sinclair QL enthalten ist. Der Preis von Archive ist mit einem Schneider CPC 464 inklusive Diskettenlaufwerk und MicroPen Software vergleichbar. Wenn Sie sich daher einen Computer mit Datenbank zulegen wollen, sollten Sie sich die angebotenen DBV-Pakete genau ansehen.



Befehlsübersicht
Das Bild zeigt die Bearbeitungsmöglichkeiten des Hauptmenüs von MicroPen. Bei DBVs mit eingebauter Programmiersprache wie dBase II können Sie Prozeduren dieser Art selbst programmieren.



Gesiebter Zahlensalat

FORTH kann nur Ganzzahlen und Festkommaformate verarbeiten. Wir untersuchen diese Eigenart am Beispiel des „Sieb des Eratosthenes", das wir schon einmal in der PASCAL-Serie abgedruckt hatten.

Bruchzahlen behandelt FORTH im „Festkommaformat". Die eingesetzten Zahlen werden durch Konstante – normalerweise Zehnerpotenzen – dividiert oder multipliziert.

Die Zahlen des FORTH-Stapels haben je 16 Bits und können entweder vorzeichenbehaftete Ganzzahlen (das Vorzeichen wird vom höchstwertigen Bit angezeigt) zwischen −32768 und +32767 darstellen oder Ganzzahlen ohne Vorzeichen zwischen 0 und 65535. Am einfachsten läßt sich dies durch die Eingabe

```
65535.
```

feststellen. Die Antwort ist −1, da die oben auf dem Stapel liegende Zahl als vorzeichenbehaftet angesehen wird. Die Zahl 65535 ohne Vorzeichen wird auf dem Stapel von dem gleichen Wert dargestellt wie −1 mit Vorzeichen.

Das Wort U. funktioniert wie. und sieht den obersten Stapelwert ohne Vorzeichen an.

```
65535 U.
```

zeigt daher 65535.

Etliche Sprachversionen setzen für diesen Zweck andere Wörter ein. Die Wahl zwischen Vorzeichen oder keinem Vorzeichen wird bei Zahlen zwischen 32768 und 65535 wichtig. Sie müssen sich dann entscheiden, ob Sie die Wörter für vorzeichenbehaftete Zahlen einsetzen wollen (in diesem Fall verursachen Zahlen über 32768 einen Überlauf) oder Zahlen ohne Vorzeichen (in diesem Fall entstehen keine negativen Zahlen). Ein gutes Beispiel dafür sind Werte, die Speicheradressen anzeigen und nur mit Wörtern ohne Vorzeichen eingesetzt werden können – zum Beispiel U< statt <. Sie können aber Stapelzahlen auch als Muster im 16-Bit-Format ansehen und damit normale Boolesche Bitoperationen vornehmen.

Unser Programmbeispiel zeigt mit dem Sieb des Eratosthenes alle Primzahlen bis 65536 an. Wie bei allen FORTH-Programmen läßt sich das Programm rückwärts leichter lesen: Fangen Sie mit dem wichtigsten Wort PRIMES an, und suchen Sie die Wortdefinitionen, die mit diesem Wort arbeiten. PRIMES schreibt zwar als erstes etwas auf den Bildschirm, muß aber als letztes – hinter den untergeordneten Wörtern – eingegeben werden.

Das Sieb des Eratosthenes funktioniert folgendermaßen: Sie schreiben alle Zahlen bis zu einer bestimmten Grenze nieder. Wenn Sie eine Primzahl finden, streichen Sie alle Vielfache dieser Zahl durch, da sie keine Primzahlen sein können. Die nächste Primzahl ist dann der nächste Wert, der nicht ausgestrichen wurde. Nach 1 (die aus verschiedenen Gründen nicht als echte Primzahl gilt) finden Sie die erste Primzahl 2 und streichen alle Vielfache davon aus. Die nächste nicht gestrichene Zahl ist nun die Primzahl 3, von der Sie wiederum alle Vielfache streichen etc.

Unser FORTH-Programm speichert acht KByte – oder 65536 Bits – je ein Bit pro Zahl zwischen 1 und 65536. Die Bits stehen zunächst auf 0, werden aber beim Ausstreichen der Zahl auf 1 gesetzt. Der Befehl

```
CREATE BITS 8192 ALLOT
```

reserviert den Speicherplatz für die Bits. (Wir haben 8192 durch die Konstante BYTES ersetzt.) Wir werden später untersuchen, wie dies funktioniert. Nun ist ein Block von 8192 Bytes irgendwo im Speicher reserviert und das neue Wort BITS angelegt, das die Adresse des ersten Bytes auf den Stapel schiebt. Der Stapel enthält jetzt ein Array von Bytes.

Variable CURBYTES

Für das Ansprechen eines dieser Bits brauchen wir zwei Zahlen – eine Zahl gibt an, welches Byte das Bit enthält (wir können für diese Zahl die Speicheradresse des Bytes einsetzen) und die andere, welches Bit innerhalb des Bytes gemeint ist (also ein Wert zwischen 0 und 7). Es ist nicht allzu schwierig, zwischen diesem Zahlenpaar und der Zahl (zwischen 1 und 65536) umzuschalten, die das Bit darstellt.

CURBYTES ist eine Variable. Ihr Wert CURBYTES @ ist die Adresse eines Bytes im Array BITS, CURBYTES @ C @ dagegen der Wert des Bytes. Wenn Sie die Programmiersprache C kennen, wird Ihnen dieses Konzept vertraut sein. Da Primzahlen und Adressen recht groß werden können, setzen wir gelegentlich Wörter ohne Vorzeichen wie U. und U< ein.

Das Wort:

```
+! ( n, Adresse — )
```

tauscht den Wert der angegebenen Adresse gegen die zuvor dort gespeicherte Zahl aus, auf die n addiert wurde. Es eignet sich daher für die Erhöhung eines Variablenwertes um

Zahlenbasis
In FORTH läßt sich die Zahlenbasis austauschen. Die eingegebenen Zahlen können daher binär, oktal, dezimal oder hex sein oder jede andere Basis haben. Das Format wird über BASE angegeben:

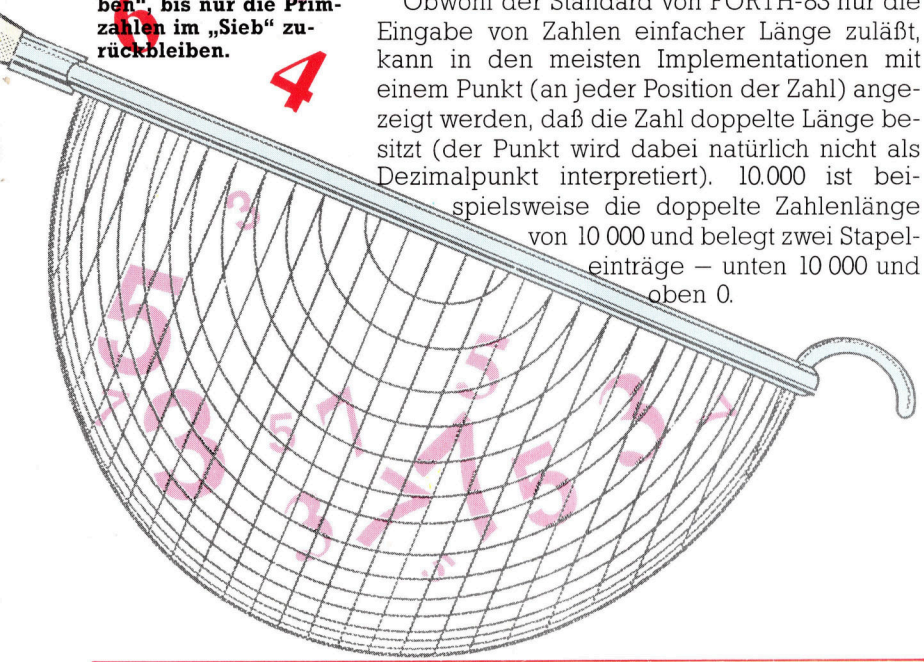
```
54 16 BASE ! . DECIMAL
```

zeigt die Hexzahl 36 (54 Dezimal) an und schaltet mit dem Wort DECIMAL auf die Basis 10 zurück. Es ist einfacher, Wörter wie

```
:HEX 16 BASE !
```

zu definieren. Beachten Sie, daß hier eine Zahl Teil einer Wortdefinition ist und von späteren Änderungen der Zahlenbasis nicht beeinflußt wird.

Das Sieb des Eratosthenes, nach dem griechischen Mathematiker benannt, der es entwickelte, ist eine wirkungsvolle Methode, die Vielfachen einer Zahlenfolge „auszusieben", bis nur die Primzahlen im „Sieb" zurückbleiben.



eine bestimmte, zuvor genau definierte Zahl.

Einige Wörter setzen die Arithmetik mit doppelten Zahlenlängen ein. Dabei stellen zwei Zwei-Byte-Zahlen eine Vier-Byte-Zahl dar. Die höherwertige Zahl liegt näher an der Stapelspitze, die niederwertige halb darunter.

Obwohl der Standard von FORTH-83 nur die Eingabe von Zahlen einfacher Länge zuläßt, kann in den meisten Implementationen mit einem Punkt (an jeder Position der Zahl) angezeigt werden, daß die Zahl doppelte Länge besitzt (der Punkt wird dabei natürlich nicht als Dezimalpunkt interpretiert). 10.000 ist beispielsweise die doppelte Zahlenlänge von 10 000 und belegt zwei Stapeleinträge – unten 10 000 und oben 0.

```
8192  CONSTANT  BYTES
CREATE BITS BYTES ALLOT BITS BYTES
+ CONSTANT BITSEND
(  1 bit for each number 1 - 65536 )
VARIABLE  CURBYTE   VARIABLE  CURBIT
(  show number being looked at.  CURBYTE )
(  points byte in BITS, CURBIT to bit  )
VARIABLE  MASK  (  2**CURBIT  )
VARIABLE  PRIME  VARIABLE  PRIME/8
VARIABLE  PRIMEMOD8
(  show prime when found  )
:  SETUP  (  --  )  (  initialises variables  )
   BYTES  0  DO  (  zero array BITS )
      0  BITS  I  +  C!
   LOOP
   BITS  CURBYTE  !  0  CURBIT  !  1  MASK  !
;
:  NEXTBIT  (  -- 0 or non-zero  )
(  adjusts CURBYTE etc to find next bit  )
(  and stacks it  )
   1  CURBIT  +!
   MASK  @  2*  MASK  !
   CURBIT  @  8  =  IF
      0  CURBIT  !
      1  MASK  !
      1  CURBYTE  +!
   THEN
   CURBYTE  @  C@  MASK  @  AND
;
:  PRIMEETC  (  --  )  (  sets up PRIME, PRIME/8  )
(  and PRIMEMOD8  )
   CURBIT  @  1+
   DUP  8  =  IF
      DROP  0  PRIMEMOD8  !  1
   ELSE
      PRIMEMOD8  !  0
   THEN
   CURBYTE  @  BITS  -  +  PRIME/8  !
   PRIME/8  @  8  *  PRIMEMOD8  @  +  PRIME  !
;
:  ANOTHER-PRIME  (  --  0 or -1  )
(  adjusts CURBYTE etc to next prime, sets  )
(  PRIME etc. 0 left if end of BITS reached  )
   BEGIN
   NEXTBIT  0=  UNTIL
   CURBYTE  @  BITSEND  U<  IF
      PRIMEETC
      -1
   ELSE
      0
   THEN
;
VARIABLE  DELBYTE  VARIABLE  DELBIT
:  DELMASK  (  --  2**DELBIT  )
   1
   DELBIT  @  IF   (  if DELBIT non-zero  )
      DELBIT  @  0  DO  (  multiply 1 by 2  )
         2*             (  2 DELBIT times  )
      LOOP
   THEN
;
:  ANOTHER-MULTIPLE  (  --  0 or -1  )
(  adjusts DELBYTE and DELBIT  )
(  to next bit if set  )
(  0 left if end of BITS array reached  )
   PRIMEMOD8  @  DELBIT  +!
   PRIME/8  @  DELBYTE  +!
   DELBIT  @  7  >  IF
      -8  DELBIT  +!
      1  DELBYTE  +!
   THEN
   DELBYTE @ BITSEND U< BITS
   1- DELBYTE @ U< AND
(  careful in case DELBYTE exceeds 65535  )
;
:  DELETE-MULTIPLES  (  --  )
(  sets bits for multiples of current prime  )
   CURBYTE  @  DELBYTE  !
   CURBIT  @  DELBIT  !
   BEGIN
   ANOTHER-MULTIPLE  WHILE
    DELBYTE  @  C@  DELMASK  OR  DELBYTE  @  C!
   REPEAT
;
:  PRIMES  (  --  )
(  prints all primes between 2 and 65535  )
   SETUP
   BEGIN
   ANOTHER-PRIME WHILE
      PRIME @ U.
      DELETE-MULTIPLES
   REPEAT
;
```

Mathematische Worte

Addition und Subtraktion: Es gibt keinen Unterschied zwischen Addition und Subtraktion mit oder ohne Vorzeichen. + und − stellen einfache Längen dar, D+ und D− doppelte Längen.

Vorzeichen-Minus: NEGATE für einfache Längen, DNEGATE für doppelte Längen. fig-FORTH nimmt dafür MINUS und DMINUS.

Multiplikation: * hat einfache Länge, mit oder ohne Vorzeichen. UM* hat nicht-vorzeichenbehaftete „gemischte" Länge – der Befehl zieht zwei Zahlen ohne Vorzeichen vom Stapel und legt dort das Ergebnis in doppelter Länge ab. In älteren FORTH-Versionen wird UM* auch U* genannt. Es gibt dort auch das vorzeichenbehaftete Produkt M*.

Division: /, MOD und /MOD haben einfache Länge mit Vorzeichen. Sie ergeben entweder den Quotienten, den Rest oder beide (mit dem Quotienten oben auf dem Stapel). UM/MOD ist die Version von /MOD für gemischte Längen ohne Vorzeichen. Sein Dividend hat dabei doppelte Länge. In älteren FORTH-Versionen ist dies U/ oder U/MOD. FORTH-83 rundet immer auf die nächst kleinere Ganzzahl. 10/3 wird daher auf 3 abgerundet, −10/3 jedoch auf −4. Ältere FORTH-Versionen runden immer in Richtung 0, so daß −10/3 −3 ergibt. Nachdem der Quotient mit diesen Regeln berechnet wurde, wird der Rest mit folgender Formel bestimmt:
Dividend = Divisor * Quotient + Rest

Multiplikation mit Division: */ hat einfache Länge mit Vorzeichen und folgende Auswirkung auf den Stapel:
n1,n2,n3——n1*n2/n3

Vergleichsoperationen: = < > und U< haben einfache Länge. < und > haben Vorzeichen, U< ist ohne Vorzeichen und = gilt für beides. D=D< und DU< haben doppelte Länge. D< hat Vorzeichen, DU keine Vorzeichen und D= beides. MAX und MIN zeigen die größere oder die kleinere von zwei Zahlen an. Sie haben einfache Länge mit Vorzeichen. DMAX und DMIN haben doppelte Länge mit Vorzeichen.

Absolute Werte: ABS und DABS gelten jeweils für einfache und doppelte Längen.

Bitweise Boolesche Operationen: AND, OR, XOR und NOT haben einfache Länge. Das Vorzeichen spielt keine Rolle.

Standardkombinationen: Einige Worte werden so oft gemeinsam eingesetzt, daß ihre Kombination ohne jeglichen Zwischenraum geschrieben wird und somit als einzelnes Wort gilt. So hat denn beispielsweise 1+ die gleiche Wirkung wie 1 +.

Darstellung: . und U. haben einfache Länge und können mit oder ohne Vorzeichen sein. D. hat doppelte Länge mit Vorzeichen.

Speicheradressierung: Eine Adresse mit einfacher Länge und ohne Vorzeichen kann sich auf die zwei Bytes dieser Adresse beziehen (wie mit @ und !) oder auf nur ein Byte (mit C@ und C! – C bedeutet „Character") oder auf die dort gespeicherte Vier-Byte-Zahl doppelter Länge (mit 2@ und 2!).

System in Betrieb

Betriebssysteme sind Schnittstellen zwischen Hardware und Programmen. Mit diesem Artikel beginnt eine Serie, in der wir die Betriebssysteme bekannter Heimcomputer untersuchen.

Betriebssysteme (Operating System=OS) sind in einer Maschinensprache geschrieben, die dem Microprozessor des gesteuerten Computers entspricht. So wurde das Betriebssystem des Acorn B im Maschinencode des 6502 entwickelt und das des Spectrum selbstverständlich im Z80-Code.

Ein OS besteht aus Routinen, die vielfältige Maschinenfunktionen ausführen. Eine Routine überprüft beispielsweise ständig, ob Tasten gedrückt wurden. Abläufe dieser Art braucht der Anwender daher nicht erst zu entwickeln. Mit einem guten OS sollte es sogar möglich sein, jede Hardwarefunktion anzusprechen, ohne daß die exakten Speicherpositionen der einzelnen Routinen oder Speichertabellen bekannt sein müssen. Auch Hardwareänderungen lassen sich per OS leichter integrieren. Dabei wird das Betriebssystem so verändert, daß die Programme älterer Maschinen ebenfalls laufen.

Das Betriebssystem des Acorn B ist durchdacht aufgebaut. Ein Programm, das auf einem Standardgerät im BASIC des Acorn B geschrieben wurde, funktioniert auch auf einer Maschine, die mit einem zweiten Prozessor ausgerüstet ist – trotz der umfassenden Hardwareänderung. Das OS des Acorn B mußte allerdings mehrere Entwicklungsstadien durchlaufen, bevor es seine augenblickliche hochentwickelte Form erhielt. Die erste Version – 0.1 – bestand aus vier EPROM-Chips, sie ist inzwischen fast vollständig vom Markt verschwunden. 0.1 war nicht ausreichend flexibel und konnte keine Diskettenlaufwerke ansprechen. Die Version 1.0 unterstützte die Speicherung auf Diskette. Sie bestand aus einer kleinen Platine mit zwei Acht-KByte-Chips im Inneren des Geräts. Die aktuelle Version – 1.2 – ist heute auf fast allen Acorn-B-Geräten zu finden. Eine Variante davon wurde speziell für den amerikanischen Markt entwickelt.

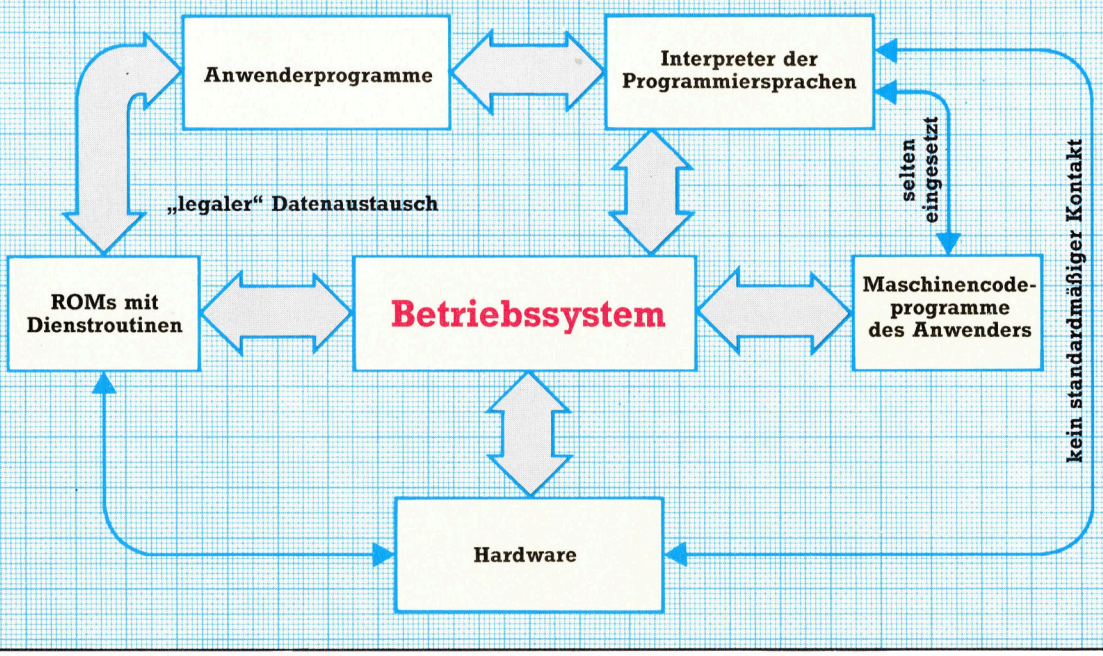
Blick ins Innenleben

Bei der Eingabe von *FX0 und Return erscheint die Systemnummer Ihrer Maschine auf dem Bildschirm. Der Befehl *HELP zeigt außer der OS-Nummer auch die Namen der eingebauten ROM-Chips.

Die Betriebssystemaufgaben des Acorn B lassen sich in vier Hauptbereiche unterteilen:
1) **Eingaberoutinen:** Diese Routinen nehmen Informationen aus dem „Current Input Stream" auf. Normalerweise wird dabei die Tastatur angesprochen, doch auch die RS423-Schnittstelle und das Dateisystem (das mit dem Befehl *EXEC aktiviert wird) gehören in diese Kate-

Betrieb läuft

Bei einem guten Betriebssystem merkt man nicht, daß es arbeitet. Für den Anwender sollte die Schnittstelle zwischen Programmen und Hardware (das heißt die Steuerung der Diskettenlaufwerke, das Zwischenspeichern von Ausgaben und die Bildschirmanzeige etc.) völlig einsichtig sein. Weiterhin sollte das OS der Soft- und Hardware ständig korrekt formatierte und zeitlich aufeinander abgestimmte Ein- und Ausgaben bieten.



gorie. Die wichtigste OS-Routine der Eingabeverarbeitung heißt OSRDCH (Operating System ReaD CHaracter – OS-Zeichen einlesen).
2) Routinen für Ausgabe und Anzeige: Diese Routinen steuern die Ausgaben des Computers. Der Acorn B kann eine ganze Reihe von Ausgabemöglichkeiten als „Current Output Stream" einsetzen, beispielsweise den Bildschirm, Drucker, die RS423-Schnittstelle und (über den Befehl *SPOOL) das aktuelle Dateisystem. Außer Druck und Datenausgabe steuern diese Routinen unter anderem auch den 6845-Bildschirmchip des Computers und den Einsatz der vom Anwender definierten Zeichen. Das OS ruft dafür OSWRCH (Operating System WRite CHaracter – OS-Zeichen darstellen), OSASCII und OSNEWL auf.
3) Dateisysteme: Jedes Betriebssystem muß dem Anwender die Möglichkeit geben, den Inhalt des Arbeitsspeichers auf ein permanentes Speichermedium zu sichern. Dieser Betriebssystembereich wird „Aktuelles Dateisystem" genannt, wobei der Acorn B die Wahl zwischen Cassette, Diskette, Econet, ROM oder Telesoftware bietet. Die OS-Routinen haben außerdem zusätzliche ROM-Module zur Verfügung, mit denen das OS die Hardware der einzelnen Dateisysteme steuern kann.

Um über die ROMs des seitenbezogenen Dateisystems einfache Schnittstellenverbindungen zu den magnetischen Speichermedien herstellen zu können, wurden einige Standardroutinen des OS für die Dateispeicherung abgestellt. Darunter gibt es Module, die ganze Dateien lesen oder schreiben, einzelne Bytes aus einer offenen Datei lesen oder Bytegruppen auf dem Speichermedium ansprechen. Die sieben Submodulaufrufe des Betriebssystems, die für Dateien zuständig sind, zeigen mit Vektoren auf die entsprechenden Subroutinen des Cassettensystems. Ein ROM mit seitenbezogenem Dateisystem kann diese Vektoren verändern und so die eigenen Dateiprogramme für BASIC oder Assembler einsetzen. Mit einem ROM dieses Typs (DFS ROM) kann der Acorn B Diskettenlaufwerke ansprechen.
4) Interrupts: Interrupts sind von der Hard- oder Software erzeugte Signale, die die CPU veranlassen, ihre aktuelle Aufgabe zu unterbrechen und Vorgänge auszuführen, die im Augenblick wichtiger sind. Danach nimmt die CPU ihre ursprüngliche Arbeit wieder auf. Der Acorn B besitzt eine Reihe Interrupt-gesteuerter Routinen, die dem Anwender über OS zur Verfügung stehen.

Zwei heiße Drähte

Außer diesen vier Hauptbereichen gibt es noch die beiden Systemaufrufe OSBYTE und OSWORD, die eine große Anzahl unterschiedlicher Funktionen steuern: den Tonerzeugungschip, die Break-Taste etc.

In den meisten Fällen steht der Aufbau des Arbeitsspeichers und der Hardware schon wenige Monate nach Vorstellung einer neuen Maschine zur Verfügung. Doch wenn diese Einzelheiten bekannt sind, warum sollen wir uns dann überhaupt um Betriebssystemaufrufe kümmern, und warum können wir Funkionseinheiten oder Speicher nicht ganz einfach und viel besser direkt ansprechen?

Große Flexibilität

Einen Teil der Antwort haben wir schon am Anfang des Artikels gegeben: Mit OS-Aufrufen sind die Programme gegenüber Änderungen flexibler, die der Hersteller in Hardware und Konfiguration vornimmt. Umgekehrt verursacht der Einsatz fester ROM-Adressen große Probleme, wenn das ROM umgestellt wird. Beim Aufruf von ROM-Routinen über das OS brauchen Sie Änderungen nicht zu beachten.

Der folgende Befehl schreibt auf einem Acorn B den Wert 200 an den User Port bei der Adresse &FE60.

```
?&FE60=200
```

Verfügt das Gerät jedoch über einen zweiten Prozessor, dann wird der Wert nicht an den User Port, sondern in eine Speicherstelle des zweiten Prozessors gesandt. Mit dem entsprechenden OS-Aufruf läßt sich das Problem jedoch umgehen. Die Routine „weiß" dann, wie der Wert zum User Port gesandt wird, unabhängig davon, ob ein zweiter Prozessor vorhanden ist oder nicht:

```
*FX 151,96,200
```

Es gibt mehrere Aufrufe, mit denen man sich auf dem Acorn B beispielsweise Zugang zu dem VIA (Versatile Interface Adaptor), dem System-VIA oder aber auch dem 1Megahertz-Bus verschaffen kann.

Warum also das Rad nochmals erfinden? Wenn eine Routine für eine bestimmte Maschinenfunktion bereits existiert, brauchen die ROM-Routinen nicht direkt angesprochen zu werden. Betriebssystemaufrufe sind im allgemeinen leistungsfähiger als der direkte Zugriff auf ROM-Routinen.

Nur wenn man hohe Verarbeitungsgeschwindigkeiten braucht oder für eine bestimmte Aufgabe keine OS-Routinen existieren, sollten Sie direkt mit dem Speicher oder der Hardware arbeiten, da sich hohe Geschwindigkeiten oft nur durch direkten Zugriff und nicht über OS-Routinen erreichen lassen. Dieser Weg ist jedoch nicht einfach, und Programme, die auf einer bestimmten Maschine laufen, funktionieren unter Umständen nicht auf Geräten mit nur geringfügig geänderter Hardware oder einer anderen OS-Version.

In der nächsten Folge sehen wir uns die OS-Routinen des Acorn B im einzelnen an. Den Spectrum und andere Maschinen untersuchen wir zu einem späteren Zeitpunkt.

Auf großer Fahrt

Die großen Entdekkungsfahrten fanden gegen Ende des 15. Jahrhunderts statt. Im 16. Jahrhundert begannen die europäischen Seefahrer mit der Erstellung von Karten. Am Anfang waren diese Karten nur Skizzen, und oft wurden Inseln – oder sogar ganze Kontinente – an den falschen Positionen eingezeichnet.

Bisher haben wir in unserem Handelsspiel die Mannschaft angeheuert und Vorräte für die Fahrt gekauft. Bevor jedoch endgültig die Segel gesetzt werden können, müssen noch Handelswaren gekauft werden.

Der Spieler muß entscheiden, wieviel Gold er in Handelsgüter investiert, und wieviel er – wenn überhaupt – zur Seite legt, um nach der Rückkehr die Mannschaft auszuzahlen. Verfügbare Waren sind Medizin, Pistolen, Salz, Kleider, Messer und Juwelen. Der Spieler wird über die Preise informiert und gefragt, wieviel er kaufen möchte. Nach jeder Transaktion wird der aktuelle Kontostand dargestellt. Reicht das Gold nicht mehr zur Bezahlung, wird der Spieler aufgefordert, seine Eingabe zu korrigieren. Der Laderaum des Schiffes faßt jede beliebige Menge Waren. Sind die Einkäufe beendet, kann die Fahrt beginnen. Bevor das Schiff Segel setzt, wird eine Aufstellung der Zusammensetzung der Mannschaft, der Vorräte und Waren sowie des noch vorhandenen Goldes dargestellt.

Wie zuvor werden die Arrays, die Namen, Preise und Mengen der gekauften Waren enthalten, am Programmanfang DIMensioniert. Die Nummer jeder gekauften Ware wird im Array OA() gespeichert. Die Beschreibungen der anderen Güter werden in dem in Zeile 31 DIMensionierten Array D$() abgelegt. Die Elemente dieses Arrays werden in den nächsten zwei Zeilen gesetzt. Das erste, D$(1), erhält den Inhalt "BOTTLE OF MEDICINE". Die übrigen fünf Elemente, D$(2) bis D$(6), beinhalten "GUN", "BAG OF SALT", "BALE OF CLOTH", "KNIFE" und "JEWEL".

Die Preise für diese Güter werden im Array OC() mit sechs Elementen in den Zeilen 34 und 35 abgelegt. Das erste Element, OC(1), repräsentiert den Preis der Medizin, die pro Flasche ein Goldstück kostet. Pistolen kosten fünf Goldstücke [OC(2)=5], und mit einem Goldstück kauft man fünf Beutel Salz [OC(3)=0.2]. Ein Bündel Kleider kostet zwei [OC(4)=2], ein Messer ein halbes [OC(5)=0.5], und Juwelen kosten ein Goldstück [OC(6)=1].

In Zeile 600 erfolgt der Aufruf der Unterroutine zum Kauf der Handelsgüter. Nach Löschen des Bildschirms erfährt der Spieler, daß auch andere Waren für die Fahrt benötigt werden.

Der Programmteil zum Kauf der Handelswaren und Güter ist als Schleife organisiert. Sie wird für jede Warenart, unter Verwendung von T als Schleifenzähler, einmal durchlaufen. Dem Spieler wird der Preis und mittels D$(T)

in Zeile 3070 die Warenart genannt.

Beim ersten Durchlauf wird T für die erste Ware – Medizin – auf 1 gesetzt. Der Einzelpreis wird in Zeile 3080 durch OC(T) angegeben. Ist T auf 1 gesetzt, repräsentiert OC(T) ein Goldstück – der Preis für eine Flasche Medizin. Beim zweiten Durchlauf des Programms wird T auf 2 gesetzt, D$(T) beinhaltet "GUN" (die Pistole) und OC(T) den Preis von fünf Goldstücken. Der Vorgang ist beendet, wenn T den Wert 6 erreicht hat.

Gold gegen Waren

Kostet eine Ware mehr als ein Goldstück, wird durch Zeile 3081 ein S an "GOLD PIECE" angefügt. Es folgt eine Verzögerung, und der Spieler wird gefragt "WOULD YOU LIKE TO BUY? (Y/N)". In Zeile 3110 wird auf eine Antwort gewartet. Das ganz links befindliche Zeichen der Eingabe wird durch Zeile 3120 überprüft. Der Programmlauf wird zu Zeile 3175 und dann zum Startpunkt der Schleife verzweigt, wenn es ein "N" ist. War die Eingabe "Y" oder "YES", erfolgt eine Verzögerung und die Frage "HOW MANY DO YOU WANT?". Durch Zeile 3135 können Sie beispielsweise 10 SACKS OF SALT oder einfach 10 eingeben. Diese Alternative ist möglich, da in Zeile 3140 die Eingabe mittels der Funktion VAL untersucht und somit der Text ignoriert wird.

Für den Fall, daß Sie versuchen, einen illegalen Kauf zu tätigen (zum Beispiel, wenn Sie nicht genug Gold haben), überprüft Zeile 45, ob die Kosten der Ware, OC(T), multipliziert mit der Menge, TT, größer als das übrige Gold, MO, sind. In diesem Fall geht das Programm zu Zeile 3150, in der der Spieler über seine Zahlungsunfähigkeit informiert wird. Nach einer Pause wird durch Zeile 3154 "PLEASE ENTER AGAIN" dargestellt. Das Programm kehrt zu Zeile 3130 zurück und fragt "HOW MANY DO YOU WANT?". Ist genug Kapital da, wird zu Zeile 3160 verzweigt, wo das Gold mittels der Formel MO=MO-(OC(T)*TT) vom Kapital subtrahiert wird. Die Kosten der Ware, OC(T), werden mit der Menge, TT, multipliziert. TT wird im Array OA() in Zeile 3165 gespeichert, und nach einer Pause wird durch Zeile 3176 eine Leerzeile und der Kassenstand ausgegeben. Nach einer weiteren Pause wird der Wert von T um 1 erhöht, und das Programm kehrt durch Zeile 3200 zum Start der Schleife zurück.

Wurden alle Waren angeboten, wird dies durch Zeile 3210 mitgeteilt. In Zeile 3230 wird unter Verwendung der Variablen K$ "PRESS ANY KEY TO CONTINUE" ausgegeben.

Nun erscheint die Meldung "YOU HAVE THE FOLLOWING CREW", und mittels einer Schleife in Zeile 645 (FOR T=1 TO 5) wird die Mannschaft aufgelistet. In Zeile 650 wird durch Überprüfung von CC(T) die Anzahl der Matrosen, Köche usw. ermittelt. Ist ein „Mannschaftstyp" nicht vorhanden (Wert von T ist 0), wird durch Zeile 670 zum Schleifenanfang zurückverzweigt und der Wert von T um 1 erhöht. Ansonsten wird die Anzahl der Männer dargestellt. Hinter der Beschreibung wird ein S oder eine Leerstelle ausgegeben. Dieser Vorgang wiederholt sich fünfmal. In ähnlicher Form wird durch die Zeilen 675 bis 710 auch eine Vorratsliste dargestellt. Die Schleife (in Zeile 685) wird viermal durchlaufen.

Handelsgüter-Abfrage

Die Handelsgüter werden nicht mittels einer Schleife aufgelistet, da ihre Bezeichnungen zu unterschiedlich sind. Deshalb werden sie separat behandelt. Wurde eine Ware nicht gekauft, geht das Programm zur nächsten über. In Zeile 730 wird beispielsweise überprüft, ob MEDIZIN gekauft wurde. Wenn nicht, so wird der Programmlauf mit Zeile 740 fortgesetzt. Wurde MEDIZIN gekauft, bestimmt das Programm, ob es eine oder mehrere Flaschen wa-

Hier sehen Sie die Waren, die die Entdecker wahrscheinlich mitnahmen, um mit den Eingeborenen der Neuen Welt zu handeln. In unserer Simulation werden die Namen der Handelsgüter im Array D$ gespeichert. Der Preis pro Einheit befindet sich im Array OC, wogegen die Anzahl gekaufter Einheiten im Array OA gespeichert wird.

	(1)	(2)	(3)	(4)	(5)	(6)
D$	Bottle of Medicine	Gun	Bag of Salt	Bale of Cloth	Knife	Jewel
OC	1	5	3	11	2	1
OA	6	20	15	32	20	10

ren. Entsprechend wird ein S oder ein Leerzeichen angehängt. Anschließend wird die Menge ausgegeben und mit der nächsten Warenart fortgefahren.

Ist die Liste komplett, wird in Zeile 792 über das verbliebene Gold informiert. Dann folgt in Zeile 797 die Meldung "PRESS ANY KEY TO START VOYAGE".

Modul Drei: Handelsgüter

Array-Dimensionierung

```
30 DIMOA(6)
31 DIMD$(6)
32 D$(1)="BOTTLE OF MEDICINE":D$(2)="GUN":D$(3)="B
AG OF SALT"
33 D$(4)="BALE OF CLOTH":D$(5)="KNIFE":D$(6)="JEWE
L."
34 DIMOC(6)
35 OC(1)=1:OC(2)=5:OC(3)=.2
36 OC(4)=2:OC(5)=.5:OC(6)=1
```

Aufruf der Handelsgüter-Subroutine

```
600 GOSUB3000
```

Eröffnungsroutine

```
605 REM **** READY TO START ****
610 PRINTCHR$(147)
615 S$="YOU ARE NOW READY TO START*":GOSUB9100
625 S$="THE VOYAGE.*":GOSUB9100
630 GOSUB9200
635 PRINT:S$="YOU HAVE THE FOLLOWING CREW:*":GOSUB
9100
640 GOSUB9200
645 FORT=1TO5
650 IFCC(T)=0THEN670
655 PRINTCC(T);
660 PRINTC$(T);
662 IF CC(T)=1THENPRINT" ":GOTO668
664 PRINT"S"
668 GOSUB9200
670 NEXT
674 GOSUB9200
675 PRINT:S$="AND THE FOLLOWING PROVISIONS:*":GOSU
B9100
680 GOSUB9200
685 FORT=1TO4
690 IFPA(T)=0THEN710
695 PRINTPA(T);U$(T);"S OF ";
700 PRINTP$(T)
708 GOSUB9200
710 NEXT
715 GOSUB9200
720 PRINT:S$="YOU HAVE ALSO GOT:*":GOSUB9100
725 GOSUB9200
730 IFOA(1)=0THEN740
733 IFOA(1)=1THENS$="BOTTLE OF MEDICINE *":GOTO
735
734 S$="BOTTLES OF MEDICINE*"
735 PRINTOA(1);:GOSUB9100
736 GOSUB9200
740 IFOA(2)=0THEN750
743 IFOA(2)=1THENS$="GUN*":GOTO745
744 S$="GUNS*"
745 PRINTOA(2);:GOSUB9100
746 GOSUB9200
750 IFOA(3)=0THEN760
753 IFOA(3)=1THENS$="BAG OF SALT*":GOTO755
754 S$="BAGS OF SALT*"
755 PRINTOA(3);:GOSUB9100
756 GOSUB9200
760 IFOA(4)=0THEN770
763 IFOA(4)=1THENS$="BALE OF CLOTH*":GOTO765
764 S$="BALES OF CLOTH*"
765 PRINTOA(4);:GOSUB9100
766 GOSUB9200
770 IFOA(5)=0THEN780
773 IFOA(5)=1THENS$="KNIFE*":GOTO775
774 S$="KNIVES*"
775 PRINTOA(5);:GOSUB9100
776 GOSUB9200
780 IFOA(6)=0THEN790
783 IFOA(6)=1THENS$="JEWEL*":GOTO785
784 S$="JEWELS*"
785 PRINTOA(6);:GOSUB9100
786 GOSUB9200
790 GOSUB9200
792 PRINT:PRINT"YOU HAVE ":MO;" GOLD PIECES
LEFT"
796 GOSUB9200
797 S$="PRESS ANY KEY TO START VOYAGE*"
798 GOSUB9100
799 GETP$:IF P$=""THEN799
999 END
```

Handelsgüter-Subroutine

```
3000 REM **** STAGE 3 OTHER GOODS ****
3001 PRINTCHR$(147): REM STAGE 3
3002 GOSUB9200
3005 PRINT"      STAGE 3 - OTHER GOODS"
3010 PRINT"      -------   -----------"
3020 GOSUB9200
3025 PRINT
3030 S$="THERE ARE OTHER THINGS THAT MAY*":GOSUB91
00
3035 S$="BE USEFUL ON THE VOYAGE, FOR  *":GOSUB910
0
3040 S$="EXAMPLE MEDICINE AND TRADING  *":GOSUB910
0
3045 S$="GOODS.*":GOSUB9100
3046 GOSUB9200
3050 S$="YOU MAY ALSO NEED WEAPONS.*":GOSUB9100
3055 GOSUB9200:GOSUB9200
3060 FORT=1TO6
3065 PRINT
3070 PRINT"A ":D$(T);
3075 S$=" COSTS*":GOSUB9100
3080 PRINTOC(T);
3081 PRINT" GOLD PIECE";
3085 IFOC(T)=1THENPRINT" ":GOTO3090
3086 PRINT"S"
3090 GOSUB9200
3095 S$="WOULD YOU LIKE TO BUY (Y/N)*":GOSUB9
100
3110 INPUTP$:P$=LEFT$(P$,1)
3115 IF P$<>"Y"AND P$<>"N" THEN 3095
3120 IF P$="N"THEN3175
3125 GOSUB9200
3130 S$="HOW MANY DO YOU WANT*":GOSUB9100
3135 INPUTP$
3140 TT=VAL(P$)
3145 IFOC(T)*TT>MOTHEN3150
3147 GOTO3160
3150 S$="YOU DON'T HAVE ENOUGH MONEY*":GOSUB9
100
3152 GOSUB9200
3154 S$="PLEASE ENTER AGAIN  *":GOSUB9100
3155 GOSUB9200:GOTO3130
3160 MO=MO-(OC(T)*TT)
3165 OA(T)=TT
3170 GOSUB9200
3175 PRINT
3176 PRINT"MONEY LEFT = ":MO
3200 GOSUB9200:NEXT T
3205 GOSUB9200:PRINT:PRINT
3210 S$="END OF STAGE 3*":GOSUB9100
3220 GOSUB9200:PRINT
3230 S$=K$:GOSUB9100
3240 GETP$:IFP$=""THEN3240
3999 RETURN
```

BASIC-Dialekte

Spectrum:
Ändern Sie folgende Programmzeilen:

```
 32 DIM D$ (6,20)
610 CLS
799 LET P$=INKEY$: IF P$ = "" THEN GO TO 799
3001 CLS
3110 INPUT P$ : LET P$ = P$ (1 TO 1)
3240 LET P$=INKEY$: IF P$ = "" THEN GO TO 3240
```

Acorn B:
Führen Sie folgende Änderungen durch:

```
610 CLS
799 A$=GET$
3001 CLS
3240 A$=GET$
```

Blick fürs Detail

Geräte für die optische Zeichenerkennung sind schon seit etwa 1955 bekannt, aber erst in jüngster Zeit werden zuverlässige Lesegeräte angeboten. Der Omnireader der Firma Oberon International mit Prozessor für die Zeichenanalyse liest und verarbeitet Textvorlagen.

Der Omni-reader ist das erste optische Zeichenlesegerät, das zuverlässig arbeitet und preislich schon für kleine bis mittlere Büros in Frage kommt. Der Leser erlaubt die Abtastung von maschinengeschriebenem oder gedrucktem Text und die unmittelbare Übergabe an ein Textverarbeitungsprogramm; Sie können also am Bildschirm des Rechners editieren, ohne erst das ganze Schriftstück eintippen zu müssen. Vorerst bietet Oberon die zugehörige Software nur für die gängigsten Bürocomputer an, aber im Prinzip ist der Omni-reader für jeden Rechner mit RS232C-Schnittstelle geeignet.

In den letzten Jahren hat es viele Spekulationen über die Idee und die Realisierung eines „papierlosen" Büros gegeben. Der Begriff rührt daher, daß die moderne Bürokommunikation theoretisch ohne Papierdokumente auskommen könnte, wenn Informationen direkt von einem Rechner zum andern übertragen würden. Die Prophezeiungen waren aber doch wohl etwas voreilig, denn die meisten Leute bringen ihre Gedanken immer noch mit der „Feder" zu Papier. Die Dateneingabe erfolgt im heutigen elektronischen Büro, das vielleicht eine Vorstufe zum „papierlosen" darstellt, immer noch durch Eintippen von Hand. Druckschrift-Vorlagen sind aber neuerdings mittels „optischer Zeichenerkennung" (Optical Character Recognition=OCR) sehr viel schneller in den Rechner zu übertragen.

Hier wird der Omni-reader von Oberon International vorgestellt, der über einen eigenen Prozessor verfügt. Er ist primär für Bürorechner gedacht; daher ist auch vorerst nur Software für IBM-kompatible PCs sowie die Apricot-Rechner und den Apple Macintosh erhältlich.

Beim Omni-reader wird die Vorlage auf eine Kunststofftafel gelegt und dann zeilenweise mit einem Lesekopf abgetastet, der über eine Schiene geschoben wird. Diese Schiene ist auf einer linker Hand angebrachten Führungsstange vertikal verschiebbar. Am Abtastkopf befinden sich zwei Drucktasten und eine Leuchtdiode.

Sie können verschiedene Betriebsarten und Schrifttypen vorwählen, die dann über LEDs am oberen Rand des Tabletts angezeigt werden. Derzeit ist der Omni-reader nur für die vier Schriftarten Courier 10, Courier 12, Letter Gothic 12 und Prestige Elite 12 eingerichtet; das sind die gängigsten Schriften für Schreibmaschinen und Typenraddrucker.

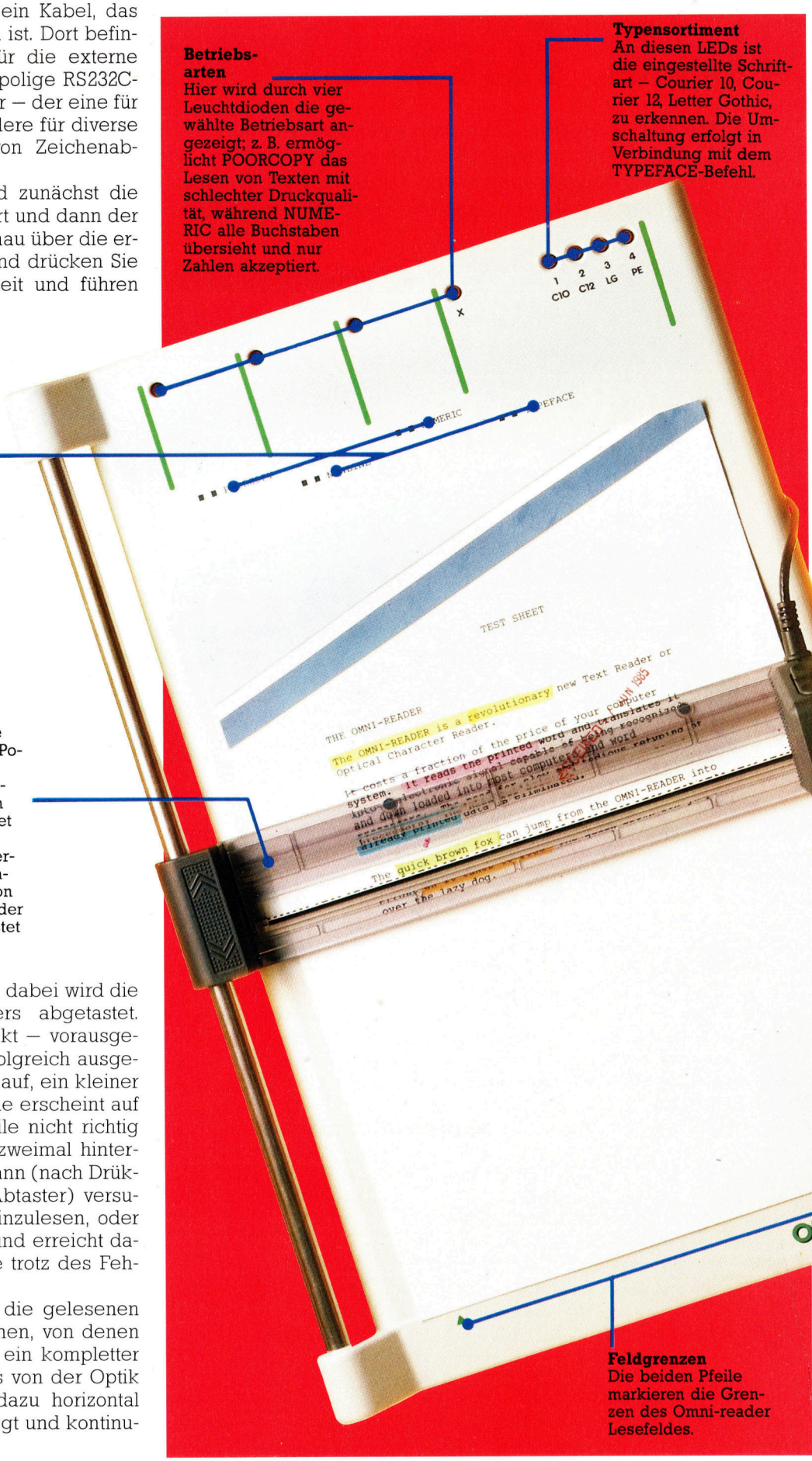
Infrarot-Lichtquellen

Die Abtasteinheit enthält an ihrer Unterseite zwei Infrarot-Lichtquellen zu beiden Seiten der Linse, die die Zeichen auf einen Bildempfänger projiziert. Das System kann nur Schriften erkennen, die mit kohlehaltiger Schwärze (z. B. Carbonfarbband) erzeugt oder aber fotokopiert sind – der Toner enthält meist auch Kohlenstoff. Das hat seine Vor- und Nachteile: Stempelfarbe oder handschriftliche Zusätze mit Kugelschreiber sind für den Detektor nicht erkennbar; außerdem funktioniert der Omni-reader auch auf farbigem Papier meist noch einwandfrei. Andererseits müssen Bleistiftstriche wegen ihres Kohlegehalts wegradiert werden, um Lesefehler zu vermeiden.

An der Abtasteinheit hängt ein Kabel, das hinten am Tablett einzustecken ist. Dort befindet sich auch eine Buchse für die externe Spannungsversorgung, eine 25polige RS232C-D-Buchse und zwei DIP-Schalter – der eine für die Baudratenwahl und der andere für diverse Funktionen wie Einstellung von Zeichenabstand oder Handshake-Betrieb.

Zum Lesen von Texten wird zunächst die Vorlage auf dem Tablett plaziert und dann der Fensterausschnitt im Lineal genau über die erste Zeile gebracht. Anschließend drücken Sie einen Knopf an der Leseeinheit und führen diese einmal über das Lineal – dabei wird die Schrift innerhalb des Fensters abgetastet. Nach etwa einer Sekunde blinkt – vorausgesetzt, daß der Lesevorgang erfolgreich ausgeführt wurde – die Leuchtdiode auf, ein kleiner Piepser ertönt, und die Textzeile erscheint auf dem Bildschirm. Wenn die Zeile nicht richtig erfaßt ist, blinkt und piept es zweimal hintereinander. Der Benutzer kann dann (nach Drükken des zweiten Knopfs am Abtaster) versuchen, die Zeile noch einmal einzulesen, oder er drückt den anderen Knopf und erreicht damit, daß der Rechner die Zeile trotz des Fehlers akzeptiert.

Der Omni-reader vergleicht die gelesenen Zeichen mit Bitmuster-Schablonen, von denen für jede vorhandene Schriftart ein kompletter Satz intern gespeichert ist. Das von der Optik übertragene Schriftbild wird dazu horizontal und vertikal in 50 Streifen zerlegt und kontinuierlich analysiert.



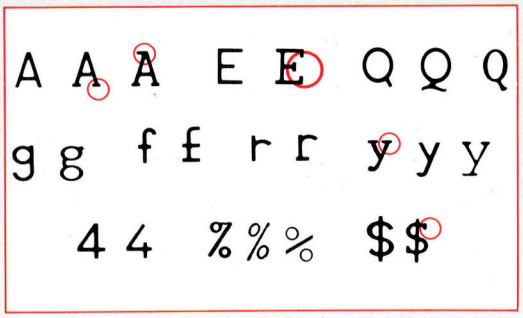
OMNI-READER

AUSSTATTUNG:
Schnittstellenkabel, Software, einjährige Wartung des Gerätes am Standort.

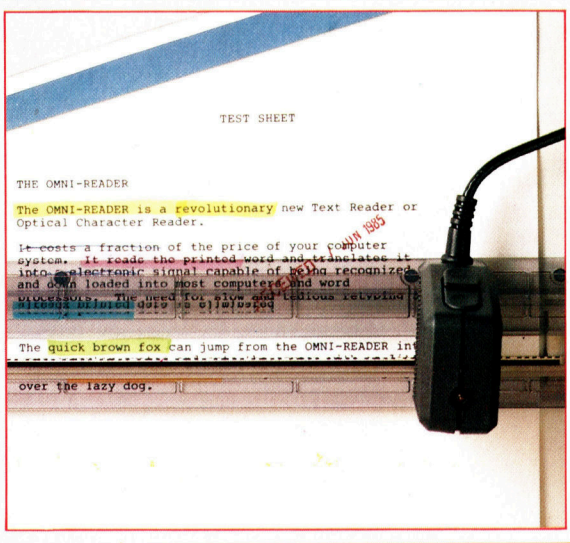
SCHNITTSTELLE:
RS232C-Normschnittstelle mit einstellbarer Baudrate.

STÄRKEN:
Zuverlässiges Arbeiten; sehr schnelle Texteingabe.

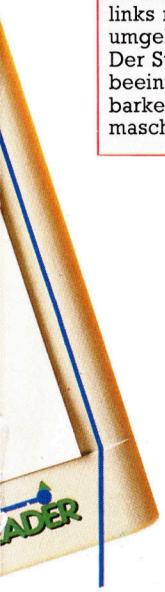
SCHWÄCHEN:
Anzahl der lesbaren Schriftarten und -größen hardwaremäßig begrenzt.



Zweifelhafte Typen
Bei oberflächlicher Betrachtung sehen die Buchstaben hier zwar fast gleich aus, aber die elektronische Erkennung wird durch die feinen Unterschiede doch sehr erschwert. Kritisch sind vor allem die eingekreisten feinen Seriphen – da der Omni-reader mit detailliert vorgegebenen Vergleichsschablonen arbeitet, akzeptiert er ein Symbol nur bei völliger Deckungsgleichheit mit dem gespeicherten Zeichenraster.



Blick durchs Fenster
Die Textzeile im Fenster des Abtastlineals muß exakt zentriert sein. Am unteren Fensterrand sind eine Reihe von Markierungen angebracht, die an einen Strichcode erinnern. An der Abfolge der Balkenlängen erkennt der Omni-reader beim Abtasten die Bewegungseinrichtung des Lesekopfs – der Benutzer kann die Zeilen nach Wunsch von links nach rechts oder umgekehrt abfahren. Der Stempelaufdruck beeinträchtigt die Lesbarkeit des Schreibmaschinentexts nicht.



Optische Rückmeldung
Nachdem die Abtasteinheit über eine Textzeile geführt ist, leuchtet eine Diode am Abtastkopf auf – einmal kurz, wenn die Zeile korrekt gelesen wurde, und zweimal bei einem Fehler.

Dabei muß der Omni-reader zunächst erkennen, wann bei der Bewegung des Abtasters über die Zeile ein vollständiges Zeichen erfaßt wird. Das Zeichen wird nur verarbeitet, wenn es mittig im Gesichtsfeld liegt. Anschließend führt das Gerät den Vergleich mit dem Typensatz im Speicher durch.

Abtastfehler möglich

Den ASCII-Code des Zeichens, mit dem Übereinstimmung festgestellt ist, sendet der Omni-reader über die RS232C-Schnittstelle an den Rechner, woraufhin das zugehörige Symbol unmittelbar auf dem Bildschirm erscheint. Wenn die Zeile nicht richtig im Abtastfenster justiert ist, können Fehler auftreten, insbesondere bei Buchstaben mit sogenannten Unterlängen wie y, g, p oder q.

Liegt umgekehrt das Fenster zu tief, versucht der Abtaster, auch die oberen Ausläufer der darunterliegenden Buchstaben zu lesen. Ähnliche Probleme gibt es mit zu kleinen oder zu großen Typen, weshalb der Hersteller die Schriftgröße auf 10- oder 12-Punkt-Schriften (mit 4 oder 5 Zeichen/cm) beschränkt hat – was darüber hinausgeht, führt zu hohen Fehlerquoten.

Im Rahmen der genannten Grenzen arbeitet der Omni-reader sehr zuverlässig. Die optimale Abtastzeit für eine A-4-Textzeile liegt zwischen 0,5 und 1,5 Sekunden. Schwierigkeiten bei der Einstellung der Zeilen treten nur anfänglich auf; nach weniger als einer halben Stunde haben Sie die Sache im „Griff", und Sie bringen problemlos eine fehlerfreie Zeile nach der anderen auf den Bildschirm.

Es gibt einige spezielle Betriebsarten, etwa zum Lesen von Schrift minderer Qualität. Zur Anwahl wird der Abtaster über einen entsprechenden Befehlstext geführt – vier solcher Kommandos stehen schon oben auf dem Tablett, und die übrigen sind der Anleitung zu entnehmen. Jeder Befehl wird durch zwei voranstehende Quadrate gekennzeichnet, damit er als solcher erkannt und nicht als Textwort eingelesen wird.

Schriftart wählen

Mit dem Kommando TYPEFACE wählt man die Schrifttype. Der Befehl NUMERIC löscht alle Buchstaben aus dem Zeichenvorrat, so daß der Omni-reader nur Zahlen akzeptiert (wodurch er schneller arbeiten kann). Diese Betriebsart ist besonders nützlich, weil beim Abschreiben von Zahlenwerten gewöhnlich die meisten Fehler auftreten.

Bei mangelhafter Schriftqualität können Sie mit POORCOPY die Lesegeschwindigkeit herabsetzen, um eine sorgfältigere Abtastung der Zeichen zu ermöglichen. Das ist vor allem beim Lesen von Texten wichtig, die mit Gewebefarbband geschrieben wurden und daher keine besonders tiefe Schwärzung aufweisen.

Die Treibersoftware für den Omni-reader liegt im Arbeitsspeicher innerhalb des Systembereichs; mit gleichzeitig geladenen Anwendungsprogrammen – etwa einem Textverarbeitungspaket wie WordStar oder einem Tabellenkalkulationsprogramm – kann die gelesene Information vom Anwender direkt weiter bearbeitet werden.

Im Gerätepreis (entspricht etwa dem eines PC) sind außer Software und Schnittstellenkabel eine halbtägige Schulung und ein Jahr kostenlose Wartung des Gerätes am Standort enthalten. Das ist nicht gerade billig; wenn im Büro aber viele maschinengeschriebenen Texte einzugeben sind, ist die Anschaffung des Omni-reader langfristig gesehen sicher kostengünstiger als die Einstellung von Schreibkräften fürs Abtippen.

Es bleibt jedoch abzuwarten, inwieweit sich Geräte dieser Art auf dem Markt durchsetzen können, und ob sich tatsächlich ein Kostenvorteil herauskristallisieren wird.

Pädagogische Simulationen und Abenteuerspiele erlauben Kindern, Welten zu erforschen, die sonst unerreichbar wären. Situationen, die den Raum Klassenzimmer sprengen. Sie sind keine Zeitvergeudung, sondern bewirken eine Beschleunigung und Ergänzung des Lernvorgangs.

Reise in andere Welten

Wir untersuchen das enorme Lehrpotential, das mit der Entwicklung von Simulationen und Abenteuer-Programmen zur Verfügung steht und Schülern die Türen zu sonst nicht erreichbaren Welten öffnet.

Die Möglichkeit spielend zu lernen, führte unter anderem dazu, den Glauben in den Computer als Schul(ungs)mittel zu intensivieren. Computersimulationen und Abenteuerprogramme weisen ein beachtliches Potential an pädagogischen Alternativen auf. Simulationen z. B. schaffen Lernwelten, die normalerweise für das Kind unerreichbar sind.

„Ballon-Fahren", ein von Hill McGibbon für den Spectrum und C 64 entwickeltes Programm, gibt Kindern die Möglichkeit, spielend mit etwas umzugehen, was sonst nicht möglich wäre. Tankuhr, Höhenmesser, Thermometer und eine Sink- und Steiganzeige am unteren Bildschirmrand reagieren in dem Umfang, wie der Brenner ein- und ausgeschaltet wird und das Gasventil geöffnet oder geschlossen wird. Sinkt der Ballon, ist der Erdboden sichtbar. Sollte Treibstoff erforderlich sein, gibt es verschiedene Landeplätze, an denen der Ballon zur Erde gebracht werden muß.

Neben den Argumenten Kostengünstigkeit und Sicherheit bieten Simulationen andere erzieherische Vorteile. Auf einem Computer durchgeführt erweist sich der Aufbau von Instrumenten für ein Experiment oder eine Übung als überflüssig. Wissenschaftliche Experimente, die Tage in Laboratorien in An-

spruch nehmen, sind in Sekundenschnelle auf dem Bildschirm darstellbar. Ein Computermodell vereinfacht Systeme, die für Schüler zu schwer zu verstehen sind. Grafiken können geeignete Funktionen mit repräsentativen Daten wichtiger Schritte darstellen. Da die Simulation unter verschiedenen Bedingungen wiederholbar ist, können die Schüler quasi unbegrenzt experimentieren. Der Ablauf des Experiments und die implementierten variablen Faktoren, die alle vom Schüler bestimmt werden können, erlauben ein größeres Verständnis der Resultate und Folgerungen. Da die Simulation als Ganzes erfolgt, wird eine umfassende Perspektive vermittelt.

Natürlich gibt es auch hier Grenzen. Da der Computer nur eine gewisse, vorbestimmte Menge an Variablen speichern kann, beinhaltet die Simulation die Gefahr einer starken Vereinfachung des behandelten Stoffes. Eine Simulation kann kein vollwertiger Ersatz für die Wirklichkeit sein. Ihre Stärke liegt in der Anregung, sich mit weiteren, verwandten Themen zu beschäftigen, nicht in der Darstellung vorprogrammierter Antworten oder einfacher Erweiterung und Übertragung von Techniken des „Programmierten Lernens". Es gibt z. B. ein Apple-Programm, in dem Glühbirnen und Batterien auf unterschiedliche Weise miteinander verbunden werden können. Offensichtlich wurde dieses Programm für eine Schule geschrieben, die sich unerklärlicherweise einen teuren Computer leisten kann, aber keinen Zugang zu Batterien und Glühbirnen hat. Die Kosten für eine 1,5 Volt-Batterie und der mögliche „Stromschlag" vermögen die Anschaffung eines solchen Programms aus Sicherheitsgründen wohl kaum zu rechtfertigen.

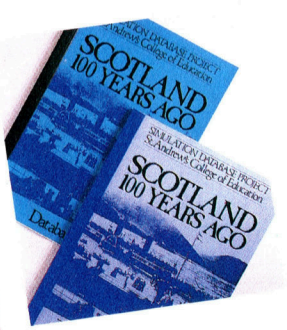
Rollenspiele wie „Dungeons and Dragons" waren nicht nur Verkaufsschlager, sondern haben auch das Bewußtsein für Rollenspiele in Therapie und Ausbildung geweckt. Interaktion macht den Computer zum idealen Medium für Rollenspiele. Seine grafischen und akustischen Möglichkeiten erlauben die Darstellung einer historischen oder imaginären Welt. Pädagogen haben diese Möglichkeiten erkannt und nutzen sie zur Schaffung von „Mikrowelten" auf dem Computer.

Nach ihrer Auffassung gibt es für ein Kind keinen besseren Weg, z. B. Geschichte zu lernen, als in der Zeit zurückzureisen, Abenteuer zu erleben und Erfahrungen zu sammeln. Das Programm muß eine Datenbank enthalten, die während des Spielverlaufs zu Rate gezogen werden kann und, möglicherweise abhängig vom Spielstand, zugänglich ist.

Computer-Pädagogik

Das „Simulations-Datenbank-Projekt" wurde von Allan Martin am St. Andrews College of Education in Glasgow entwickelt, um „zu erforschen, auf welche Weise Computer nachhaltige Beiträge zur Grundschulausbildung leisten können, indem sie eine neue Art pädagogischer Stoffe auf Computerbasis schaffen – die Simulations-Datenbank". Die Simulation basiert auf einem Abenteuerspiel. Ein Kind betritt eine historische Welt, in der es bei der Reise mit verschiedenen Situationen und Zwischenfällen fertig werden muß. Text-Files in der Datenbank geben bei Bedarf Informationen über bestimmte Vorfälle. Die Programme wurden für die Verwendung in einem größeren Projekt entwickelt, in dem der Großteil des Lernens nichts mit dem Computer zu tun hat; er ist nur eines von vielen Hilfsmitteln.

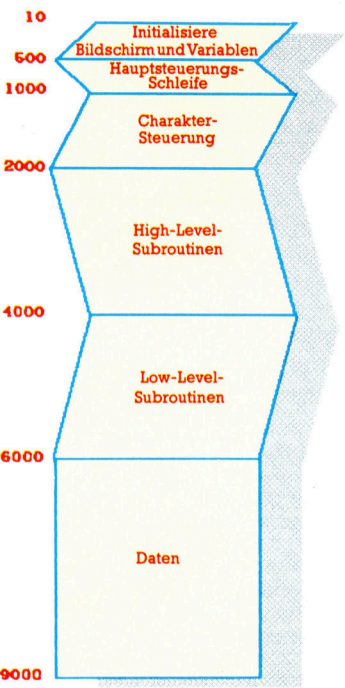
Eine zweite Datenbank, die Informationen über den entsprechenden Bereich beinhaltet, kann aus Büchern, Filmen oder anderen Medien bestehen. Um dieses Programm mit anderen Aktivitäten im Klassenzimmer verbinden zu können, ist ein Handbuch beigefügt, ferner gehören simulierte Briefe, Bilder und anderes Referenzmaterial dazu. Das in LOGO geschriebene Programm kann auf örtliche Bedingungen zugeschnitten werden.



Das Simulations-Datenbank-Projekt, am St. Andrews College Of Education entwickelt, nutzt den Computer ergänzend zu anderen Aktivitäten – nicht als Ersatz. Das Programm ist dreigeteilt: eine Textsammlung, die ergänzende Hintergrundinformationen liefert, sowie eine Datenbank, die auf weitere Quellen verweist, neben einem „Abenteuerspiel". Die Klasse wird in Gruppen geteilt, die die Rolle eines Charakters übernehmen und – in diesem Beispiel – im 19. Jahrhundert eine Reise durch Schottland unternehmen.

Szenenaufbau

Die Gestaltung eines Abenteuer-Programms mit interaktiven Charakteren setzt strukturiertes Programmieren voraus. Das erleichtert nicht nur die Fehlersuche, sondern erlaubt auch unkompliziertes Hinzufügen neuer Elemente.



Durch modulares Programmieren können wir, falls erforderlich, unser Spiel zu einem späteren Zeitpunkt modifizieren und den Charakter-Steuerer mit anderen Programmen kombinieren. Der meiste Speicherplatz wird für die Daten benötigt. Diese enthalten Botschaften, die auf dem Bildschirm dargestellt werden.

Das Programm, das wir als Umgebung für unsere interaktiven Charaktere schreiben, weist die in unserem Diagramm gezeigte Struktur auf. Wir arbeiten uns schrittweise durch das Programm und beginnen mit den Initialisierungs-Routinen. Um das zu vereinfachen, benutzen wir zur Datenspeicherung String-Arrays.

Zunächst benötigen wir drei Ortsbeschreibungen – eine für jeden Raum der berüchtigten Kneipe „Dog and Bucket". Zudem müssen wir wissen, wie die Räume miteinander verbunden sind, so daß, geht ein Darsteller von der Lounge nach Osten, wir genau wissen, daß er oder sie in den Salon gelangt ist. Der zweidimensionale Array enthält all diese zum Spiel notwendigen Informationen.

Die Darstellung der Daten für die Gegenstände in unserem Spiel hängt vornehmlich davon ab, welche Rolle sie spielen sollen. In diesem Spiel benötigen wir Gegenstände, die die Charaktere nehmen, ablegen oder auch aufeinander werfen können. Zudem ist zu berücksichtigen, ob die Gegenstände eßbar sind – und natürlich sollten reichlich Getränke vorhanden sein! Objekt-Datenstrukturen müssen deshalb folgende Fragen beantworten können: Wo ist es? Ist es eßbar? Und ist es trinkbar?

Zweidimensionale Arrays

Um das zu erreichen, bedienen wir uns des zweidimensionalen Arrays b$. Wir können die wichtigsten Elemente dieses Arrays jederzeit während des Programmablaufs überprüfen, um die drei zuvor genannten Fragen dann korrekt zu beantworten.

Nun konzentrieren wir uns darauf, wie Darsteller-Informationen zu speichern sind. Im „Dog und Bucket" sind folgende fröhliche Zecher zu Gast: Toby Belcher, Fiona Frappe, Steve Swigg, Sally Short, Rupert Beer und Molly Mixer. Dazu stellen wir noch den Barmann Fred. Letzterer ist nicht interaktiv. Er wurde lediglich in die Szenario-Beschreibung eingefügt, und der Character-Generator meldet gelegentlich, was er „tut".

Um zu entscheiden, welche Attribute für die Speicherung unserer Charaktere erforderlich sind, sollten wir uns noch einmal die Liste der für eine computergesteuerte „Person" erforderlichen Attribute vergegenwärtigen. Wichtig ist die Bewegung von einem Ort zum anderen. Also muß man die jeweilige Position eines Charakters verfolgen und Gegenstände manipulieren können.

Die anderen wichtigen Attribute eines Darstellers sind die Kommunikation mit dem Spieler und das Erkennen seiner Umgebung. Die erste Forderung braucht nicht in der Charakter-Steuerungsroutine implementiert zu sein. Um das zu verstehen, müssen Sie sich nur einmal vergegenwärtigen, was geschieht, wenn man während eines Spiels einen Befehl gibt. Gibt man beispielsweise „NIMM DOLCH" ein, und der Dolch ist vorhanden, wird das Pro-

Objekt-Tafel

Objekt-Nr.	Beschreibung	Ausgangsort	Eßbar?	Trinkbar?
1	Ein Glas Bier	2	N	J
2	Eine leere Dose Katzenfutter	3	N	N
3	Ein „Dog and Bukket"-Plätzchen	1	J	N
4	Ein Barhocker	2	N	N
5	Ein Aschenbecher	1	N	N
6	Ein Schinkenbrot	2	J	N
7	Ein Magenbitter	0	N	J
8	Ein Créme de Minte	0	N	J
9	Ein Whisky mit Wasser	0	N	J
10	Ein Wodka pur	2	N	J
11	Ein Pils	0	N	J
12	Ein Gin mit Ginger Ale	0	N	J

Ausgangspositionen
Im „Dog and Bucket" gibt es zwölf verschiedene Gegenstände. Dabei ist zu beachten, daß die Dinge 7 bis 12 im „Besitz" von Darstellern (siehe Darsteller-Tafel) sind. Deshalb ist ihre Ausgangsposition mit Ausnahme von 10 immer 0. Bei Spielbeginn jedoch hat Sally Short ihren Drink (10) abgestellt. Er befindet sich nun im Salon.

gramm auf die die Position des Dolchs enthaltende Variable zurückgreifen und sie entsprechend ändern. Zugleich wird das Inventar des Spielers um den Dolch ergänzt.

Der einfachste Weg, um sicherzustellen, daß Darsteller und Spieler gleiches tun, ist offensichtlich: Man bringt sich selbst als Darsteller ins Spiel. Dazu ist lediglich die Definierung eines Darstellers erforderlich, der mit „Du" angesprochen wird. Jedesmal, wenn diese Darsteller-Routine aufgerufen wird, sind Sie ins Leben der fiktiven Gefährten völlig lebensecht einbezogen.

Darsteller überprüfen

Kommunikation mit dem Spieler ist als Prinzip nicht schwer zu begreifen und recht leicht ins Programm einzufügen, aber die Wahrnehmung der Umgebung – die wichtigste Vorbedingung für einen interaktiven Charakter – kann sich als recht schwer erweisen. In gewissem Maße kann dieses Merkmal einfach in unseren Charakter-Steuerer auf der Basis gelisteter Daten erfolgen. Die Routine kann überprüfen, welche Gegenstände vorhanden sind, sie manipulieren oder sogar intelligente Anmerkungen machen, die der Darsteller „spricht". Sie kann ebenfalls die jeweilige Position eines Darstellers überprüfen und vielleicht diesen dazu veranlassen, einen entsprechenden Hinweis dazu zu geben. Entscheidend ist jedoch, daß die anderen Darsteller wahrgenommen werden können. Zudem muß der computergesteuerten „Person" eine Art „Kontinuität" gegeben werden. Mit anderen Worten: Die Darsteller müssen mit einer Art fortlaufender Geschichte ausgestattet werden.

Um das zu erreichen, führen wir zwei weitere Attribute ein, die für jeden Darsteller gespeichert werden müssen – den letzten Befehls-Code (LBC) und den letzten Charakter-Code (LCC). Diese zeigen die letzten vom Darsteller durchgeführten (aktiven oder passiven) Aktionen an. Dies kann in unseren Darsteller-Data-Array integriert werden, und als Beispiel für seine Anwendung mag folgender Fall dienen. Toby Belcher (Darsteller Nr. 1) gibt Fiona Frappe (Darsteller Nr. 2) ein Plätzchen. Fionas „LBC" wird nun so gesetzt, daß eine „erhalten"-Aktion angezeigt wird. Ihr „LCC" enthält die Nummer 1. Wird die Steuerungs-Routine beim nächsten Mal aufgerufen, wird sie die Daten überprüfen und aus dem in Fionas Inhaltsverzeichnis enthaltenen Objekt (dem Plätzchen) ableiten können, was geschehen ist. Die Routine kann nun entscheiden, ob Fiona „THANK YOU" sagen soll oder nicht. Tut sie es, wird ihr LBC auf „kommunizieren" gesetzt und ihr LCH wird noch immer 1 enthalten.

Unternimmt Fiona hingegen nach Tobys Handlung nichts, werden ihr LCC und ihr LBC auf 0 gesetzt, und die vorangegangenen Vorgänge werden nicht gewertet. Die für die Verwendung des LBC erforderlichen Daten sind in der Array-Tafel enthalten.

Sechs andere Attribute, die alle gleichermaßen wichtig sind, werden für unsere Darsteller gespeichert:

- **Stärke.** Dieses Attribut bestimmt die Fähigkeit des Darstellers, sich zu bewegen und zu kommunizieren. In diesem speziellen Fall wird ein Darsteller als bewußtlos betrachtet, wenn seine Stärke auf oder unter Null sinkt. Gewisse Handlungen (wie etwa zuviel trinken) werden die Stärke eines Darstellers verringern – andere dagegen können sie vergrößern.
- **Stimmung.** Hiermit wird festgelegt, wie die Darsteller miteinander umgehen. Wenn zum Beispiel ein Charakter einen bestimmten Gegenstand trägt, so muß entschieden werden, ob der Gegenstand abgelegt oder behalten

Darsteller-Tafel

	Ort 2	Inventar 3	Stärke 4	Stimmung 5	Gegenstand 6	Geschlecht 7	LCC 8	LBC 9	Steuerungs-Frequenz 10	Bewegungs-Frequenz 11
1. Tony Belcher	2	7	10	10	7	M	0	0	7	4
2. Fiona Frappe	1	8	30	10	8	F	0	0	3	5
3. Steve Swigg	1	9	8	10	9	M	0	0	4	6
4. Sally Short	2	0	20	10	10	F	0	0	5	5
5. Rupert Beer	2	11	10	6	11	M	0	0	4	6
6. Molly Mixer	1	12	15	6	12	F	0	0	5	5
7. Du	1	0	255	255	-	?	0	0	0	0

Ausgangssituation
Unsere Tafel zeigt die Ausgangswerte der verschiedenen Attribute der Darsteller in „Dog and Bucket". Alle Charaktere haben – mit Ausnahme von Sally Short – bei Spielbeginn einen Drink in ihrer Inventarliste.

Darsteller Nr. 7 stellt den Spieler dar und wurde eingefügt, um sicherzustellen, daß die anderen Charakter „Ihnen Aufmerksamkeit schenken". Da aber der Steuerungsfaktor von 7 gleich Null ist, wurde sichergestellt, daß Sie selbst „verantwortlich" für Ihre Handlung während des Spiels bleiben.

Array-Tafel

Array dimensioniert bis		Enthält Informationen betreffend
I$	3,5 (Spectrum-3,5,255)	Ortsbeschreibung (I$(n,1)) und die vier Ausgangs-Daten (I$(n,2...5))
b$	12,4 (Spectrum-12,4,30)	Objektbeschreibung (b$(n,1)) und Ort/eßbar/trinkbar (b$(n,2,...4))
c$	7,11 (Spectrum-7,11,15)	Darsteller-Namen (c$(n,1)) und ihre Attribute (c$(n,2,...11))

Der „letzte Befehls-Code" (LBC) zeigt an:

1) Die Zeit, wann die Darsteller-Steuerung zum letzten Mal aufgerufen wurde und der Darsteller einen Gegenstand von dem durch LCC angezeigten Charakter erhielt.
2) Der zuvor genannte Darsteller sagte etwas zu dem durch LCC angezeigten Darsteller.
3) Monolog. Der Darsteller hat eine Bemerkung gemacht. Dieser Code wird benutzt, um Charaktere daran zu hindern, ständig dieselbe oder ähnliche Bemerkungen zum Spielverlauf zu machen.
4) Genommen. Der Gegenstand im Inventar des Darstellers wurde bei der letzten Darsteller-Steuerung aufgenommen.
5) Getroffen. Der Charakter wurde von einem Objekt getroffen, das ein anderer Darsteller geworfen hat.
6) Essen. Der Darsteller ißt einen in seinem Inventar befindlichen Gegenstand.
7) Eingetreten. Der Darsteller hat soeben seine derzeitige Position bezogen.

Speicher-Muster
Die drei grundlegenden Arrays (I$, b$ und c$) enthalten alle Daten der Orte, Gegenstände und Darsteller. Dabei ist zu beachten, daß keine Nullwerte verwendet wurden. String-Arrays werden verwendet, um sowohl alphanumerische als auch numerische Daten zu speichern (STR$ und VAL$). Die LBCs werden benötigt, um der Handlung eine gewisse Kontinuität bzw. „Geschichte" zu verleihen. Sie sind in c$ (n,9) enthalten – siehe Darsteller-Tafel.

Orts-Tafel

Ort	Beschreibung	N	S	O	W
1)	Sie befinden sich in der Lounge des „Dog and Bucket". Mehrere zwielichtige Typen sitzen in einer Ecke und spielen Domino. Hinter dem Tresen steht der Barmann Fred. Eine Tür führt nach Osten.	0	0	2	o
2)	Das ist der Salon des „Dog and Bucket", der – entsprechend renoviert, einen prächtigen Eindruck machen könnte. Es scheint, als sei auf dem Fußboden schon reichlich Bier verschüttet worden. Türen führen nach Westen und Süden.	0	3	0	1
3)	Oje! Hier ist die Küche, in der die berühmten „Dog and Bucket"-Pasteten für die Gäste vorbereitet werden. Ihnen fallen die zahlreichen leeren Katzenfutter-Dosen auf. Das ist merkwürdig, weil es hier keine Katzen gibt.	2	0	0	0

Ausgang, Bühne, West
Die drei Orte haben jeweils eine eigene Beschreibung zusammen mit vier Ausgangsdaten. Der Ausgangscode enthält die Zahl des Bestimmungsortes. Bewegt man sich zum Beispiel aus der Küche Richtung Norden, gelangt man nach Ort Nummer zwei, dem Salon. Eine Ortsnummer mit dem Wert 0 zeigt an, daß eine Bewegung in diese Richtung nicht möglich ist.

werden soll. Liegt die Stimmung des Darstellers niedrig (sagen wir nahe Null), kann die Routine anweisen, daß der Gegenstand auf jemanden geworfen wird. Zu den Faktoren, die die Stimmung bei „Dog and Bucket" beeinflussen, gehört auch das Umstoßen eines Drinks.

● **Gegenstand.** Jeder Darsteller des Programms hat sein oder ihr eigenes „Objekt". In diesem speziellen Fall handelt es sich um Drinks, die Auswirkungen auf das Verhalten der Darsteller haben.

● **Geschlecht.** Hiermit wird festgelegt, ob eine Person männlich oder weiblich ist. Diese Angabe wird beispielsweise benötigt, um die richtigen Pronomia bei Mitteilungen für die Personen festzulegen.

● **Steuerungs-Frequenz (HF).** Sie zeigt die Zeit an, die verstreichen muß, bevor der Darsteller von der Charakter-Steuerungs-Routine geprüft wird. Ein Charakter mit dem HF 5 wird jeweils nach dem fünften Aufruf der Routine gesteuert. Ein HF von 0 dagegen würde bedeuten, daß der Darsteller überhaupt nicht aufgerufen wird. Gemeinsam mit der Bewegungsfrequenz kann HF zum „Einfrieren" des Darstellers benützt werden (HF auf 0). So kann man die Handlungsweise verlangsamen (HF verkleinern) oder sie aktiver werden lassen (HF vergrößern).

● **Bewegungs-Frequenz.** Sollen bestimmte Darsteller im Hinblick auf ihre Gefährten und die Umgebung aktiver sein, aber gleichzeitig an einer bestimmten Stelle bleiben oder sich nicht zu schnell bewegen, so ist das die Aufgabe dieser Routine. Sie zeigt an, wie oft der Darsteller bewegt werden soll.

Unser Karte zeigt die individuellen Werte der Darsteller-Attribute.

Die Memory Map des Acorn B

In dieser Folge untersuchen wir, wie der Acorn B seinen Speicher einsetzt und wie ein Betriebssystem mit Vektoren arbeitet.

Viele Speicherbereiche des Acorn B führen mehrere Funktionen aus. Ein typisches Beispiel ist die Speicherseite &9 (ein RAM-Bereich zwischen &900 und &9FF), die zu unterschiedlichen Zeiten als RS232-Ausgabebuffer, Cassettenausgabebuffer oder Arbeitsbereich für Sprache oder Tongenerierung dient.

Der Block von &0E00 bis &1900 ist ein weiteres Beispiel. Auf einem Acorn B mit Cassettenrecorder als Dateisystem steht dieser Speicherbereich für BASIC-Programme zur Verfügung. Beim Anschluß eines Diskettensystems dient er jedoch als Arbeitsbereich und verringert dadurch den Platz für BASIC-Programme um über 2,5 KByte. In den Modi 0 und 2 ist das besonders lästig, da dort der Speicher bereits eingeschränkt ist.

Auch der Block zwischen &8000 und &BFFF wird mehrfach genutzt. Hier liegt der Einsatzbereich für das DFS-ROM, das Textsystem oder ein ROM mit Dienstroutinen, aber auch der vom ROM des BASIC-Interpreters angesprochene Speicherbereich. Die „Paged"-ROMs lassen sich nur einzeln aktivieren. Dabei wählt das Betriebssystem das ROM und schaltet es ein. Normalerweise ist das BASIC-ROM mit dem BASIC-Interpreter aktiv, so daß BASIC-Programme sofort eingegeben und ausgeführt werden. Bei einem DFS-Befehl schaltet das OS jedoch das DFS-ROM ein, führt den Befehl aus und aktiviert wieder das BASIC-ROM. Während der gesamten Ausführung des DFS-Befehls wird das BASIC-ROM ignoriert. Andere ROMs, beispielsweise das Telesoftware Dateisystem oder der Textchip Wordwise, belegen ebenfalls diesen Speicherbereich und schalten sich auf Anforderung ein oder aus. Ohne das „Paging" würde nur ein sehr kleiner Arbeitsspeicher zur Verfügung stehen.

Nachdem das Betriebssystem seinen Speicher belegt hat, steht uns der Rest zur Verfügung. Die BASIC-Variable PAGE enthält die Startadresse des Programmbereichs für BASIC-Texte, während HIMEM auf den Anfang des Bildschirmspeichers zeigt. Der Platz zwischen diesen beiden Punkten steht für BASIC-Programme, Variablen und Maschinencodeprogramme zur Verfügung. Der folgende Befehl zeigt die Adressen und die Größe des zur Verfügung stehenden Speichers an:

```
PRINT PAGE, HIMEM (HIMEM-PAGE)
"FREIE BYTES"
```

Die Tabelle zeigt eine kurze Beschreibung der verschiedenen Speicherbereiche des Acorn B: Für die meisten mit BASIC-Programmen eingesetzten Routinen eignet sich der Bereich für BASIC-Programmtexte am besten.

Speicherbereiche, die für Maschinencodeprogramme geeignet sind	Dateisystem	
	Cass.	Disk.
&0D00 — &0DFF Seite &D – nur wenn kein „Paged ROM" eingesetzt ist	(✓)	X
&0B00 — &0CFF Seite &B und &C – nur wenn keine vom Anwender definierten Zeichen oder Funktionstasten eingesetzt werden	(✓)	(✓)
&0A00 — &0AFF Seite &A – wenn die serielle Schnittstelle nicht angeschlossen ist	(✓)	(✓)
Mit DIM im BASIC-Programmbereich reservierter Platz		

✓ = OK, (✓) = OK mit Einschränkung, X ungenutzt

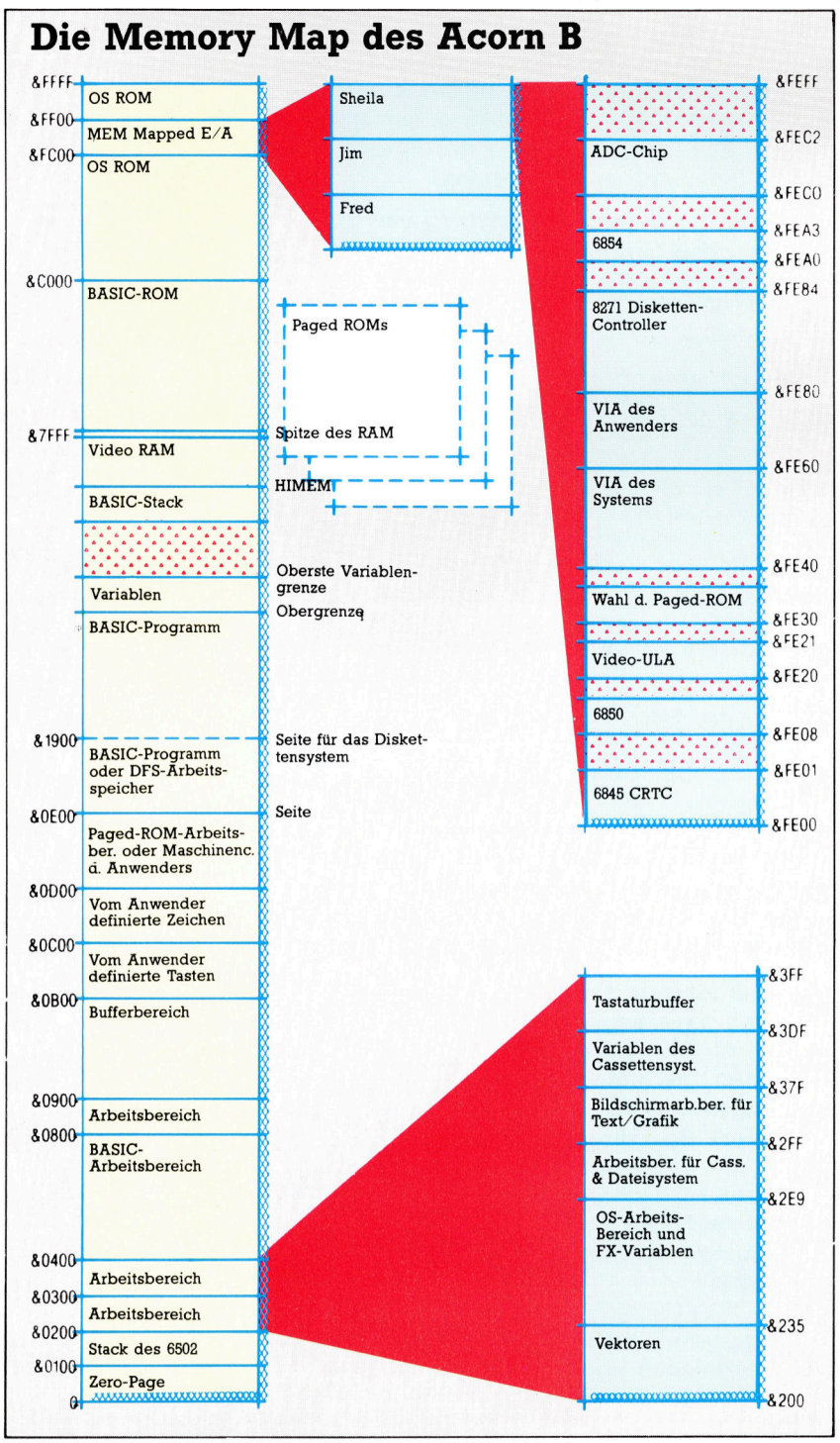
Maschinencode braucht für seine Abläufe normalerweise nur wenig Speicherplatz. Die meisten Maschinencodeprogramme des 6502 setzen dafür die Seite 0 (Zero-Page) ein, da die indizierte Adressierung nur mit dieser Seite funktioniert. Obwohl der Acorn B die Zero-Page oft für sein OS einsetzt, wurden einige Bytes für die Maschinencodeprogrammierung reserviert. Eine Liste mit den Funktionen der einzelnen OS-Speicherstellen nutzt dem Programmierer nur wenig. Diese Adressen sollten außerdem nicht direkt angesprochen werden, wenn das Programm auch auf anderen Betriebssystemversionen laufen soll.

Es gibt zwei Hauptmethoden, die OS-Routinen des Acorn B einzusetzen: OSBYTE und OSWORD.

- Mit OSBYTE können Sie die Abläufe vieler Betriebssystemroutinen beeinflussen, indem Sie Steuercodes und Parameter in die A-, X- und Y-Register des 6502 laden. Von BASIC aus läßt sich OSBYTE über den Befehl *FX anspre-

chen. Auf *FX müssen dabei zwei oder drei Zahlen folgen: Die erste Zahl gibt den Steuercode an, der an das Register A übergeben wird, die zweite Zahl geht in das X-Register und die dritte in Y. Im Maschinencode wird OSBYTE über die Adresse &FFF4 angesprochen. Die folgenden beiden Versionen haben die gleiche Funktion:

BASIC	Assemblersprache
*FX4,1	LDA #4 LDX #1 JSR &FFF4



Der Wert des A-Registers definiert exakt die Aufgabe, die ein bestimmter Aufruf von OSBYTE ausführt. Unser Beispiel beeinflußt die Cursortasten. Der an X übergebene Parameter bestimmt, ob die Cursortasten ihre normale Editierfunktion beibehalten oder einen ASCII-Wert liefern.

Leider lassen sich nicht alle OS-Routinen mit OSBYTE ansprechen, da das Modul nur wenige Parameter aufnehmen kann. Wenn Sie den Inhalt des A-Registers (das dem OS mitteilt, welche Routine von OSBYTE eingesetzt werden soll) ignorieren, können Sie nur zwei Parameter an X und Y übergeben. Für längere Parameterlisten müssen Sie dann entsprechend OSWORD einsetzen.

● Mit OSWORD lassen sich komplexe Vorgänge wie Tongeneration, Diskettenzugriffe etc. steuern. Hier zeigt sich der Unterschied zwischen den beiden Modulen: OSBYTE gibt an, wie das OS bestimmte Aufgaben ausführt, während sich mit OSWORD Aufgaben tatsächlich ausführen lassen. OSWORD erhält seine Parameter von einem „Parameterblock", auf den die Routine kurz nach dem Einsprung mit den X- und Y-Registern des 6502 zeigt. Dieser Parameterblock liegt im RAM. Seine Größe und sein Aufbau hängen von der aufgerufenen Funktion ab. Der Inhalt des A-Registers bestimmt, welche Funktion von OSWORD vom OS ausgeführt wird. Nach der Vorbereitung der Register und des Parameterblocks wird OSWORD über die Adresse &FFF1 angesprochen. OSBYTE und OSWORD sind die beiden Hauptmethoden, das OS einzusetzen.

Doch auch andere OS-Routinen lassen sich von BASIC aus mit einem Stern gefolgt von einem Befehl ansprechen. Der * veranlaßt, daß der Befehl nicht über den BASIC-Interpreter läuft, sondern an eine OS-Routine mit dem Namen OSCLI (Operating System Command Line Interpreter) übergeben wird. Diese Routine nimmt den Befehl entgegen und ruft die entsprechenden OS-Routinen auf. In der Tabelle haben wir die „Sternbefehle" des Acorn B aufgeführt. Eingaben mit Fehlern oder Befehle ohne die richtige Parameterzahl erzeugen die Meldung „Bad Command".

Ganzzahlvariablen beachten

Außer mit dem „Command Line Interpreter" (CLI) können Befehle auch direkt an das Betriebssystem übergeben werden. OSCLI hat zwei Einsatzmöglichkeiten: Sie können die Befehle (siehe Tabelle) vom Maschinencode aus oder als BASIC-Stringvariablen an CLI übergeben. Beachten Sie, daß die Ganzzahlvariablen A%, X% und Y% ihre Werte direkt in die Register A, X und Y laden. X% (oder das X-Register) und Y% (oder das Y-Register) zeigen im Speicher auf die Position des Strings, der als Sternbefehl interpretiert und von OSCLI ausgeführt werden soll.

BASIC-Version

```
10 DIM C 100
20 OSCLI=&FFF7
30 INPUT"ENTER COMMAND
",A$
40 $C=A$
50 X%=C MOD 256
60 Y%=C DIV 256
70 CALL OSCLI
80 GOTO 30
```

BASIC + Assemblerversion

```
 10 DIM C 100
 20 OSCLI=&FFF7
 30 FOR I%=0 TO 2 STEP 2
 40 P%=&C00
 50 [ OPT I%
 60 .code    LDX #C MOD 256
 70          LDY #C DIV 256
 80          JSR OSCLI
 90          RTS
100 ]:NEXT I%
110 INPUT"ENTER COMMAND ",A$
120 $C=A$
130 CALL code
140 GOTO 110
```

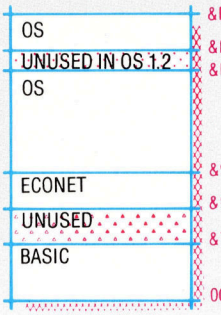
Dieses Programm erzeugt einen Prompt, wenn der Anwender einen Sternbefehl und Return eingibt. Mit dem DIM-Befehl in Zeile 10 reserviert das Programm 100 Bytes im Speicherbereich für BASIC-Variablen und initialisiert die Variable C mit der Anfangsadresse dieses Blocks. In die 100 Bytes können entweder Maschinencodeprogramme oder Daten gelegt werden. Der Befehl $C=A$ setzt die Bytes des in A$ gespeicherten Befehlsstrings in diesen Block von 100 Bytes (vom ersten reservierten Byte an aufwärts). In beiden Programmen werden die Register X und Y (die Variablen X% und Y%) geladen und dann die Steuerung an OSCLI übergeben. Damit wird der Befehlsstring ausgeführt.

Keine Programmunterbrechung

Dieses Modul läßt sich mit menügesteuerten Anwenderprogrammen einsetzen, wo es beispielsweise Disketten oder Cassetteninhalte ansprechen kann, ohne das laufende Programm zu unterbrechen. Dabei wird einfach der gewünschte Befehl in die Stringvariable geladen und zur Ausführung an OSCLI übergeben. Die Eingabe von *A$ funktioniert jedoch nicht. Das OS versucht dann einen Befehl namens *A$ zu finden, der nicht existiert.

Auch numerische Variablen lassen sich an Sternbefehle übergeben, wenn sie mit der Funktion STR$ zuvor in Strings umgewandelt wurden. Normalerweise nimmt das CLI keine Variablennamen an, sondern gibt die Fehlermeldung „Bad Command" aus.

Jeder vom OS nicht erkannte Sternbefehl wird an die Paged ROMs übergeben. Dabei wird jedes ROM „gefragt", ob es den Befehl identifizieren kann. Ist dies der Fall, wird er ausgeführt. Befehle, die auf diese Weise nicht zur Ausführung kommen, werden jedoch nur dann als „Bad Commands" behandelt, wenn kein schnelles Dateisystem (beispielsweise ein Diskettenlaufwerk) angeschlossen ist. In diesem Fall wird auch untersucht, ob auf der Diskette eine Datei mit dem angegebenen Befehlsnamen (allerdings ohne *) vorhanden ist. Wenn ja, wird die Datei in die Maschine geladen und als Maschinencodeprogramm interpretiert. Wird dagegen keine Datei gefunden, erscheint hier ebenfalls die Fehlermeldung „Bad Command".



Die Zero-Page des Acorn B
Der 6502 besitzt zwar nur eingeschränkte Indiziermöglichkeiten, kann aber die Zero-Page als Array von zusätzlichen Indexregistern einsetzen. Obwohl fast alle dieser Speicherstellen vom Betriebssystem belegt werden, stehen dem Programmierer einige Bytes zur Verfügung.

Die *-Befehle

	Beschreibung und Kommentar
*HELP	Gibt die Versionsnummer des BASIC an. Kann auch Informationen über Paged ROMs liefern – z. B. *HELP DFS
*BASIC	Ruft das BASIC-System auf. Eine Variante des Befehls – z. B. *WORD – ruft ein Paged ROM auf (hier View).
*CODE	Nur in OS 1.2. Damit kann man neue
*LINE	Befehle in das OS einsetzen.
*KEY	Für die Programmierung der roten Funktionstasten
*MOTOR n	Steuert Relais des Bandmotors: n=0 schaltet das Relais ab, n=1 schaltet es an
*ROM, *TAPE, *DISK, *NET	Initialisiert das Dateisystem – *TAPE aktiviert das Bandsyst. mit 1200 Baud und *DISK das Diskettensyst.
*FX	Gibt dem Programmierer die Möglichkeit, die Werte der OS-Variablen und damit die Abläufe des OS zu steuern
*RUN, *OPT, *LOAD, *. *CAT, *SAVE	Dateisystembefehle, die in späteren Folgen untersucht werden
*SPOOL *EXEC	*SPOOL sendet Bildschirmausgaben an den Bildschirm und in eine Datei. *EXEC liest Daten aus einer Datei, als ob sie über die Tastatur eingegeben wären
*TV x,y	Beeinflußt die vertikale Anzeige des Bildschirms und der Bildschirmschnittstelle: x=0 löst keine Veränderung aus. x=1 bewegt die Bildschirmanzeige um eine Zeile nach unten, x=255 um eine Zeile nach oben. y=0 Interlace an, Y=1 Interlace aus.

Weitere Erklärungen dieser Befehle befinden sich im Handbuch des Acorn B.

Lichtreiter

Obwohl erfolgreiche kommerzielle Spiele zwecks Geschwindigkeit in Maschinensprache geschrieben werden, kann man auch in BASIC unterhaltsame Spiele programmieren. Das folgende Spiel braucht nur 35 Programmzeilen, ist jedoch trotzdem recht interessant. Zwei Spieler treten auf „Lichträdern" gegeneinander an.

Das Spiel heißt „Lichtreiter" und basiert auf einer Szene des Walt-Disney-Films „Tron". Thema ist ein Wettstreit zweier Gegner, der schnelle Reaktionen erfordert. Jeder verfügt über ein „Lichtrad", das nicht gestoppt werden kann. Das Rad kann bei Höchstgeschwindigkeit seine Fahrtrichtung um 90 Grad ändern. Die Reifen hinterlassen in ihrer Spur eine Wand aus Licht. Ziel des Spiels ist es, den Gegner gegen diese Wand zu treiben.

Das Spiel wurde für den ZX Spectrum geschrieben, der für seine geringe BASIC-Geschwindigkeit bekannt ist. Aufrufe von Unterroutinen und andere Strukturen wurden vermieden, da sie die Ausführungsgeschwindigkeit weiterhin reduzieren.

Zuerst wird das Spielfeld und die Punktanzeige entworfen. Ein wichtiger Punkt ist, daß die Grenze des Spielfeldes innerhalb des benutzbaren Bildschirmbereiches liegt. Dadurch wird gewährleistet, daß die Grafik bei einer Kollision mit einer Wand nicht für Sie unsichtbar außerhalb des Bildschirms stattfindet.

```
10 LET p=0: LET q=0
100 BORDER 0: PAPER 0: CLS
110 PRINT AT 0,1; INK 6; "Bike One= "; q
120 PRINT AT 0,19; INK 5; "Bike Two= "; p
130 INK 2
140 PLOT 8,8: DRAW 239,0: DRAW 0,159
150 DRAW -239,0: DRAW 0,-159
```

Das Spielfeld wird rot gezeichnet. Für die Räder haben wir die Farben gelb und blau gewählt. Die Variablen p und q beinhalten den Punktestand der beiden Wettstreiter.

Danach werden alle Variablen initialisiert. Die Routine für das Zeichnen des Motorrads ist einfach. Unter Verwendung von POINT wird überprüft, ob die aktuelle Position des Rades belegt ist. Trifft dies nicht zu, wird mit PLOT die neue Position belegt und dann die Tastatur abgefragt, ob eine Richtungsänderung eingegeben wurde. Andernfalls wird die Kollisions-Routine aufgerufen. Abschließend wird die Position in der momentanen Richtung um eins erhöht und der Vorgang erneut ausgeführt. Man braucht also vier Variablen: zwei für die aktuellen X- und Y-Koordinaten und zwei für die aktuelle Richtung entlang der X- und Y-Achse.

In diesem Spiel sollen sich jedoch zwei Motorräder gleichzeitig bewegen. Eine elegante Lösung wäre die Verwendung von vier Arrays mit je zwei Elementen, doch würde dies den Spielablauf verlangsamen, so daß wir acht separate Variablen benutzen müssen:

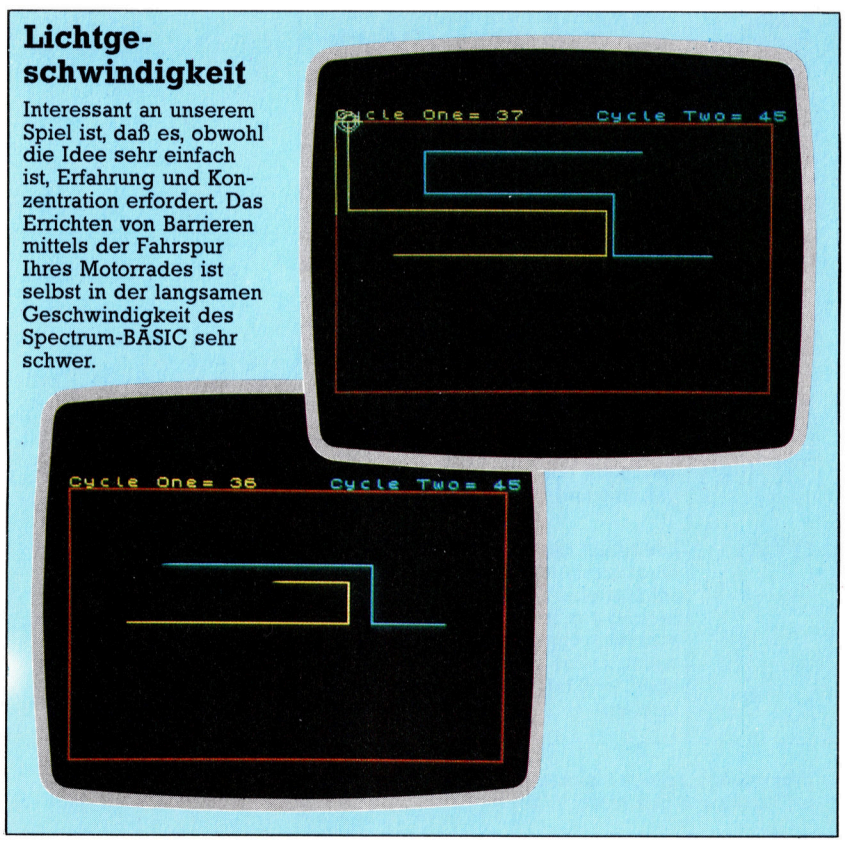
```
200 LET x=40: LET y=88
210 LET m=215: LET n=88
220 LET a=1: LET b=0
230 LET i=-1: LET j=0
```

Diese Zeilen setzen die Anfangspositionen der beiden Räder, so daß sie sich Pixel für Pixel aufeinander zubewegen. Die eigentlichen Routinen sind dann relativ einfach. Mit GOTO 700 wird in die Explosionsroutine verzweigt, die wir weiter unten erklären.

```
400 IF POINT (x,y)=1 THEN LET col=6: GOTO 700
410 IF POINT (m,n)=1 THEN LET col=5: LET x=m:
    LET y=n: GOTO 700
420 PLOT INK 6;x,y:PLOT INK5;m,n
```

Lichtgeschwindigkeit

Interessant an unserem Spiel ist, daß es, obwohl die Idee sehr einfach ist, Erfahrung und Konzentration erfordert. Das Errichten von Barrieren mittels der Fahrspur Ihres Motorrades ist selbst in der langsamen Geschwindigkeit des Spectrum-BASIC sehr schwer.



(Die Zeilen 500 bis 570 enthalten die Tastatur-Routine, die neue Werte für a, b, i und j setzt.)

```
600 LET x=x+a: LET y=y+b
610 LET m=m+i: LET n=n+j
620 GOTO 400
```

Der einzig verwirrende Punkt befindet sich in Zeile 410, wo die Variablen für Rad 1 auf die von Rad 2 gesetzt werden und eine neue Variable, col, benutzt wird. So kann dieselbe Routine für die Kollision verwendet werden, wobei x und y den Ort der Kollision angeben und col die Farbe setzt.

Für die Tastaturabfrage kann man, anstelle der relativ langsam arbeitenden IF. . .THEN-Anweisungen, IN-Befehle benutzen. Die Kontrolltasten für Auf- und Abbewegungen von Rad 1 sind Q und A. P und ENTER werden für Rad 2 gedrückt. Für Links/Rechtsbewegungen von Rad 1 dienen die Tasten X und C, sowie N und M für Rad 2.

```
500 IF IN 64510= 190 THEN LET a=0: LET b=1
510 IF IN 65022= 190 THEN LET a=0: LET b= -1
520 IF IN 65278= 187 THEN LET a= -1: LET b=0
530 IF IN 65278= 183 THEN LET a=1: LET b=0
540 IF IN 57342= 190 THEN LET i=0: LET j=1
550 IF IN 49150= 190 THEN LET i=0: LET j= -1
560 IF IN 32766= 187 THEN LET i=1: LET j=0
570 IF IN 32766= 183 THEN LET i= -1:LET j=0
```

Jetzt muß nur noch die Kollisions-Routine und die Aktualisierung des Punktestandes programmiert werden. Zur Kollision wurde eine Serie auseinanderlaufender konzentrischer Kreise mit Radien von vier, sechs und acht Pixeln gewählt:

```
700 FOR d=1 TO 3
710 CIRCLE BRIGHT 1; INK col; x,y,2+d*2
720 NEXT d
730 IF col=6 THEN LET p=p+1: GOTO 750
740 LET q=q+1
750 GOTO 100
```

Mit diesen Zeilen wird das Spiel beendet, wobei durch die letzte Anweisung zur Initialisierungs-Routine verzweigt wird. Um das Spiel anwenderfreundlicher zu machen, kann man eine Start-Routine integrieren:

```
300 PRINT AT 10,5; INK 7; "PRESS ANY KEY TO
    START"
310 IF INKEY$= "" THEN GOTO 310
320 PRINT AT 10,5; "                    "
```

Durch diese Zeilen entsteht eine Pause zwischen den einzelnen Runden. Der Computer wartet geduldig auf Ihren Tastendruck. Jetzt müssen Sie das Spiel nur noch auf Cassette speichern. Verwenden Sie den Befehl SAVE „LICHTREITER" LINE 10, so daß das Spiel nach Einladen automatisch startet.

Gegen den Rechner

In der Ein-Spieler-Version ist der Computer Spieler 1. Der Programmteil, der die Befehle für Spieler 1 abfragt, wird übergangen, und der im Programm enthaltene Algorithmus erlaubt Spieler 1, sich vorwärts zu bewegen, bis ein illegaler Zug gemacht wird oder Spieler 2 sich bewegt. Später wird der Zug gespiegelt bzw. dupliziert, und dann auf Korrektheit überprüft. Bei einem illegalen Zug sucht das Programm nach legalen Bewegungsmöglichkeiten. Kann keine gefunden werden, ist es aus, Wie folgt ändern:

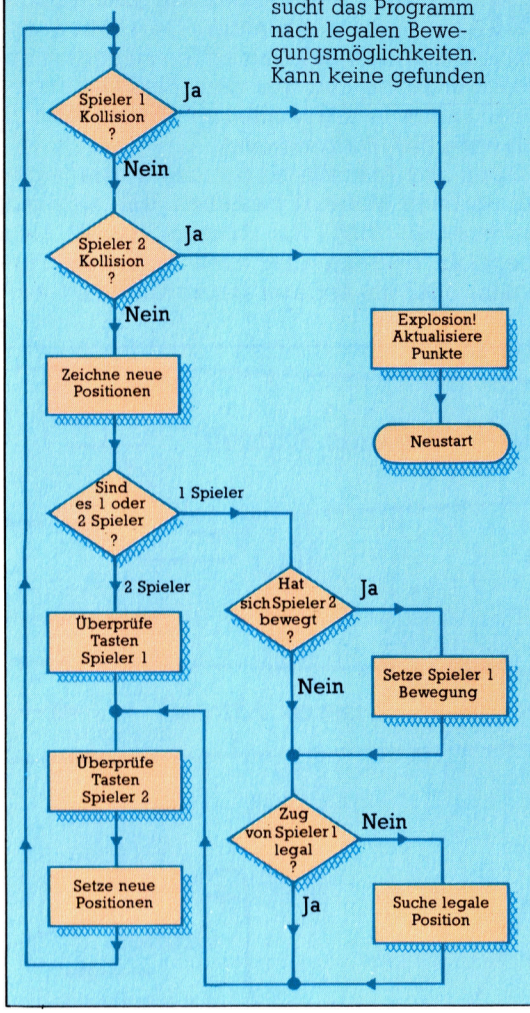
```
20 LET keybd=500:LET
   pt=-1:LET umoved
   =0: RANDOMIZE
260 PRINT AT 10,5;"Zahl
   der Spieler (1/2)?"
270 LET a$=INKEY$:IF
   a$<>"1" AND
   a$<>"2" THEN
   GOTO 270
280 IF INKEY$<>"2"
   THEN LET keybd=440
```

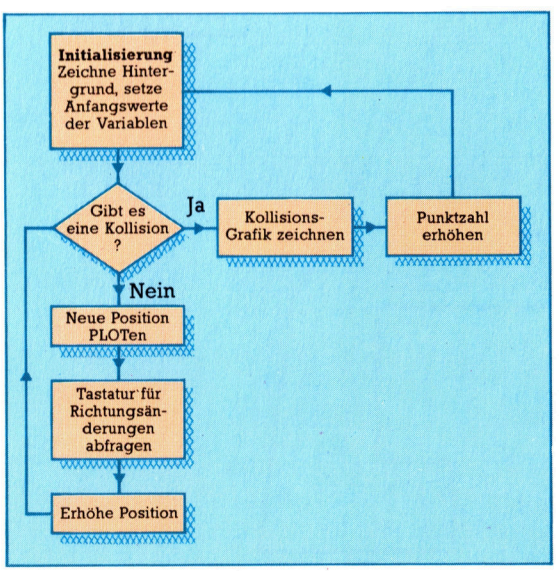
Diese Zeilen geben dem Spieler die Spiel-Auswahl. Entsprechend befindet sich zwischen den Zeilen 440 bis 460 die Routine für einen Spieler, und zwischen 500 und 570 die Version für zwei Spieler. Unser Vorschlag sieht wie folgt aus:

```
430 GOTO keybd
440 LET
   pt=SGN(RND-0.5):
   IF unmoved=pt THEN
   LET
   a=pt*j:LET
   b=pt*i:LET
   umoved=0
450 IF POINT
   (x+a,y+b)<>
   1 THEN GOTO 540
460 LET
   pt=SGN(RND-.5):
   LET a=a*pt:LET
   b=b*pt:
   LET d=a:LET
   a=b:LET
   b=d: IF POINT
   (x+a,y+b)
   <>1 THEN GOTO
   540
490 LET a=-a:LET
   b=-b:
   GOTO 540
540 IF IN 57342=190
   THEN LET i=0:LET
   j=1:LET
   umoved=1
```

Fügen Sie außerdem an das Ende der Zeilen 550 bis 570 jeweils:

```
: LET umoved = 1
```





Dieses vereinfachte Flußdiagramm zeigt die Programmstruktur für einen Spieler. Jeder Vorgang wird im Zwei-Spieler-Modus wiederholt.

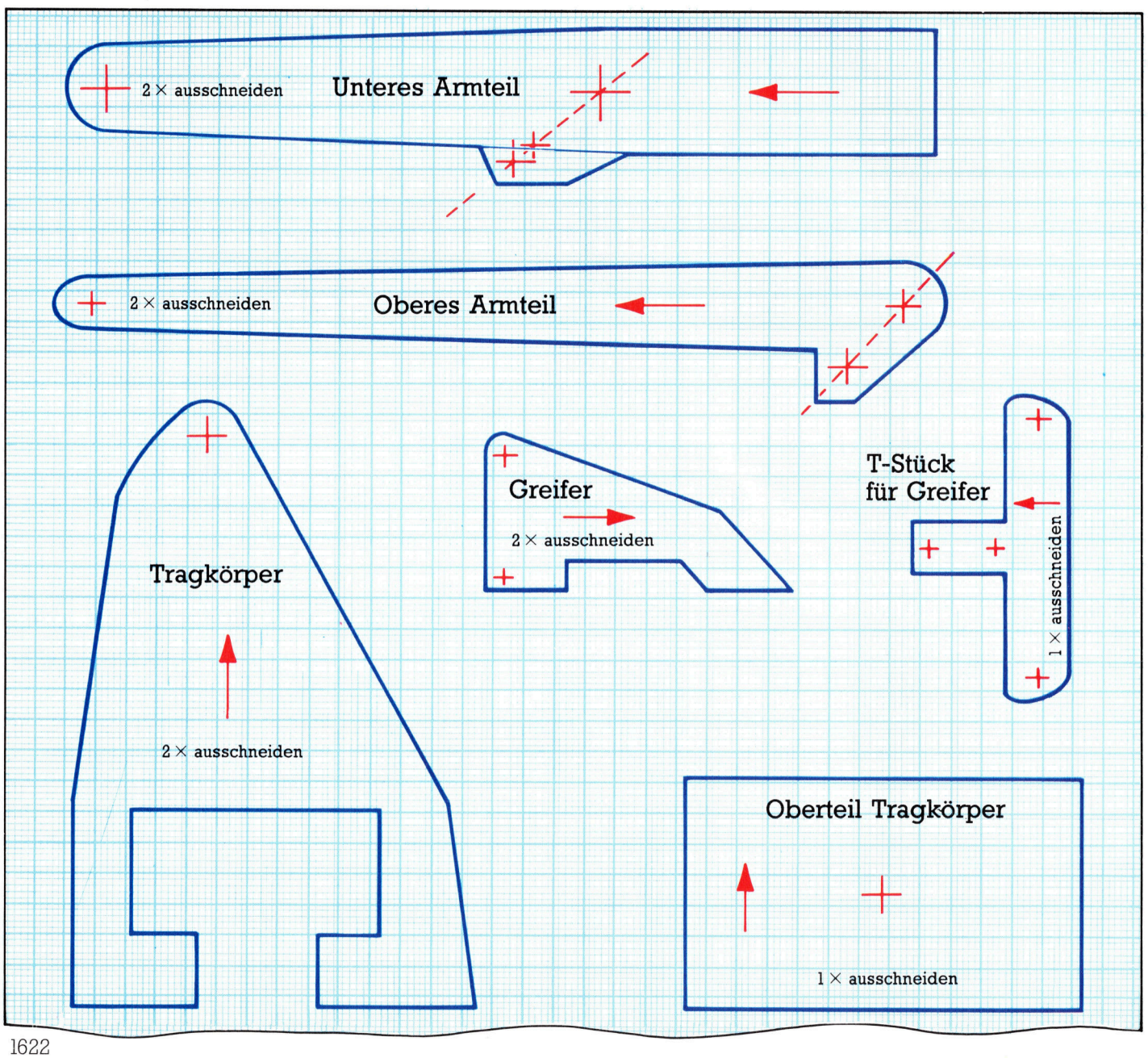
Muskeln für den Roboter

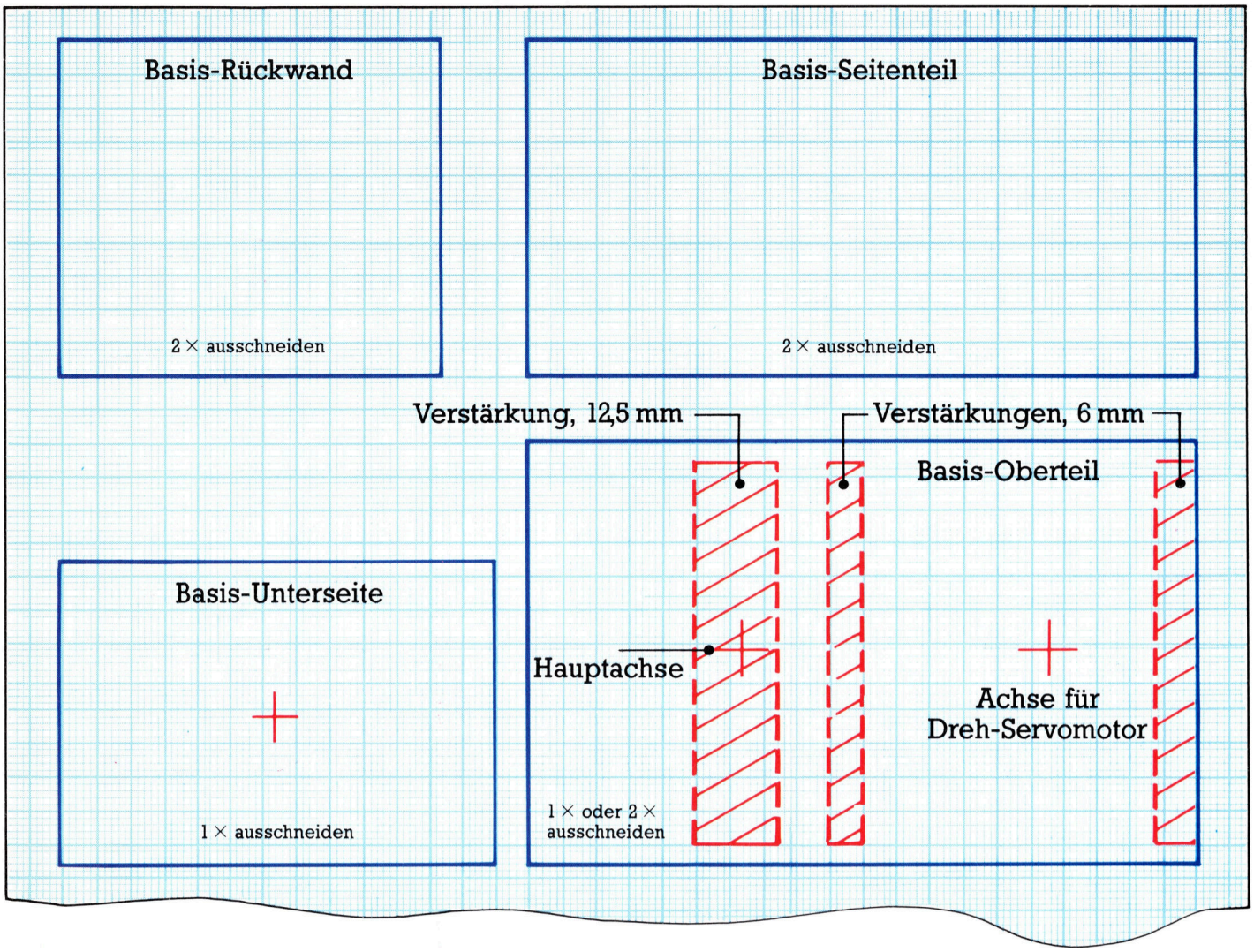
Für die Konstruktion des Robot-Arms finden Sie in diesem Kursabschnitt Schablonen, nach denen einzelne Elemente zugeschnitten werden.

Motor- und Getriebeteile für den Robot-Arm sind in Modellbau-Geschäften erhältlich, den Rest bekommen Sie sicherlich im Elektronikladen und aus dem Eisenwarengeschäft. Alles in allem müssen Sie mit Kosten von etwa 250 Mark rechnen.

Basis und Armteile sollten aus einem leichten, stabilen Material bestehen, das sich gut zuschneiden läßt. Am besten eignet sich Sperrholz, es kann aber auch Acrylglas (Plexiglas) oder ein anderer fester Kunststoff verwendet werden. Wenn Sie mit Sperrholz arbeiten, brauchen Sie eine 4 mm dicke Platte von 50 × 50 cm. Für den Zuschnitt haben wir maßstabsgetreue Schablonen abgedruckt. Neben dem Sperrholz sind auch noch drei 55 mm lange Holzleisten erforderlich (zwei 6 mm starke und eine mit einer Dicke von 12,5 mm), die zur Verstärkung dienen.

Für den Arm werden vier 5-Volt-Motoren gebraucht (z. B. Futaba S128). Zum Aufstecken auf die Motorachsen müssen Sie außerdem





Kunststoffscheiben (Ø 30 mm) haben. Soll Ihr Robot-Arm kräftiger sein, können Sie natürlich auch stärkere Motoren als den Futaba S128 verwenden.

Zur Stromversorgung der Motoren ist ein externes 5-Volt-Netzteil erforderlich. Zwar stehen auch am User Port 5-Volt-Pegel zur Verfügung, diese sind aber nur für die Versorgung von TTL-Schaltungen ausreichend: Maximal darf ein TTL-Ausgang 100 mA liefern, ein Servomotor unter voller Last benötigt aber mehr als doppelt so große Ströme. Ein Netzteil für 5 Volt/1 A Gleichstrom wäre zur Versorgung des Robot-Arms optimal. Etwas billiger geht es aber auch mit einem Batteriehalter für drei 1,5 Volt Monozellen. Die Gesamtspannung von 4,5 Volt reicht zum Antrieb des Systems aus.

Zum Befestigen der Buchsen für die Motoranschlüsse brauchen Sie ein Stück Lochplatine (Veroboard) mit 20 Bahnen à 9 Löcher. Zur Verbindung von Robot-Arm und Computer dient ein etwa zwei Meter langes, 20-poliges Flachkabel. Beim Acorn B wird ein 20-poliger IDC-Stecker verwendet, für den Commodore 64 übernimmt ein 24-poliger Platinenstecker den Anschluß am User Port. Die Verbindung mit dem Sinclair Spectrum wird über das bereits gebaute Interface hergestellt. Die Veränderungen am Interface behandeln wir in einem späteren Kursabschnitt.

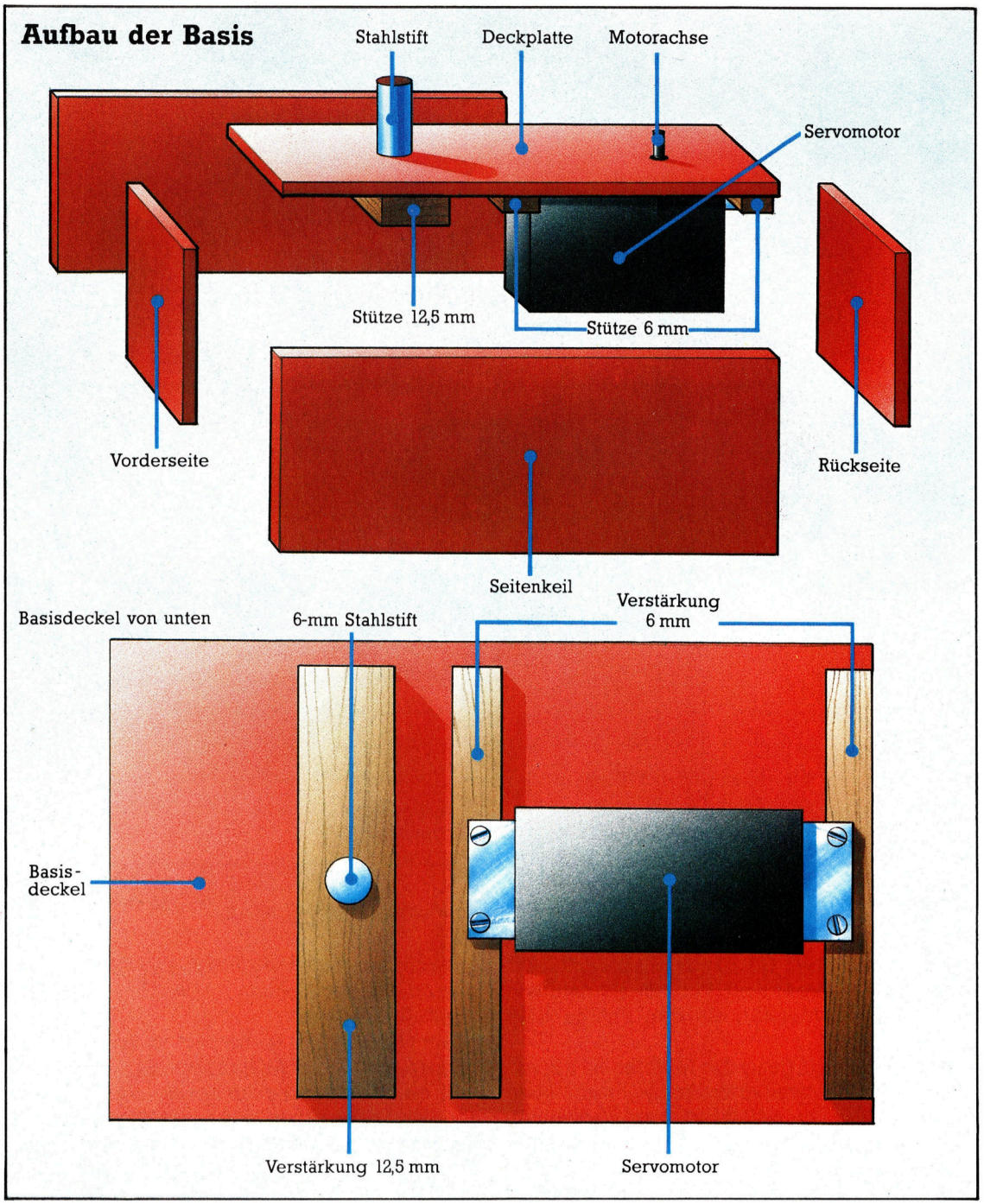
Als Gelenklager dienen sechs 75 mm lange Messingröhrchen (Abstandshalter), von denen drei einen Außendurchmesser von 4 mm, die anderen drei einen Innendurchmesser von 5 mm haben sollten. Die Röhrchen werden ineinandergeschoben und an den Enden abgeschnitten. Wichtig ist, daß die Röhrchen glatt und maßhaltig sind, damit sie ohne viel Spiel zusammenpassen, sich aber dennoch leicht ineinander drehen lassen.

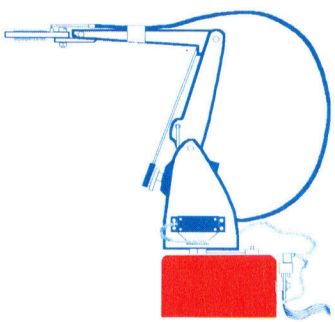
Der Tragkörper wird über ein Kugellager mit der Basis verbunden. Das Kugellager sollte gekröpft sein und einen Innendurchmesser von

Die Schablonen für die einzelnen Teile werden ausgeschnitten und auf die Sperrholzplatte geklebt. Zeichnen Sie um die Schablonen herum, und sägen Sie gemäß den Rißlinien aus. Denken Sie daran, daß einige Teile mehrfach gebraucht werden! Löcher an den Seitenteilen der Arme nur vorbohren, erst nach dem Zusammenbau auf das endgültige Maß bringen. Alle Teile sollten gründlich geschmirgelt werden, damit keine scharfen Kanten stehenbleiben.

Liste der Teile

- ○ Gewindeschrauben und -Muttern zur Montage der Greifklauen.
- ○ Messing-Abstandshalter (Röhrchen), Außendurchmesser 4 bzw. 5 mm Innendurchmesser
- ○ Holzleim- und Kunstharzleim
- ○ 16 Holzschrauben mit Gummi-Unterlegscheiben zum Befestigen der Motoren
- ○ 5 cm Schaumstoff für die Greifklauen
- ○ Isolierband und Lötzinn





6 mm haben. Solche Lager erhalten Sie im Modellbau-Geschäft. Als Achse für das Lager dient ein 25 mm langer, 6 mm starker Stahlstift, der fest mit der Basis verbunden und durch das Kugellager hindurchgesteckt wird.

Der Motor für den Greifer sitzt im Tragkörper des Arms. Mit den Greifklauen wird er durch einen 500 mm langen, flexiblen Bowdenzug verbunden, wie er bei Modellflugzeugen zur Bewegung der Klappen eingesetzt wird. Außerdem ist noch ein 50 mm langes Stück Klavierdraht für das Öffnen und Schließen der Greifklauen nötig. Dieser wird mit dünnem Stahldraht am Ende des Bowdenzuges befestigt. Den Arm selbst treiben Schubstangen an, die aus vier 150 mm-Stücken eines 2 mm starken, steifen Stahldrahtes hergestellt werden. Die Schubstangen können entweder direkt oder über kleine Kugellager mit den Motorscheiben verbunden werden.

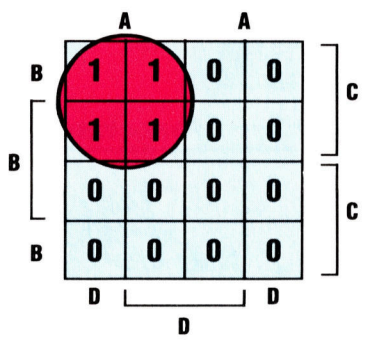
Aufbau der Basis

Zuerst wird die 12,5-mm-Leiste auf die Unterseite des Basisdeckels geklebt, danach kann das Loch für den 6-mm-Stahlstift gebohrt werden. Durch Deckel und Stütze bohren und festkleben. Danach die beiden anderen Verstärkungsleisten einleimen. Sie dienen zur Befestigung eines der beiden Servomotoren. Die Löcher im Deckel werden passend zur Motorachse gebohrt, der Motor selbst mit Schrauben und Gummiunterlegscheiben befestigt. Erst danach Vorder- und Rückwand ankleben. Das fertige Gehäuse mit Schleifpapier glätten.

Fachwörter von A bis Z

K = K
Das kleine „k" ist die Abkürzung für die Vorsilbe „Kilo", das heißt, den Faktor 1000 bei physikalisch-technischen Einheiten. In der Datenverarbeitung bezeichnet die Vorsilbe Kilo dagegen den Faktor $2^{10}=1024$ (mit einem großen „K" abgekürzt). 1 Kilobyte sind 1024 Byte, ünd 64 KByte sind $64 \times 1024=65\,536$ Byte. Bei Speicherkapazitäts-Angaben wird die Einheit Byte auch oft weggelassen – man spricht einfach von „64 K RAM".

Karnaugh Map = Karnaugh-Diagramm
Ein Karnaugh-Diagramm ist die grafische Darstellung einer Wahrheitstabelle (zweidimensional). So wie ein Venn-Diagramm Durchschnitt und Vereinigung von Mengen deutlich macht, illustriert das Karnaugh-Diagramm logische Verknüpfungen.



Komplizierte logische Ausdrücke lassen sich durch Umformung nach den Regeln der Booleschen Algebra vereinfachen, was aber mühsam und fehlerträchtig ist. Ein Karnaugh-Diagramm liefert dagegen ein unmittelbares Abbild der Verknüpfungen und erlaubt die Vereinfachung aus der Anschauung heraus. Nebenstehend ist der Ausdruck A AND B dargestellt.

Kernel = Kern
Der Kern eines Rechner-Betriebssystems beinhaltet die lebenswichtigen Funktionen. Wenn dieser nicht geladen ist oder nicht richtig arbeitet, läuft das System nicht; er stellt das Hauptsteuerprogramm dar. Der Kern sorgt durch unmittelbare Anweisungen an die CPU für die Aktivierung und Überwachung aller anderen Systemprogramme, die beim Rechnerbetrieb benutzt werden.

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.

Key = Taste
Das englische Wort „Key" ist in der Datenverarbeitung mehrdeutig: Wenn damit nicht die einzelne Taste eines Keyboard gemeint ist, dann ein „Schlüssel" im übertragenen Sinn, der den Zugang zu Informationen eröffnet.

Bei einer Datenbank bezeichnet man z. B. das Schlüsselfeld für Suchvorgänge als „Key". Die Datensätze bilden jeweils eine aus mehreren Feldern bestehende Einheit. In einem Adressenverzeichnis etwa bilden Name, Anschrift und Rufnummer jedes Teilnehmers den individuellen Datensatz – es gibt ein Namen-, Adressen- und Nummernfeld. Wird ein Auszug aus dem Verzeichnis benötigt, ist eins dieser Felder als „Key" für die Auswahl anzugeben.

Bei einigen Datenbanksystemen ist die Suche unter mehreren Gesichtspunkten möglich; dabei können Sie dann als Hauptkriterium einen „Primärschlüssel" (Primary Key) und dazu einen „Sekundärschlüssel" (Secondary Key) benennen usw. Kompliziert wird es, wenn das System in einem der Felder die Eingabe von unformatierten Textbeschreibungen in Form ganzer Sätze oder sogar Abschnitte vorsieht.

Bei geschützten Dateien dient oft ein Schlüsselwort (Paßwort) oder eine Schlüsselnummer zur Benutzeridentifikation – damit wird die Sperre gegen möglichen unbefugten Gebrauch geöffnet.

Schließlich bezeichnet man auch den Umsetzungscode, nach dem bei Chiffrierverfahren die Nachrichten verschlüsselt werden, als Key. Bei einem Substitutions-Code enthält die Key-Tabelle die Ersatzzeichen für die Originalbuchstaben. Dabei wird die Zuordnung mit Hilfe von Zufallszahlengeneratoren im Rechner erzeugt, so daß der Code ohne den zugehörigen Schlüssel praktisch nicht zu knacken ist.

Keypad = Tastenblock
Ein „Keypad" ist ein kleines Tastenfeld für spezielle Zwecke, zum Beispiel für die Eingabe von Zahlenwerten. Bei vielen Microcomputern enthält die Tastatur einen numerischen Block, in dem Ziffern- und Funktionstasten nach dem gleichen Schema wie bei Taschen- oder Tischrechnern angeordnet sind.

Bei manchen Rechnern ist der numerische Block in das alphanumerische Tastenfeld integriert, indem bestimmten Tasten eine Doppelfunktion zugewiesen ist. Meist werden die Buchstaben M, J, K, L, U, I und O zusätzlich mit den Ziffern 0–6 belegt; zusammen mit den Ziffern 7, 8 und 9, die in der obersten Tastenreihe darüberliegen.

Microrechner wie der ACT Apricot oder der IBM PC haben dagegen einen getrennten Ziffernblock, der rechts neben dem alphanumerischen Feld liegt.

Bildnachweise
1597: Science Photo Library
1598: Courtesy of Robocom
1599: Modelbox des Theatre Royal, Stratford East; Courtesy of Whitechapel Computer Centre
1602, 1603: Ian McKinnell
1605: Liz Dixon
1606, 1607: Ian McKinnell
1609–1611: Crispin Thomas
1612: Terry Kennett
1613: Ian McKinnell; Allan Martin; Courtesy of Simulation Database Project, Glasgow College of Education
1615: David Higham
1618: Liz Dixon
1620: Ian McKinnell
1621: Liz Dixon
1622–1624: Kevin Jones

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

computer kurs

Produktionsmittel

Computergesteuerte Maschinen arbeiten schneller als menschengesteuerte.

Abgeräumt

Billard mit dem Computer. Interessante strategische Spielmöglichkeiten für Fans des ruhigen Spiels.

Montage des Robot-Arms

Langsam nimmt der Arm Gestalt an. Der Zusammenbau beginnt.

Leinen los!

Alle Vorbereitungen für die Expedition sind abgeschlossen. Es geht los!