Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 42

BASIC: Abenteuerspiel

Perspektivisches Colour Genie von Eaca

Geisteskraft

Ein wöchentliches Sammelwerk

computer kurs

Heft 42

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

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

Abonnement

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

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

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

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

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

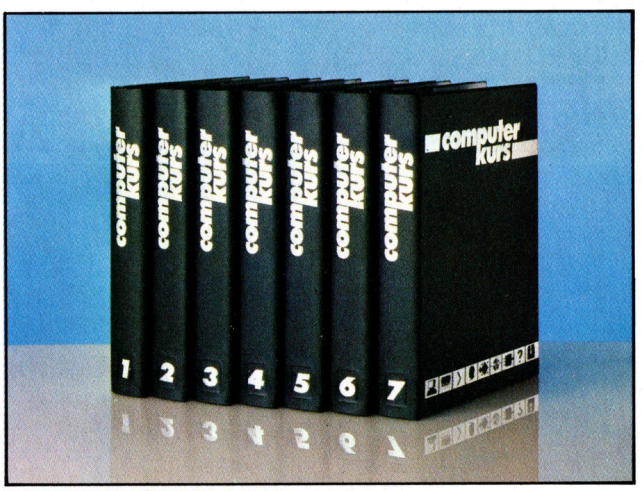
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

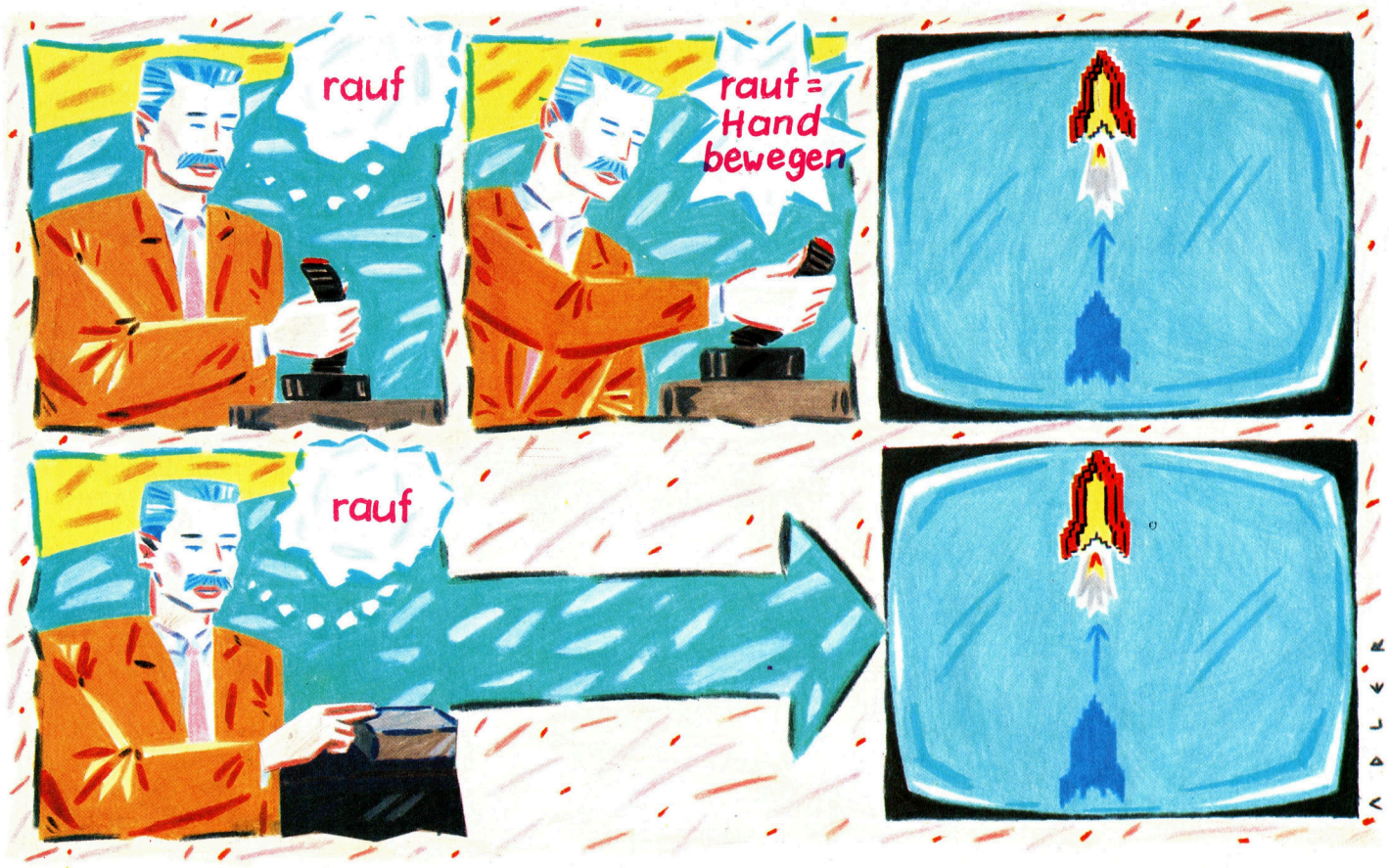
Alle 12 Hefte erscheint ein Teilindex. Die letzte Ausgabe von Computer Kurs enthält den Gesamtindex — darin einbezogen sind Kreuzverweise auf die Artikel, die mit dem gesuchten Stichwort in Verbindung stehen.

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

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



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



Geisteskraft

Die Vorstellung, einen Computer durch Gedankenkraft steuern zu können, mutet an, als stamme sie aus einem Science-Fiction-Film. Und doch handelt es sich dabei bereits um eine Tatsache. Wir stellen das Phänomen vor, auf dem die Technik basiert.

Beim Mindlink-Konzept entfällt das zwischengeschaltete mechanische Eingabegerät. Die Impulse des Spielers werden direkt in für den Computer verständliche Signale übersetzt. Durch diesen direkteren Kontakt zwischen Benutzer und Rechner gibt es neue Möglichkeiten für anwenderfreundliche Software und schnellere Eingaben.

Seit es Computer gibt, dient vor allem die Tastatur als Eingabeeinheit. Über sie also bringt man Informationen in einen Computer. Es ist die effizienteste Methode, Text einzugeben, zumindest so lange, bis universell verwendbare Stimm-Erkennungssysteme entwikkelt worden sind. So gut eine Tastatur zur Texteingabe sein mag, für die Eingabe anderer Informationsarten ist sie nicht unbedingt die beste Eingabeeinheit. Will man rasch Zugriff zu Richtungsdaten haben, wie es etwa bei vielen Spielen der Fall ist, wird man einen Joystick, Trackball oder eine Maus vorziehen. Maus, Lichtgriffel oder berührungsempfindlicher Bildschirm sind für die Eingabe von Positionsdaten wie bei der Menüwahl besser geeignet.

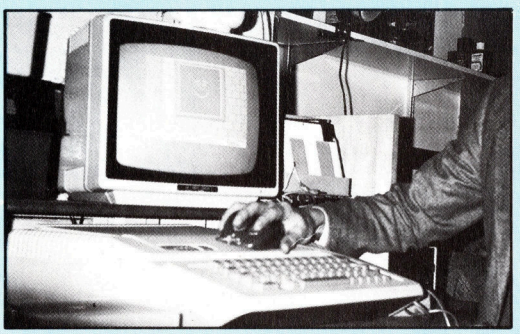
Alle aufgeführten Eingabeeinheiten haben einen gemeinsamen Nachteil: Sie sind indirekt. Zwar ist ein Joystick für Spiele geeigneter als eine Tastatur, und doch erfolgt eine Unterbrechung, eine Pause, zwischen dem, was Sie wollen, das der Computer tun soll, und dem, was tatsächlich geschieht. Man denkt „aufwärts", überträgt dies geistig in „Joystick vorwärts" und betätigt dann physisch den Joystick. Will man zu einer schnelleren und direkteren Interaktion zwischen Benutzer und Computer kommen, muß dieser Zwischenschritt beseitigt werden: Warum nicht einfach „aufwärts" denken und den Computer auf diesen Gedanken reagieren lassen?

Stellen Sie sich vor, man würde „Defender" durch Denken der Befehle „rauf", „runter", „wenden", „Feuer" usw. spielen oder einen Brief durch einfaches Denken der Wörter zu Papier bringen. Es mag noch ein paar Jahre dauern, bis es eine Textverarbeitung auf telepathischer Basis gibt, aber das durch Gedanken gesteuerte Spiel ist heute schon Wirklichkeit geworden. Möglich wurde das durch ein als „Mindlink" bekanntes Verfahren.

Beim Mindlink-Konzept steuert man mit Gedanken einen elektrischen Verbraucher. Genauer: Die physiologischen Veränderungen,

Geistige Notizen

Hier läuft ein GSR-Prototyp mit einem Apple-Computer. Ähnliche Systeme werden auch für IBM- und Commodore-Computer entwickelt. Doch die Erstellung geeigneter Software braucht noch Zeit. Roger Dilts, Präsident von Behavioral Engineering und Autor der GSR-Software, ist speziell daran interessiert, Mindlink mit NLP (Neuro-Linguistischen Programmen) zu verbinden – einem Zweig der Psychologie, der sich vornehmlich mit dem Studium der Lerntechniken befaßt. Unter Einbeziehung der GSR-Software kann der geistige Zustand des Benutzers überwacht werden. Die Software überprüft den Grad der Anspannung des Benutzers und korrigiert das Tempo des Programmablaufs entsprechend. Roger Dilts ist davon überzeugt, daß beispielsweise Ausbildungsprogramme auf die emotionale Reaktion von Schülern eingehen können. Ist die Darstellung zu kompliziert, wird die emotionale Spannung vom Computer registriert. Anschließend kann die Ablaufgeschwindigkeit des Programms verändert oder sogar eine vereinfachte Darstellung des entsprechenden Stoffes erzeugt werden.



die sich aus Veränderungen im Gedanken-Muster ergeben, tun dies. Das Gerät ist ebenso eine Schnittstelle zum Computer wie ein Joystick oder jedes andere Eingabegerät. Mindlink basiert auf dem „GSR"-Phänomen. Das heißt: Emotionale Änderungen des Menschen schlagen sich in der elektrischen Leitfähigkeit seiner Haut nieder. Über Elektronen wird der Benutzer an einen Widerstandsmesser angeschlossen. Signale aus diesem Meßgerät werden nun an den User-Port des Computers weitergeleitet und dort von spezieller Software interpretiert und ausgeführt. Die in Kalifornien ansässige Firma Behavioral Engineering hat Spiele und Anwendungs-Software entwickelt, die auf dieser Basis arbeiten. Darunter befindet sich auch eine vereinfachte Version von Defender.

Beim herkömmlichen Spiel steuert man ein Raumschiff, das einen Planeten umkreist. Es geht darum, Außerirdische abzuschießen, ohne selbst getroffen zu werden. In der Version von Behavioral Engineering wird lediglich die Höhe des Schiffes gesteuert. Der Unterschied aber ist: Dies geschieht durch Gedankenkraft! Das Unternehmen hat ein GSR-Interface für den Apple IIe entwickelt. Der Benutzer legt lediglich Zeige- und Mittelfinger einer Hand auf ein Maus-ähnliches Gerät, das den Widerstand über die beiden Finger mißt und die Ergebnisse an den Computer weitergibt. Die Software ist so angelegt, daß eine Stromzunahme (größere Spannung durch niedrigen Widerstand) das Schiff steigen läßt, wogegen ein Sinken der

GSR: Das Mindlink-Prinzip

Es basiert auf einem Phänomen, das unter drei verschiedenen Namen bekannt ist: die Galvanische Haut-Reaktion (GSR), der Psychogalvanische Reflex (PGR) und der Electrodermale Reflex (EDR).

GSR basiert auf den Veränderungen der elektrischen Leitfähigkeit der Haut, die so auf Änderungen des emotionalen Zustands einer Person reagiert. Experimente haben bewiesen, daß der Hautwiderstand sinkt, je angespannter eine Person ist. Die wohl am besten bekannte GSR-Anwendung sind „Polygraphen", die sogenannten „Lügendetektoren".

Obwohl das GSR-Phänomen seit dem 19. Jahrhundert bei Menschen wie Tieren bekannt ist, weiß man sehr wenig über seine Ursache. Anfangs glaubte man, daß der durch Aufregung oder Angst erzeugte Schweiß als elektrolytischer Konduktor wirkt und so den Hautwiderstand herabsetzen würde. Durch neuere Experimente wurde diese Theorie in Frage gestellt.

Man weiß, daß GSR unmittelbar mit dem Grad der Spannung im sympathischen Nervensystem zusammenhängt. Dies wiederum ist abhängig vom zentralen Nervensystem und somit vom Gehirn – folglich können auch Gedanken bestimmte Reaktionen hervorrufen, die an den Rechner weitergegeben werden. Veränderungen der Hirnaktivität haben Veränderungen im Zustand des zentralen Nervensystems zur Folge. Dies erzeugt einen Zustandswechsel im sympathischen Nervensystem, woraus eine Veränderung des Hautwiderstands resultiert. Und Widerstandsveränderungen bzw. Veränderungen elektrischer Ströme sind die Grundlage jeder Eingabe-Einheit für Computer.

Spannung so auch ein Sinken des Schiffes zur Folge hat.

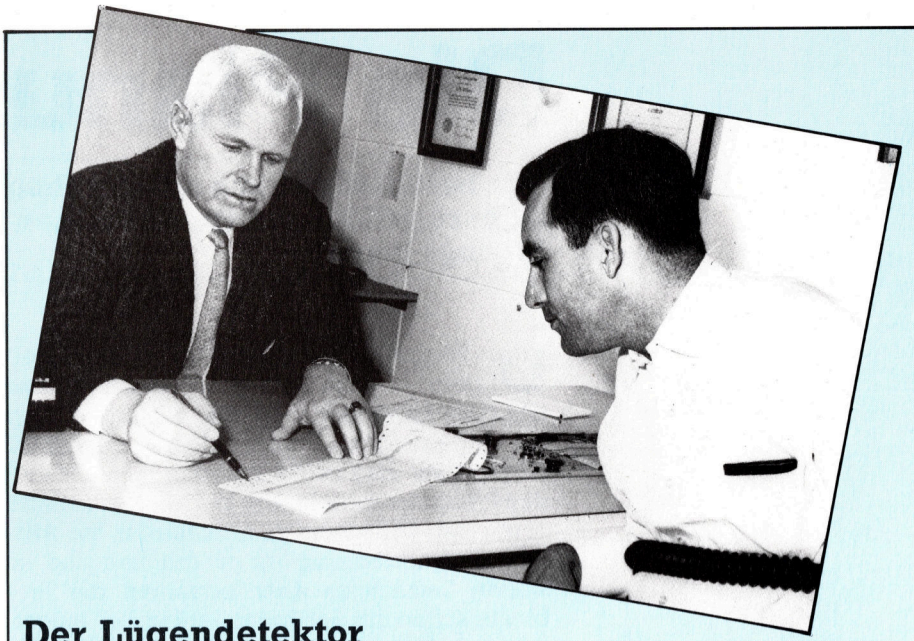
Der Begriff „Mindlink" (also etwa „Gedankenverbindung") ist mißverständlich, da es sich um eine Verbindung mit dem Nervensystem und nicht mit dem Gehirn direkt handelt. Da jedoch Nerven und Gehirn nicht voneinander unabhängig arbeiten, ist die Bezeichnung im Grunde gleichgültig. Testpersonen haben festgestellt, daß sie etwa 20 Minuten lang Kontrolle in dieser Form halten können, wobei sie oft nicht einmal wissen, wie sie es getan haben. Andere Menschen tun das bewußt, indem sie ihren Körper anspannen oder entspannen, wogegen wieder andere einfach „rauf" und „runter" denken und ihr Nervensystem den Rest erledigt.

Neben Spielen hat Mindlink natürlich auch praktische Anwendungsmöglichkeiten. Gelähmte Menschen können allein durch Gedankenkraft GSR-Effekte erzeugen, trotz der Tatsache, daß sie keine Kontrolle über ihre Muskulatur habe. Ein GSR-Gerät wurde bereits mit Erfolg über einen Apple-Computer an einen Topo-Roboter angeschlossen. So konnte die gelähmte Person den Roboter steuern. Eine andere interessante Applikation ist die Schwerelosigkeit im Weltall. Es kann außerordentlich schwer sein, mechanische Kontrolleinheiten ohne Schwerkraftausgleich zu bedienen. GSR wäre ein idealer Ersatz für mechanische Bedienungselemente.

Bei Streß: Kaffeepause

Zu den ungewöhnlicheren Anwendungen gehört die „Stimmungen fühlende Software". Lernsoftware würde besonders dafür geeignet sein, da über GSR der innere Zustand des Benutzers deutlich wird. Die Person würde ein mit Elektroden versehenes Armband tragen, und die Software würde zunächst einmal auf den entspannten Zustand kallibriert. Wenn während des Programmverlaufs das GSR-Gerät einen deutlichen Anstieg der Spannung registriert, ist anzunehmen, daß der Benutzer Schwierigkeiten hat und dementsprechend handelt. Selbst Business-Software könnte Streß beim Benutzer feststellen und zum Beispiel eine Kaffeepause vorschlagen.

Wie schnell die Entwicklung der GSR-Technologie voranschreitet, hängt vor allem von der Entwicklung genauerer und schneller reagierender Geräte sowie von unserer Fähigkeit ab, die Interpretation der Effekte zu verfeinern. Das Anfangsproblem bei GSR war, daß die Reaktion erst zwei Sekunden nach dem Ereignis erfolgt und nach einem Zeitraum von zwischen zwei und zehn Sekunden schwindet. Die derzeit verfügbaren Geräte haben dieses Problem nicht mehr, da sie die Veränderungsrate und nicht mehr die Stärke der Reaktion messen. Aber weitere Arbeit an dem GSR-Phänomen ist noch erforderlich.



Der Lügendetektor

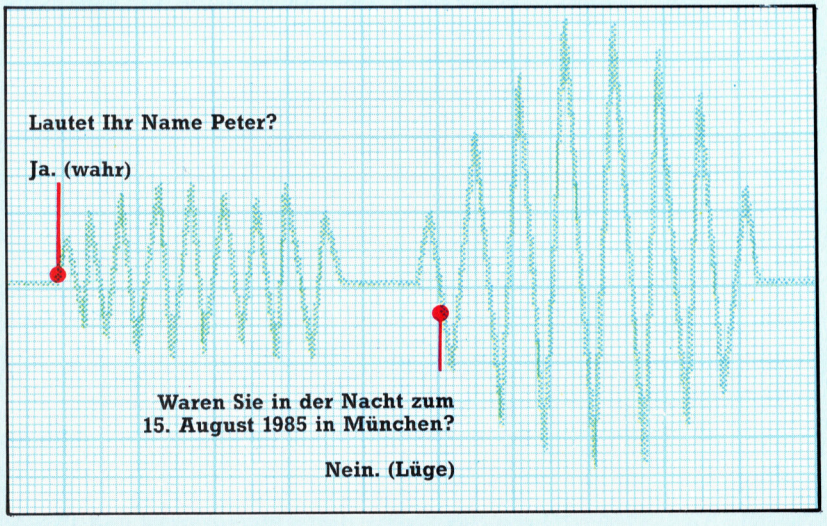
Eines der Hauptprobleme, mit dem Polizei und Justiz bei der Aufklärung von Verbrechen immer wieder konfrontiert werden, ist die Frage, ob ein Verdächtiger oder ein Zeuge lügt. Obwohl ein geschulter Beobachter das aufgrund winziger Anzeichen zu beurteilen vermag – Wechsel der Hautfarbe, Veränderungen beim Atmen etwa – konnten diese Faktoren nie katalogisiert oder generell definiert werden. Aus diesem Grunde hat man sich mit der Entwicklung von Geräten befaßt, die objektiv den Unterschied zwischen wahren und falschen Aussagen feststellen können. Eines der Forschungsergebnisse war der „Lügendetektor".

Im Prinzip ist er nichts Anderes als ein Widerstandsmesser. Die Theorie besagt, daß eine Person deutlich angespannter ist, wenn sie lügt oder ein Schlüsselwort hört, das in Verbindung mit dem Verbrechen steht. Diese Spannungszunahme macht sich im gesunkenen Hautwiderstand bemerkbar.

Der Polygraph ist ein umstrittenes Gerät. Obwohl Untersuchungsbehörden, vor allem in den USA, seinen Nutzen loben, argumentieren Kritiker, daß man zu wenig über das GSR-Phänomen wisse, um es zuverlässig interpretieren bzw. messen zu können. Dazu kommt, daß verschiedene Menschen unterschiedlich reagieren, unabhängig von Schuld oder Unschuld.

GSR wird in den USA manchmal angewendet, um Stellenbewerber zu testen. Auch hier bedient man sich eines Lügendetektors während eines Bewerbungsgesprächs. Im Bild wird gerade das Ergebnis eines Tests besprochen.

Ein Lügendetektor mißt die GSR-Veränderungen, die durch die emotionale Reaktion der Person hervorgerufen werden. Ein Lügendetektor ist jedoch außerstande, auch ein nur annähernd so deutliches Ergebnis zu bringen wie hier gezeigt. Emotionale Antworten variieren entsprechend der Sensibilität der Person gegenüber verschiedenen Dingen und hängen nicht einfach von der Wahrheit der Antworten ab.



Die Planung

Wir beschäftigen uns mit der Programmgestaltung und überlegen, was zu berücksichtigen ist, bevor man überhaupt mit dem Schreiben eines Listings beginnt.

Die Programmgestaltung wird von den Beteiligten – dem Designer/Programmierer und dem Anwender – als eine wichtige Übung zur angewandten Problemlösung betrachtet. Häufig geht man davon aus, daß die zu lösenden Probleme vornehmlich technischer Art sind – wie den Bildschirm zu formatieren, die Ausführungsgeschwindigkeit zu erhöhen und so weiter. Tatsächlich aber beginnen die Probleme schon mit der Formulierung der eigentlichen Aufgabe des Programms. Die Grundlagen zu ihrer Lösung werden gewöhnlich im ersten Gespräch zwischen dem Anwender und dem „Experten" geschaffen. Anwender sind sich jedoch selten über die wahre Natur der Probleme im klaren. Sie hoffen, vom Experten eine Problemanalyse und eine Lösung zu bekommen. Und Experten glauben oft, ein Problem zu kennen – und auch die Lösung –, bevor der Anwender dieses überhaupt erklärt. Durch diese Kommunikationsschwierigkeiten kommt es zu einer unvollständigen Problembeschreibung, und die Belange des Anwenders werden ungenügend berücksichtigt. Auf dieser Grundlage ausgeführte Arbeiten haben als Ergebnis ein für den Anwender unbefriedigendes Produkt, das er nur widerwillig akzeptiert.

Kommunikationsprobleme

Bei Heimcomputer-Projekten ist der Programmierer meistens Designer (oder „Systemanalytiker") und Anwender in einer Person. Man sollte annehmen, daß die Kommunikationsprobleme nicht auftauchen. Trotzdem kann es zu denselben Schwierigkeiten kommen, wenn man sich nicht selbst die Probleme, Lösungen und Erfordernisse so darlegt, als spräche man mit einer anderen Person darüber.

Beschäftigen wir uns mit einem imaginären Anwender und seinem Problem: Er ist ein guter Flugzeugmodellbauer, der einen Computer mit Cassettenspeicher hat. Er will detaillierte Beschreibungen der beim Bau verwendeten Materialien all seiner Modelle speichern, um bei späteren Bauten nachsehen zu können, welcher Leim und welche Verbindungen eingesetzt wurden. Der Programmierer benötigt deshalb vom Anwender folgende Anweisungen:

- Programmfunktion: Dies kann eine vage Absichtserklärung sein wie „Es sollte meine Modellbau-Aufzeichnungen speichern können", die dann aber durch Nachfragen und Überlegen des Programmierers weiter spezifiziert wird. Zum Beispiel: „Es sollte meine Modellbeschreibungen, seine Konstruktionen und die dabei verwendeten Materialien so speichern, wie ich sie über Tastatur eingebe, und sie darstellen, wenn ich die Modellbezeichnung oder ein bestimmtes Konstruktionsmerkmal eingebe." Das macht die Bedürfnisse des Anwenders deutlicher und verweist auf bestimmte Programmieraufgaben (Speichern, Suchen, Inhaltsverzeichnis, Zugriff etc.).

Genaue Beschreibung erwünscht

- Programmnutzung: Einige der formellen Einzelheiten der gewünschten Nutzung mögen zwar aus der Funktionsbeschreibung deutlich werden, sind aber eventuell nicht vollständig. Vielleicht möchte der Anwender beispielsweise nicht, daß die Daten auf dem Bildschirm dargestellt werden, sondern wünscht lediglich eine exakte Druckerausgabe.
- Gestaltung des Programms: Ein- und Ausgabe-Formate. Professionelle Programmierer benutzen häufig vorgedruckte Karten, worauf jeder Screen so dargestellt ist, wie ihn der Anwender in den Eingabe/Ausgabe-Phasen sieht oder sehen möchte. Bildschirm-Formate sind wichtige Eigenschaften der sogenannten „Benutzer-Schnittstelle", womit die Zusammenarbeit von Anwender und Maschine gemeint ist.
- Programmorganisation: Datei- und Programm-Formate. Der Anwender möchte vielleicht mindestens 100 Flugzeugmodell-Beschreibungen speichern. Die Größe der Programm-Datenfiles hat erheblichen Einfluß auf ihr Format und die Zugriffsmöglichkeiten. Der Suchlauf durch beispielsweise sechs Modellbeschreibungen auf Cassette beansprucht circa fünf Minuten, was für den Benutzer zumutbar ist. Auf das Durchsuchen von 100 Modellen zu warten wäre dagegen indiskutabel. Eine Lösungsmöglichkeit bestünde darin, Programm und Index-File auf einer Cassette zu speichern, die Beschreibungen dagegen auf 20 anderen Cassetten abzulegen, beispielsweise klassifiziert nach Flugzeugtypen.
- Aufgabenstellung: spezielle Prozeduren und Berechnungen. Bei unserem Beispiel mag sich dieses Problem nicht stellen. Doch es wird deutlich, wenn man andere Probleme mit berücksichtigt. Es mag 20 perfekte und gleichermaßen befriedigende Wege zur Lösung eines

bestimmten Problems geben. Der Anwender aber verlangt nur nach einem bestimmten.

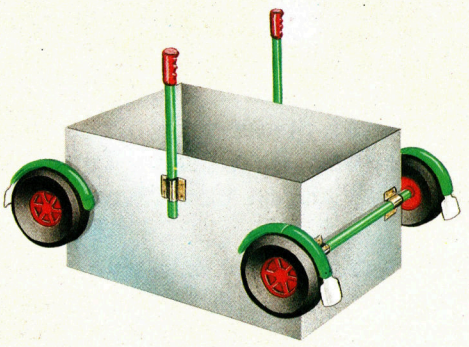
Berücksichtigt man dies nicht, wird der Benutzer mit dem Programm unzufrieden sein. Der Autor mag sich von der vom Anwender bevorzugten Methode wegen besserer Wirtschaftlichkeit anderer Methoden entfernt haben. Der sich daraus ergebende Vorteil aber erweist sich als null und nichtig, wenn der Anwender ein solches Programm nicht benutzen will. Die Arbeitsweise des Anwenders zu verstehen kann bei der Programmberechnung nützlich sein.

Liegen all diese Informationen vor, kann man mit der Umwandlung der Einzelheiten in ein Programm beginnen. Als sinnvoll hat sich erwiesen, zunächst den Anwender-Programm-Dialog zu schreiben, dann die Daten-Files und schließlich die Steuerprozesse. Mit dem Wort „Dialog" ist die zweiseitige Kommunikation, der Austausch von Informationen zwischen Anwender und Programm, gemeint. Diese besteht bei unserem Beispiel nicht allein aus der Eingabe von Modellflugzeug-Einzelheiten und ihrer richtigen Darstellung, sondern schließt auch jedes Prompt, Menü und jede Nachricht ein, die das Programm erzeugen kann, sowie alle Eingaben oder Befehle, die der Benutzer gibt. Die Festlegung der Art, wie der Dialog erfolgen soll, ist in diesem Stadium ebenfalls wichtig. Für das Flugzeugprogramm könnte eine Mischung zwischen Menü und befehlsgebundener Interaktion geeignet sein. Die hier getroffene Entscheidung beeinflußt die Struktur des gesamten Programms. Inhalte und Format des Dialogs müssen in allen Einzelheiten berücksichtigt werden. Die „Belohnung" für all diese Mühen ist, daß die vom Programm verarbeiteten Daten nun auch noch spezifiziert werden müssen. Danach kann der für Fehlermeldungen und Bestätigungen erforderliche Speicherplatz berechnet werden, und die Files können programmiert werden. Für das Flugzeugmodell-Programm, dessen Files große und lange Textblöcke enthalten, mag die Untergliederung in mehrere Unterprogramme auf verschiedenen Cassetten die beste Lösung sein.

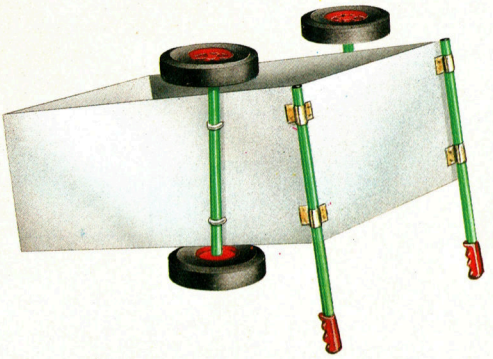
Angepaßte Funktionen

Die erforderlichen Funktionen werden jetzt offenkundig. So muß es Routinen geben, die das Hinzufügen und Editieren von Text-Daten erlauben, ferner das Speichern neu eingegebener Texte (womit automatisch das Inhaltsverzeichnis aktualisiert werden muß), die Suche und Darstellung von Beschreibungen muß möglich sein usw. All dies sollte dem Anwender in Form von Optionen angeboten werden.

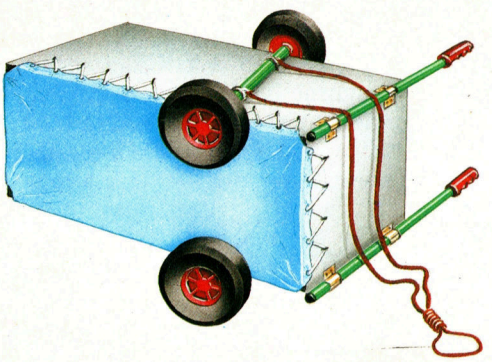
An diesem Punkt sollte der Benutzer die Programmgestaltung sorgfältig überprüfen, um festzustellen, ob es wirklich allen Erfordernissen genügt. Erst dann sollte das Programm codiert werden. Das ist natürlich leichter gesagt als getan – doch dazu später Genaueres.



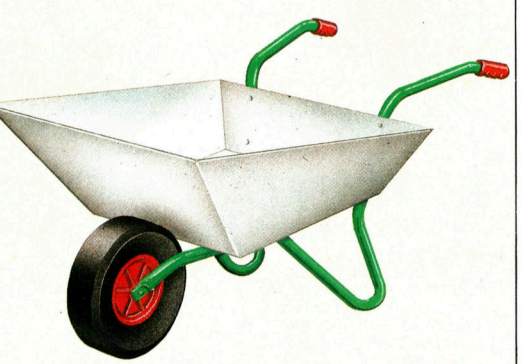
Der Designer glaubte, der Anwender wolle dies



Der Programmierer vermutete, dies meine der Designer



Das wurde dem Anwender schließlich verkauft



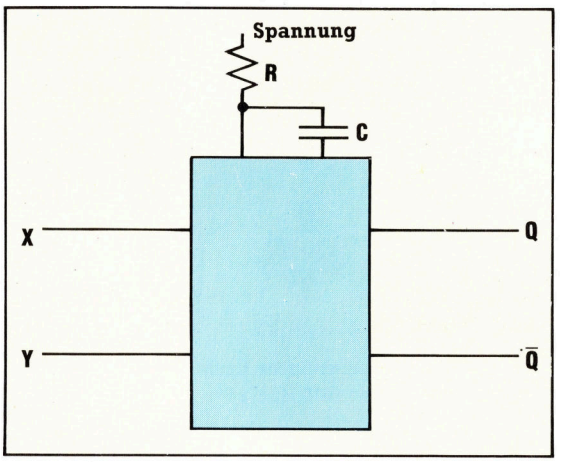
Und das wollte der Anwender eigentlich haben

Wenn das typische Software-Entwicklungsteam – bestehend aus Anwender, Gestalter und Programmierer – das Problem zu lösen hätte, wie man schwere Lasten im Garten bewegt, könnte dies dabei herauskommen. Kommunikationsschwierigkeiten zwischen Experten und Laien und Experten untereinander sind noch immer Kernproblem aller Entwicklungsteams.

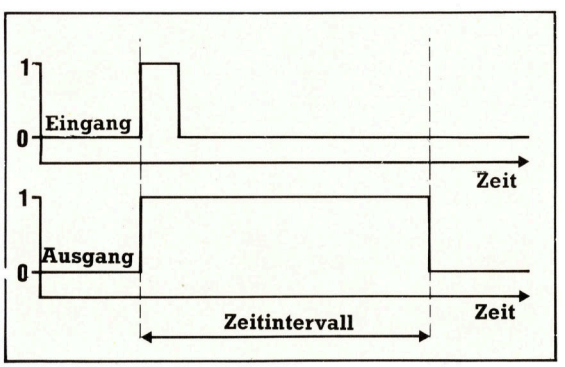
Taktvoll

Zur exakten Steuerung eines Computers ist genaues „Timing" unabdingbar. In diesem Kursabschnitt wollen wir Ihnen monostabile Kippschaltungen und zwei spezielle Flipflops vorstellen.

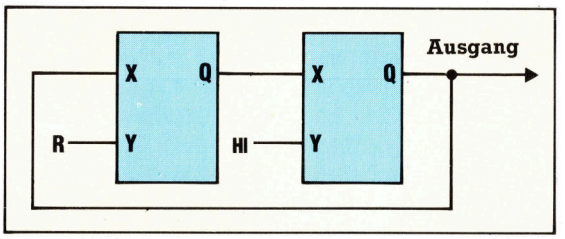
Mit monostabilen Schaltungen lassen sich logische Abläufe in exakten Zeiteinheiten steuern. Empfängt die Kippschaltung einen Impuls, wird der Ausgang für eine bestimmte Zeit auf 1 (HI) gesetzt, bevor er in den Ruhezustand (LO) zurückkehrt. Hier das Beispiel einer Kippschaltung:



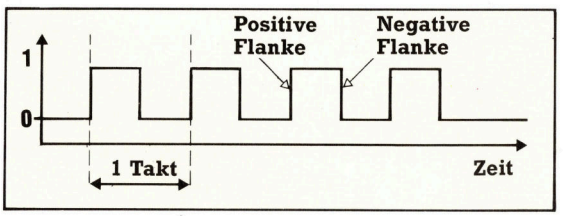
Der beschriebene Ablauf wird entweder durch Umschalten des X-Einganges von HI auf LO oder durch Umschalten des Y-Einganges von LO auf HI gestartet. Die Werte von Widerstand (R) und Kondensator (C) entscheiden über das Schaltintervall. Diese Kurven stellen den zeitlichen Zusammenhang der Ein- und Ausgangszustände dar:



Das HI-Intervall am Ausgang läßt sich etwa zur Steuerung eines Cassettenrecorder-Schrittmotors oder zur Verzögerung der Ubertragung eines Bits einsetzen. Mit zwei monostabilen Kippschaltungen kann ein Oszillator aufgebaut werden, der in gleichbleibenden Zeitabständen zwischen HI- und LO-Intervallen hin- und herschaltet: Am Ausgang ergibt sich eine

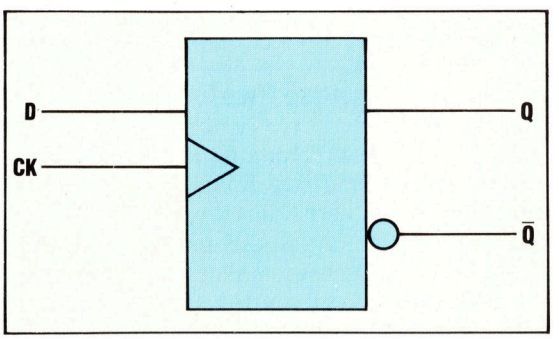


Rechteckspannung. Die Zeit zwischen zwei HI-Zuständen wird als „Zyklus" bezeichnet. Gewöhnlich liegen diese Zykluszeiten im Bereich von einer Millionstelsekunde. Der Maschinentakt ist der „Herzschlag" Ihres Computers, der alle Abläufe in der CPU dirigiert. Die zwei Flanken der einzelnen Rechteck-Impulse haben verschiedene Namen, die wir in der Zeichnung angeben:



Als nächstes sollen Sie zwei weitere Flipflop-Typen kennenlernen, die jedes Umschalten mit den Taktzyklen eines Oszillators koordinieren können.

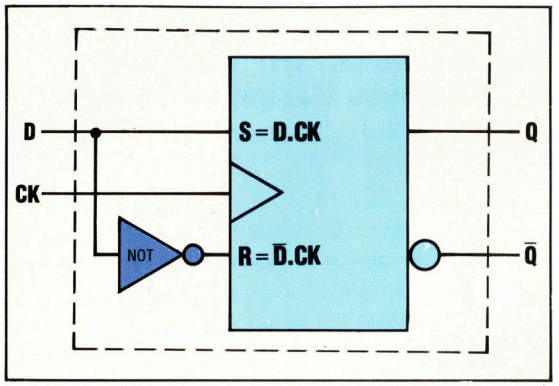
D-Flipflops haben einen logischen (D) und einen Takteingang (CK = clock = Uhr):

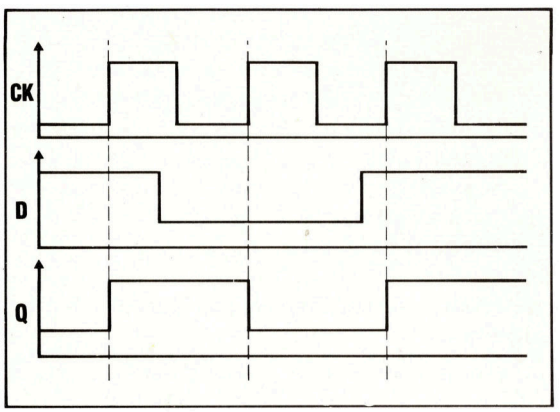


Der D-Flipflop baut auf dem bekannten RS-Typ aus unserem letzten Kursabschnitt auf. Durch den zusätzlichen Clock-Eingang kann jedoch der Eingang der Schaltung zeitweilig verriegelt werden.

Der Ausgangszustand von Q wird am Beginn jedes neuen Taktzyklus festgelegt. Liegt zu

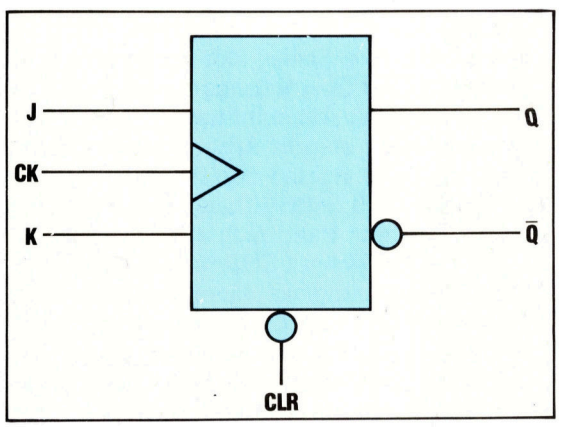
diesem Zeitpunkt ein HI auf Eingang D, so wird auch Q auf HI gesetzt, ist D aber LO, geht auch Q auf LO.



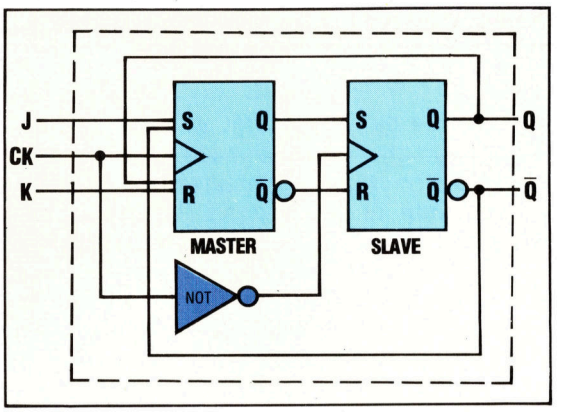


Aus der Zeichnung wird deutlich, daß Ausgang Q nur veränderlich ist, wenn der Eingang CK gerade von LO nach HI springt.

Der JK-Flipflop wirkt nach dem „Master-Slave"-Prinzip: In diesem Schaltungstyp arbeitet einer der zwei zusammengehörigen RS-Flipflops abhängig vom anderen. Bei der Veränderung des Eingangswertes wird das HI oder LO in einem Flipflop zwischengespeichert. Am Ausgang steht bis zum nächsten Taktzyklus der Eingangswert des vorherigen Taktes. Nützlich ist dies etwa bei der in vielen Prozessoren häufigen Verschiebung von Bits um eine Stelle nach links oder rechts. Hier das Schaltbild:

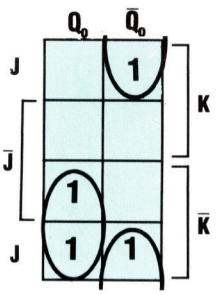


Das nächste Bild zeigt die Schaltung von zwei RS-Flipflops. Einer ist „Master" (Herr), der andere „Slave" (Sklave). Wenn CK HI ist, geht das Eingangssignal zum Master-Flipflop. Bei einem LO auf CK wird das Eingangssignal wegen der „Triggerung" mit der positiven Flanke auf den Slave-Flipflop geleitet. Es ist jeweils nur ein Flipflop aktiv, während der andere als Zwischenspeicher dient:



Rechts ist eine Tabelle mit den Schaltzuständen des JK-Flipflops abgedruckt. Q_o bezeichnet dabei den Ausgangswert während des vergangenen Taktzyklus. Wenn sowohl am J- als auch am K-Eingang ein HI liegt, ändert der Ausgang bei jedem Taktzyklus seinen Zustand. Dieses als „toggling" bezeichnete Verhalten entsteht durch Rückkopplung vom Slave- zum Master-Flipflop. Stellt man die k-Tafel auf, müssen Q_o, J und K als Eingangsvariablen aufgefaßt werden.

Q_o	J	K	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



Aus der Tafel ergibt sich der Ausdruck

$$Q = \overline{Q}_o \cdot J + Q_o \cdot \overline{K}$$

Diese Gleichung wird als die „Charakteristik" des JK-Flipflops bezeichnet.

Lösungen der Übung 7

1) Einen Flipflop nennt man bistabil, weil er in zwei Zuständen stabil bleibt – wenn Q=1 und $\overline{Q}$=0 oder Q=0 und $\overline{Q}$=1 ist.
2) a) Ist kein stabiler Zustand.
b) Wird der obere Eingang zuerst berücksichtigt, geht der Flipflop auf RESET. Auf SET schaltet er, wenn zuerst der untere Eingang berücksichtigt wird.
c) Ja (siehe auch Antwort b).
d) Um den Zustand aller Register nach dem Einschalten festzulegen, müssen sie bei dem Systemstart entsprechend auf SET oder RESET gesetzt werden.

Dienstbarer Geist

In unserer Serie über Kalkulationssysteme nehmen wir TK!Solver unter die Lupe. Dieses Programm führt ein neues Konzept ein: die Verarbeitung von Gleichungen.

Es wurde bereits gezeigt, daß sich Kalkulationsprogramme ausgezeichnet für eine ganze Reihe von mathematischen Aufgaben eignen. Alle umfangreichen Tabellen, deren Berechnung per Hand und Taschenrechner viel Aufwand erfordert, werden mit dieser Art Programm in kürzester Zeit erledigt. Trotz ihrer Vorteile sind Kalkulationssysteme in einigen Bereichen jedoch Beschränkungen unterworfen. So eignet sich das Format der Zeilen und Spalten zwar ideal für finanzielle Modelle, für Aufgaben aus Bereichen der höheren Mathematik und der Wissenschaft ist es jedoch sehr umständlich oder überhaupt nicht einzusetzen. Die meisten Programme können außerdem Gleichungen nur innerhalb fester Grenzen verarbeiten.

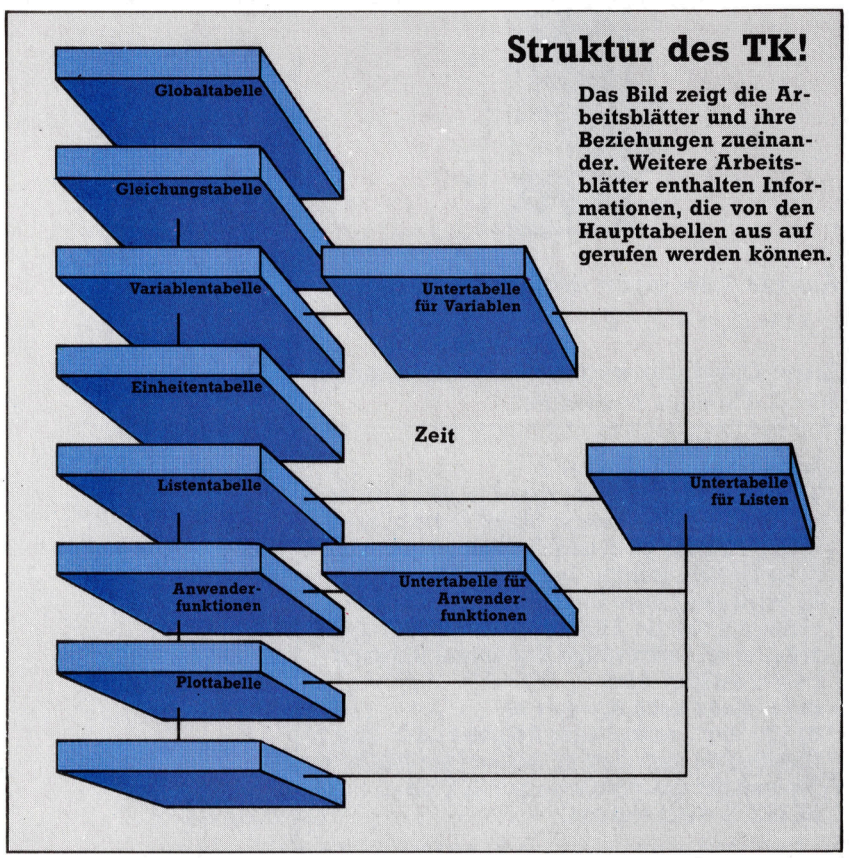
Software Arts – eine amerikanische Firma, die auch VisiCalc entwickelte – hat nun ein Programm mit dem Namen TK!Solver entwickelt, das von Format und Funktion her weit über die Möglichkeiten der herkömmlichen Kalkulationssysteme hinausgeht. „TK!" steht dabei für Tool-Kit, während „Solver" ein Teil des Codes ist, der Gleichungen verarbeitet. Außer einem völlig andersartigen Bildschirmformat bietet TK! folgende interessante Möglichkeiten:

Rückwärtsberechnungen – Viele Systeme können dies nur für eine Variable ausführen. Mit TK! läßt sich jede Variable einer Gleichung auf diese Art berechnen; vorausgesetzt, die entsprechenden Daten sind vorhanden.

Iteration – Fehlende oder unbekannte Werte für die Ausführung von Gleichungen findet TK! von einem angegebenen Anfangswert aus mit einer Reihe von Annäherungen.

Umwandlung von Einheiten – Über Umformungstabellen kann TK! Feet in Meter, Dollar in Pfund etc. umwandeln.

Mathematische Funktionen – TK! besitzt eine große Anzahl eingebauter Funktionen.



Viele Arbeitsblätter

TK!Solver arbeitet mit drei linierten Arbeitsblättern, die unterschiedliche Aufgaben haben. So enthält die Variablentabelle außer den Namen aller definierten Variablen weitere Spalten mit den vom Anwender eingegebenen Werten, Bezügen zu anderen Einheiten und ausreichenden Platz für Kommentare. Die Variablentabelle erscheint im oberen Teil der Eingangsanzeige, wobei ihre Einzelheiten in einer Untertabelle gespeichert sind. In die Gleichungstabelle (Rule Sheet) im unteren Bildschirmbereich werden Formeln eingetragen, die TK! lösen soll.

Mit diesen drei Tabellen löst TK!Solver die meisten Gleichungen. Es stehen aber noch weitere Arbeitsblätter zur Verfügung: Über die Globaltabelle kann der Anwender einige TK!-Funktionen für seine Zwecke umformen. In der Listentabelle (List Sheet) läßt sich eine Matrix für Variablenwerte unterbringen, eine weitere Tabelle enthält selbstdefinierte Anwenderfunktionen, und es gibt Arbeitsblätter für das Plotten von Punkten und den Druck von Listen.

Fangen wir mit dem Aufbau eines einfachen Modells aus dem Handbuch des TK!Solver an. Das Modell berechnet Entfernung und Durchschnittsgeschwindigkeit einer Autofahrt und wandelt Meilen in Meter um. In der eingangs dargestellten Anzeige steht der Cursor in der Gleichungstabelle im unteren Bildschirmbereich. Die Variablendefinition beginnt mit folgender Gleichung:

distance/time=speed

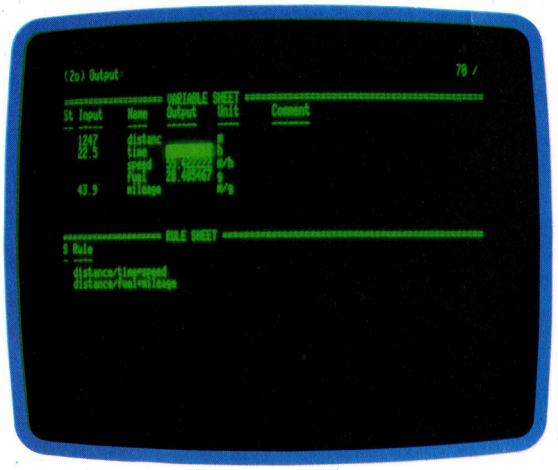
Nach dem Drücken von Return überträgt der

TK!Solver die Variablennamen automatisch in die Namensspalte der darüberliegenden Variablentabelle. Dabei wird in der Statusspalte hinter jedem Variablennamen ein Stern dargestellt. Er zeigt an, daß die Variablentabelle für diese Gleichung keinen Wert enthält. Die zweite Gleichung wird ebenso eingegeben:

distance/fuel=mileage

Nach Eingabe dieser Gleichung zeigt die Namensspalte der Variablentabelle alle fünf Variablen wie folgt an:

Gleichungen und Variablen



Drücken Sie nun die ; -Taste, um den Cursor in die Variablentabelle zu stellen und Werte einzugeben. Der Cursor steht jetzt in der Eingabespalte hinter der ersten Variablen (distance). Die Eingabe der folgenden Werte wird jeweils mit Return oder Abwärtspfeil abgeschlossen.

INPUT	NAME	OUTPUT
500	distance	
8.5	time	
	speed	
14	fuel	
	mileage	

Die Einträge für „speed" und „mileage" wurden freigelassen, da TK! diese Werte finden soll. Das Berechnungsergebnis wird in Spalte „OUTPUT" gezeigt. Um nun die Werte für „speed" und „mileage" zu erhalten, drücken Sie das Ausrufungszeichen (!), das TK! die „Aktionstaste" nennt. Über der Variablentabelle erscheint nun die Meldung „Direct Solver" (Direkte Lösung), da das Programm alle für die Lösung benötigten Daten bereits hat. Nach kurzer Zeit werden die Werte für „speed" und „mileage" ausgegeben. Vorhandene Eingabewerte lassen sich durch neue Werte überschreiben.

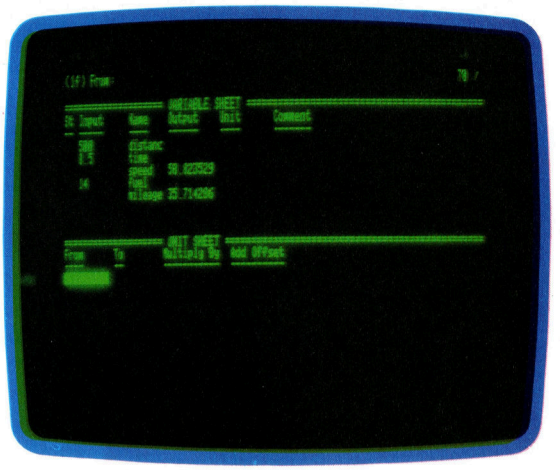
Die Werte des Modells sind noch nicht mit Einheiten gekennzeichnet. Wir können jedoch nicht einfach „Meilen" oder „Gallonen" in die Variablentabelle eintragen, da diese Bezeichnungen noch nicht definiert wurden. Bewegen Sie also den Cursor wieder in die Gleichungstabelle, indem Sie (;) drücken, und geben Sie =U ein. TK! zeigt nun statt der Gleichungstabelle die Einheitentabelle (Unit Sheet).

Die Einheitentabelle hat vier Spalten:

From To Multiply by Add Offset

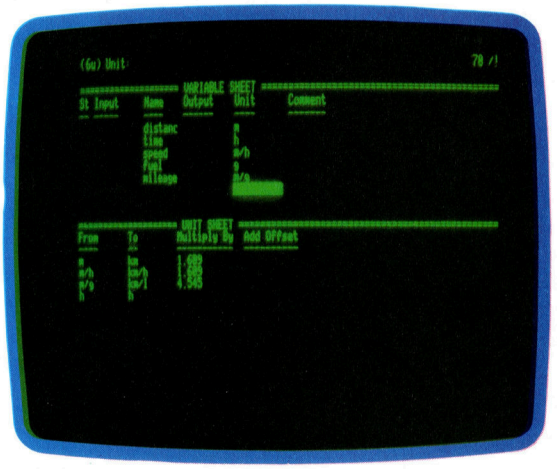
Der Cursor steht hier unterhalb des Wortes From. Tragen Sie jetzt die Einheiten und Umwandlungswerte ein.

Umwandlungstabelle für Einheiten



Drücken Sie =R, um wieder die Gleichungstabelle zu erhalten, und (;) für die Variablentabelle. In der Einheitenspalte können Sie nun die definierten Namen eintragen – m für distance; h für time; m/h für speed; g für fuel und m/g für mileage. Tragen Sie jetzt folgende Werte ein: 1247 distance; 22.5 time und 43.9 mileage. Wenn Sie die !-Taste drücken, erhalten Sie die Ergebniswerte in Metern.

Umwandlung von Meilen in Kilometer



Setzen Sie jetzt den Cursor über das m der Einheitenspalte und geben Sie km (Kilometer) ein. Bei Return wandelt TK! den Wert „distance" von Meilen in Kilometer um und zeigt statt 1247 die Zahl 2006,423 auf.

Dieses einfache Modell spricht nur wenige Möglichkeiten des TK!Solver an. In der nächsten Folge werden wir uns daher ein hochentwickeltes Modell ansehen, das die Funktionen von TK! einsetzt und Werte plottet.

Abenteuerlich!

Adventures zählen zu den beliebtesten Computerspielen. Das Spielen eines solchen Abenteuers ist jedoch nur der halbe Spaß – das Schreiben ist eine noch unterhaltsamere und zudem kreativere Beschäftigung. Mit diesem Artikel beginnt ein Programm-Projekt, mit dem wir Ihnen alle Entwicklungsabschnitte erklären.

Abenteuerspiele wurden Anfang der 70er Jahre populär, als das Spiel „Dungeon and Dragons" entwickelt wurde. In diesem Spiel schlüpfen die Spieler in die Rolle verschiedener Charaktere in einer imaginären Welt. Diese besteht oftmals aus einem verworrenen Labyrinth von Räumen, in denen der Spieler Objekte zu sammeln und Gefahren zu bewältigen hat. Das Ziel des Spiels ist im allgemeinen, aus dem Labyrinth zu entkommen und dabei eventuell eine Person oder einen Gegenstand zu finden.

Einige Abenteuerspiele bestehen ausschließlich aus Text, wogegen andere Farbgrafiken verwenden. Einige Kritiker sind jedoch der Meinung, daß die Verwendung von Grafik wertvollen Speicherplatz verbraucht, der sinnvoller zur Erweiterung der Spiel-Struktur eingesetzt werden könnte. Außerdem argumentieren sie, daß eine Computergrafik nicht die Wirkung habe wie das Phantasiebild, das der Spieler sich aufgrund einer Textbeschreibung machen könne. Trotzdem ist die Beliebtheit von Abenteuerspielen sicherlich auch wegen der Grafikunterstützung gestiegen.

In diesem Programm-Projekt wollen wir uns mit den Techniken zur Entwicklung eines solchen Spiels befassen. Im Verlauf des Projekts besprechen wir schrittweise Teile des Listings des Abenteuerspiels „Digitaya". In diesem Spiel nimmt der Spieler als „elektronischer" Agent teil. Dabei hat er die Aufgabe, innerhalb eines Computers die mysteriöse „Digitaya" zu finden und aus der Gewalt der Maschine zu befreien. Sie müssen Ihr ganzes Wissen über Microcomputer einsetzen, um unbeschadet zu entkommen. Das Programm ist, so weit wie möglich, in „normalem" BASIC geschrieben, wobei einige Besonderheiten entsprechend erklärt werden. Vorausgesetzt, Ihr Computer verfügt über genügend Speicherplatz, sollte das Programm auf Ihrem Computer lauffähig sein. Parallel zu „Digitaya" wird ein kürzeres Spiel mit dem Namen „Haunted Forest" erstellt, mit dem wir Ihnen die Techniken und Algorithmen zur Entwicklung des großen Programms vermitteln. So entdecken Sie die Geheimnisse von „Digitaya" erst beim Spielen.

Ausgangspunkt für die Entwicklung des Abenteuerspiels ist das Konstruieren einer Karte der Phantasiewelt. Auf dieser Karte vermerkt man die verschiedenen Orte, die Position von zu findenden Objekten sowie spezielle Gebiete. Die meisten Orte auf der Karte kann der Spieler problemlos betreten, dort etwas aufnehmen oder ablegen und die Orte wieder verlassen. Spezielle Gebiete sind zumeist gefährlich (zum Beispiel ein Sumpf oder eine Drachenhöhle), oder es müssen bestimmte Aktionen durchgeführt werden, bevor man den Ort betreten oder verlassen kann.

Zuerst: die Karte

Der beste Weg, mit der Erstellung einer Karte anzufangen, ist, sich ungefähr zu überlegen, wieviele Orte für das Spiel gebraucht werden. Bei „Haunted Forest" gibt es zehn Orte, und es wurde auf einem 5 × 5-Raster entworfen (siehe Bild), wogegen es bei „Digitaya" nahezu 60 Orte gibt und für den Entwurf ein 10 × 10-Raster verwendet wurde.

Die Quadrate des Rasters sind am Anfang noch nicht numeriert, sondern es werden erst einmal die Orte eingetragen. Auf der Karte von „Haunted Forest" findet man einen Pfad, zwei Tunnel, einen Sumpf, eine Lichtung und ein Dorf. Außerdem sind die Positionen verschiedener Objekte am unteren Rand der jeweiligen Quadrate gekennzeichnet. Die Gebiete, die mit einem Stern (*) markiert sind, sind „speziell" und müssen anders gehandhabt werden als die restlichen Gebiete.

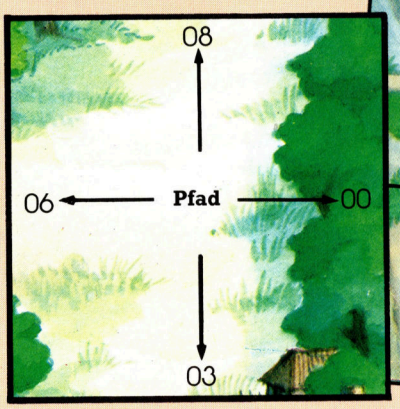
Ist der Entwurf fertig, kann jede Position numeriert werden. Dabei ist nur zu beachten, daß wir die speziellen Gebiete zuerst beziffert haben. Die Reihenfolge, in der die restlichen Gebiete numeriert werden, ist unwichtig. Denken Sie jedoch daran, daß Sie die Nummern später nicht mehr ändern können und dürfen.

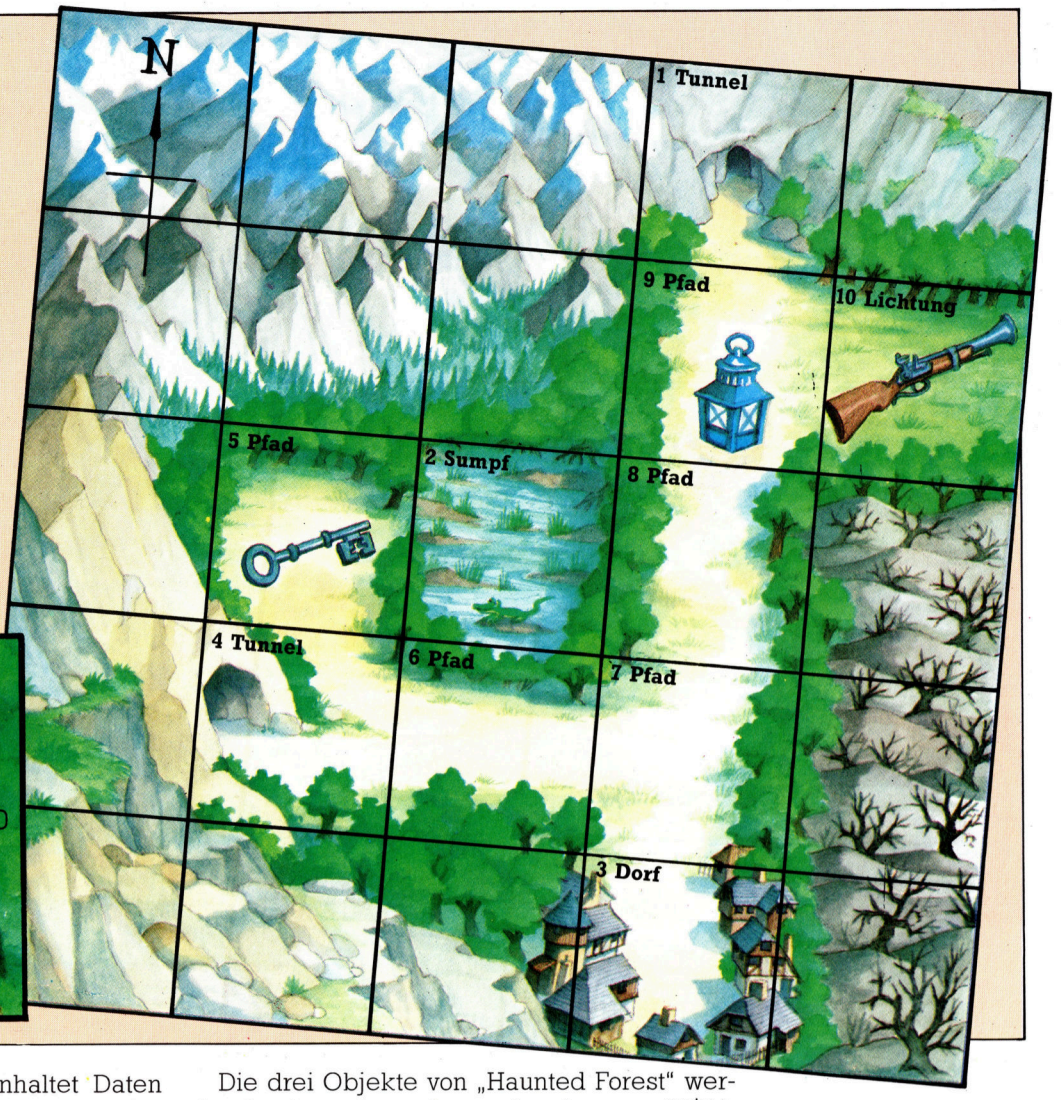
Die erste Programmieraufgabe ist, die Informationen der Karte in Daten für das Programm umzuwandeln. Hierzu gibt es diverse Möglichkeiten. Für unser Beispiel verwenden wir zwei eindimensionale Arrays zum Speichern der Karten-Daten. Das erste Array, LN$(), enthält die Ortsbeschreibungen. Das bedeutet beispielsweise für Ort 7, daß LN$(7) den Text „auf einem Pfad" (on a path) enthält. Werden diese Daten später im Programm zur Beschreibung eines Ortes verwendet, werden noch die Worte „Sie sind" (You are) vorangestellt.

Die Karte

Der erste Schritt bei der Entwicklung eines Abenteuerspiels ist das Zeichnen einer Karte, die alle Orte zeigt, die von den Spielern aufgesucht werden können. Jeder Ort erhält eine kurze Beschreibung der Umgebung sowie Angaben, ob irgendwelche Objekte gefunden werden können. Außerdem wird angegeben, ob der entsprechende Ort innerhalb des Spiels eine besondere Bedeutung hat. Anschließend wird die Karte codiert und im Programm gespeichert.

MÖGLICHE BEWEGUNGEN VON ORT 7
EX$(7)="08000306"





Das zweite Array, EX$(), beinhaltet Daten über die Bewegungsmöglichkeiten. Beide hier vorgestellten Spiele sind auf die Richtungen Norden, Süden, Westen und Osten begrenzt. EX$() enthält also für jede der vier Richtungen die Nummer des Ortes, zu dem man gehen kann. Die Daten bestehen aus einem String von je acht Zeichen. Die Orts-Nummer für jede Richtung wird in der Folge NESW (E für East = Osten) mittels einer zweistelligen Ziffer für jede Richtung eingegeben.

Später: Bewegungsmöglichkeiten

Nehmen wir an, Ort 7 kann in Richtung Norden, Süden und Westen, jedoch nicht nach Osten verlassen werden. Die ersten beiden Stellen von EX$(7) sind 08 (nicht nur 8), wodurch festgesetzt ist, daß sich Ort 8 im Norden befindet. Die nächsten beiden Stellen, 00, zeigen an, daß sich in dieser Richtung (Osten) kein Ausgang befindet. Die Paare 03 und 06 repräsentieren die Orte, die im Süden und Westen von Ort 7 liegen. Bei Verwendung dieses Systemes können bis zu 39 Orte bezeichnet werden. Gibt es mehr Orte, so müssen die Daten für EX$() in Gruppen mit jeweils drei Stellen eingegeben werden.

Die drei Objekte von „Haunted Forest" werden in ein anderes Array eingelesen – IV$(,). Dieses zweidimensionale Array speichert die jeweils aktuelle Position jedes Objektes, wenn es durch den Wald bewegt wird. Jedes Objekt hat eine Beschreibung und eine Startposition auf der Karte. Nehmen wir beispielsweise an, IV$(C,1) ist die PISTOLE (GUN), und die Startposition wird durch IV$(C,2) angegeben. Wird das Objekt während des Spieles herumgetragen, so wird dieser Teil des Arrays ständig aktualisiert.

Am Ende der Karten-Daten in den beiden Listings liegt noch ein weiterer Satz von Daten. Es handelt sich dabei um Prüfsummen, um sicherzustellen, daß die Richtungsdaten korrekt eingegeben wurden. Hierzu wird die Gesamtsumme aller Daten ermittelt und mit der Prüfsumme verglichen. Stimmt das Ergebnis nicht überein, liegt ein Fehler vor, und der Programmlauf wird gestoppt. Sie werden feststellen, daß bei „Digitaya" zwei Prüfsummen verwendet werden. Dies liegt daran, daß die Gesamtsumme aller Richtungsdaten zu groß wäre, um sie problemlos in einer Prüfsumme zu speichern. Daher wird jeweils für die links und rechts befindlichen vier Stellen eine Prüfsumme gebildet.

Digitaya

```
6090 REM **** READ ARRAY DATA S/R ****
6100 REM ** READ INVENTORY **
6110 DIM IV$(8,2),IC$(4)
6120 FOR C =1TO8
6130 READ IV$(C,1),IV$(C,2)
6140 NEXT C
6150 :
6160 REM ** READ LOCATION & EXIT DATA **
6170 DIM LN$(55),EX$(55)
6180 C1=0:C2=0:REM INITIALISE CHECKSUMS
6190 FOR C=1TO54
6200 READ LN$(C),EX$(C)
6210 C1=C1+VAL(LEFT$(EX$(C),4))
6220 C2=C2+VAL(RIGHT$(EX$(C),4))
6230 NEXT C
6240 READ CA:IFCA<>C1THEN PRINT"CHECKSUM ERROR":ST
OP
6250 READ CB:IFCB<>C2THEN PRINT"CHECKSUM ERROR":ST
OP
6260 RETURN
6270 REM **** INVENTORY DATA ****
6280 DATA ADDRESS NUMBER,45,KEY,34,LASER SHIELD,25
6290 DATA TICKET TO THE TRI-STATE,26,DATA CREDIT C
ARD,28
6300 DATA DIGITAYA,30,CODE BOOK,19,BUFFER ACTIVATI
NG DEVICE,13
6310 :
6320 REM **** LOCATION & EXIT DATA ****
6330 DATA IN THE TV OUTLET,00000000
6340 DATA IN THE USER PORT,00090100
6350 DATA IN THE CASSETTE PORT,00110000
6360 DATA IN THE JOYSTICK PORT,00130000
6370 DATA IN A TRI-STATE DEVICE,00170000
6380 DATA IN THE ARITHMETIC & LOGIC UNIT,00310016
6390 DATA AT THE GATEWAY TO MEMORY,00490000
6400 DATA ON THE I/O HIGHWAY,09000001
6410 DATA ON THE I/O HIGHWAY,10000802
6420 DATA ON THE I/O HIGHWAY,11000900
6430 DATA ON THE I/O HIGHWAY,12001003
6440 DATA ON THE I/O HIGHWAY,13531100
6450 DATA ON THE I/O HIGHWAY,14001204
6460 DATA ON THE I/O HIGHWAY,15001300
6470 DATA ON THE I/O HIGHWAY A SIGN SAYS 'S OUT H
,00001400
6480 DATA IN THE DATA REGISTER,00061700
6490 DATA ON AN 8 LANE HIGHWAY,16001805
6500 DATA ON AN 8 LANE HIGHWAY,17001900
6510 DATA ON AN 8 LANE HIGHWAY,18002000
6520 DATA ON AN 8 LANE HIGHWAY,19292100
6530 DATA ON AN 8 LANE HIGHWAY,20282200
6540 DATA ON AN 8 LANE HIGHWAY,21272300
6550 DATA ON AN 8 LANE HIGHWAY,22262400
6560 DATA ON AN 8 LANE HIGHWAY,23250000
6570 DATA IN THE CHARACTER MATRIX,26360024
6580 DATA HIGH IN THE MEMORY,27352523
6590 DATA IN THE MIDDLE OF MEMORY,28342622
6600 DATA IN THE MIDDLE OF MEMORY,29332721
6610 DATA LOW IN THE MEMORY,00542820
6620 DATA IN THE ACCUMULATOR'S LAIR,00000600
6630 DATA IN A LONG CORRIDOR,00420006
6640 DATA IN AN INDEX REGISTER,31000000
6650 DATA LOW IN THE MEMORY,54403428
6660 DATA IN THE MIDDLE OF MEMORY,33393527
6670 DATA HIGH UP IN MEMORY,34383626
6680 DATA IN THE CHARACTER MATRIX,35370025
6690 DATA IN A RANDOM VECTOR TABLE,00000000
6700 DATA HIGH IN MEMORY OVERLOOKING A HIGHWAY,390
03735
6710 DATA IN THE MIDDLE OF MEMORY,40003834
6720 DATA IN MEMORY - TO THE EAST IS A GATEWAY,410
03933
6730 DATA LOW IN MEMORY,00004054
6740 DATA IN A CORRIDOR,00430031
6750 DATA IN A CORRIDOR,00440042
6760 DATA IN A CORRIDOR,00004543
6770 DATA IN THE ADDRESS REGISTER,00004600
6780 DATA ON A 16 LANE HIGHWAY,45004700
6790 DATA ON A 16 LANE HIGHWAY,46004800
6800 DATA ON A 16 LANE HIGHWAY,47004900
6810 DATA ON A 16 LANE HIGHWAY A LARGE GATE LOOMS
TO THE WEST,48005007
6820 DATA ON A 16 LANE HIGHWAY,49005100
6830 DATA ON A 16 LANE HIGHWAY,50005200
6840 DATA ON A 16 LANE HIGHWAY,51000000
6850 DATA IN A VECTOR TO MEMORY,00290012
6860 DATA LOW IN MEMORY,00413329
6870 REM ** CHECKSUM DATA **
6880 DATA 100169,103973
```

Haunted Forest

```
6000 REM **** READ OBJ & MAP DATA ****
6010 DIM IV$(3,2),LN$(10),EX$(10),IC$(2)
6020 FOR C=1 TO 3
6030 READ IV$(C,1),IV$(C,2)
6040 NEXT C
6050 :
6060 FOR C=1 TO 10
6065 READ LN$(C),EX$(C)
6070 CC=CC+VAL(EX$(C)):REM CHECKSUM TOTAL
6080 NEXT C
6090 :
6100 READ CD:IFCD<>CC THENPRINT"CHECKSUM ERROR":STOP
6110 :
6120 REM ** OBJECT DATA **
6130 DATA GUN,10,LAMP,9,KEY,5
6140 :
6150 REM ** MAP DATA **
6160 DATA NEAR A TUNNEL ENTRANCE,00000900
6170 DATA IN A SWAMP,00000000
6180 DATA IN A VILLAGE,07000000
6190 DATA NEAR A TUNNEL ENTRANCE,05060000
6200 DATA ON A PATH,00020400
6210 DATA ON A PATH,02070004
6220 DATA ON A PATH,08000306
6230 DATA ON A PATH,09000702
6240 DATA ON A PATH,01100800
6250 DATA IN A CLEARING,00000009
6260 REM ** CHECKSUM DATA **
6270 DATA 32253121
6280 RETURN
```

BASIC-Dialekte

Die hier gezeigten Listings wurden für den Commodore 64 geschrieben. Bei den folgenden Computern müssen einige Änderungen durchgeführt werden:

Spectrum:

Beim Digitaya-Listing führen Sie folgende Änderungen durch:

```
6110 DIM V$(8,2,24):DIM I$(4,24)
6170 DIM L$(55,52):DIM E$(55,8)
6200 READ L$(C),E$(C)
6210 LET C1=C1+VAL(E$(C)(TO 4))
6220 LET C2=C2+VAL(E$(C)(LEN(E$(C))-3 TO))
```

Außerdem müssen sämtliche DATA-Werte in Anführungszeichen gesetzt werden. Ausnahme ist die Prüfsumme in Zeile 6880. Im Listing von „Haunted Forest" sind folgende Änderungen vorzunehmen:

```
6010 DIM V$(3,2,5):DIM L$(10,22)
6015 DIM E$(10,8):DIM I$(2,5)
6030 READ V$(C,1)V$(C,2)
6065 READ L$(C),E$(C)
6067 LET CC=0
6070 LET CC=CC+VAL(E$(C))
```

Auch hier gilt, daß alle DATA-Werte, außer der Prüfsumme in Zeile 6270, in Anführungszeichen gesetzt werden müssen.

Acorn B:

Die folgende Zeile muß in das Listing von „Haunted Forest" eingefügt werden:

```
6067 CC=0
```

Beim Digitaya-Listing sind keine Änderungen notwendig.

Robuster Rechner

Das Colour Genie von Eaca ist eine robuste, für den Heimgebrauch konstruierte Maschine. Sie bietet viele Möglichkeiten, die bei Rechnern dieser Preisklasse ungewöhnlich sind, unter anderem eine interne Stromversorgung und viele Peripherie-Schnittstellen.

Das Colour Genie ist mit einer 62 Tasten umfassenden Schreibmaschinentastatur und einem Z80 ausgestattet. Zur Tastatur gehören vier Funktionstasten, zwei Resets (die gleichzeitig gedrückt werden müssen) und eine Mode-Select-Taste, mit der vordefinierte Grafikzeichen aufgerufen werden können.

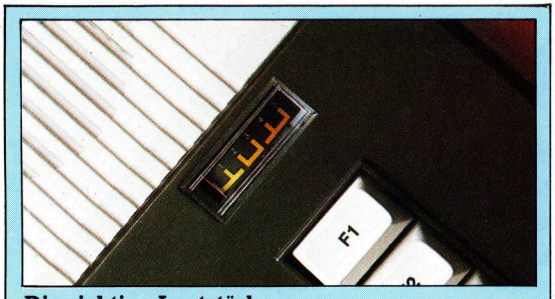
Die Maschine hat 32 KByte Speicherkapazität. Davon werden zwei für das Betriebssystem benötigt, weitere vier KByte sind für die hochauflösende Grafik erforderlich. In den 16 K ROM ist eine erweiterte Version des Microsoft-BASIC enthalten, die allerdings keine Möglichkeiten für strukturiertes Programmieren bietet, wie es bei neueren Dialekten der Fall ist.

Die Klangmöglichkeiten dieses Computers sind sehr vielseitig. Es gibt drei Kanäle (womit Akkorde spielbar sind). Die Ausgabe erfolgt über den Lautsprecher des Fernsehers. Die Klangerzeugung wird durch zwei BASIC-Befehle gesteuert. PLAY erzeugt einen vordefinierten Klang, ähnlich einem Glockenspiel. SOUND generiert andere Töne.

Die Grafikmöglichkeiten des Colour Genie sind zwar aufwendig und leistungsfähig, aber heute doch etwas überholt. Die Bildschirmdarstellung wird mit Hilfe von zwei separaten Speicherbereichen generiert. In einem sind Text, Grafikzeichenblocks und vom Benutzer definierte Grafikzeichen gespeichert, die zweite Seite wird für die Darstellung sogenannter „hochauflösender" Grafik benutzt. Im Textmode kann das Genie bis zu 25 Zeilen mit je 40 Zeichen darstellen. Im Grafikmode beträgt das Format 160 × 102 Punkte.

Mit der Mode-Select-Taste erhält man Zugang zur hochauflösenden Grafik. Das BASIC enthält zahlreiche Grafik-Befehle. Man kann Linien zeichnen, Flächen komplett mit Farbe füllen sowie Konturen definieren, zeichnen und löschen. In ein BASIC-Programm integriert, stellt man mit dem FGR-Befehl die Grafikseite dar. Am Ende des Programms kehrt der Rechner aber automatisch in den Text-Mode zurück. Das BASIC umfaßt außerdem Befehle zum Löschen der Grafikseite (FCLS) sowie zur Veränderung der Hintergrund-(FILL) und der Vordergrund-Farben (FCOLOR). Dieses Verfahren ist zwar umständlicher als die Einzelseiten-Verarbeitung bei den meisten

Obwohl es das Colour Genie schon sehr lange gibt, hat dieser Computer nie die Popularität erreicht, die der Spectrum oder der C 64 haben. Dennoch gibt es eine Reihe getreuer Anwender. Die Maschine hat 32 KByte Speicher und ist mit ungewöhnlichen Joysticks ausgestattet, die paarweise geliefert werden.



Die richtige Lautstärke
Das Einstellen eines Cassettenrecorders auf die für den Computer richtige Lautstärke kann extrem schwierig sein. Das Colour Genie hat dafür eine eigene Lautstärkeregelung.

neueren Rechnern, erlaubt aber die individuelle Einfärbung jedes Punktes.

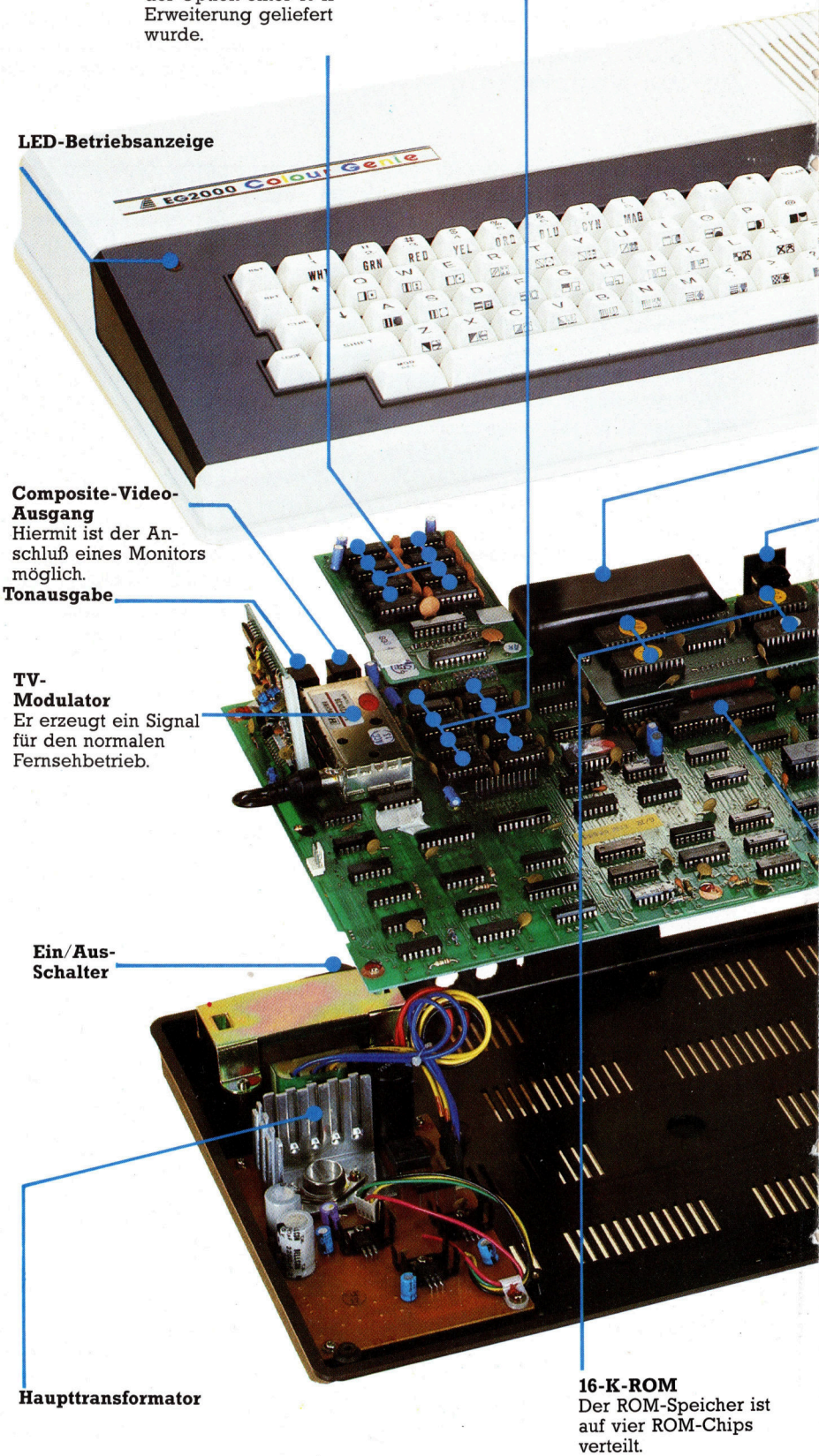
Die Bildschirmdarstellung ist deutlich und stabil, doch durch den verwendeten Zeichensatz ist Text vergleichsweise schwer lesbar. Das Genie verfügt über acht Farben – weiß, rot, grün, gelb, cyan, magenta, blau und orange –, die auf dem Textschirm gleichzeitig dargestellt werden können. Bei Benutzung der hochauflösenden Grafik stehen dem Anwender nur vier Farben (rot, blau, grün und schwarz) zur Verfügung. Es gibt aber einen weiteren Befehl (BGRD), mit dem der Hintergrund der Grafikseite rosa gefärbt werden kann.

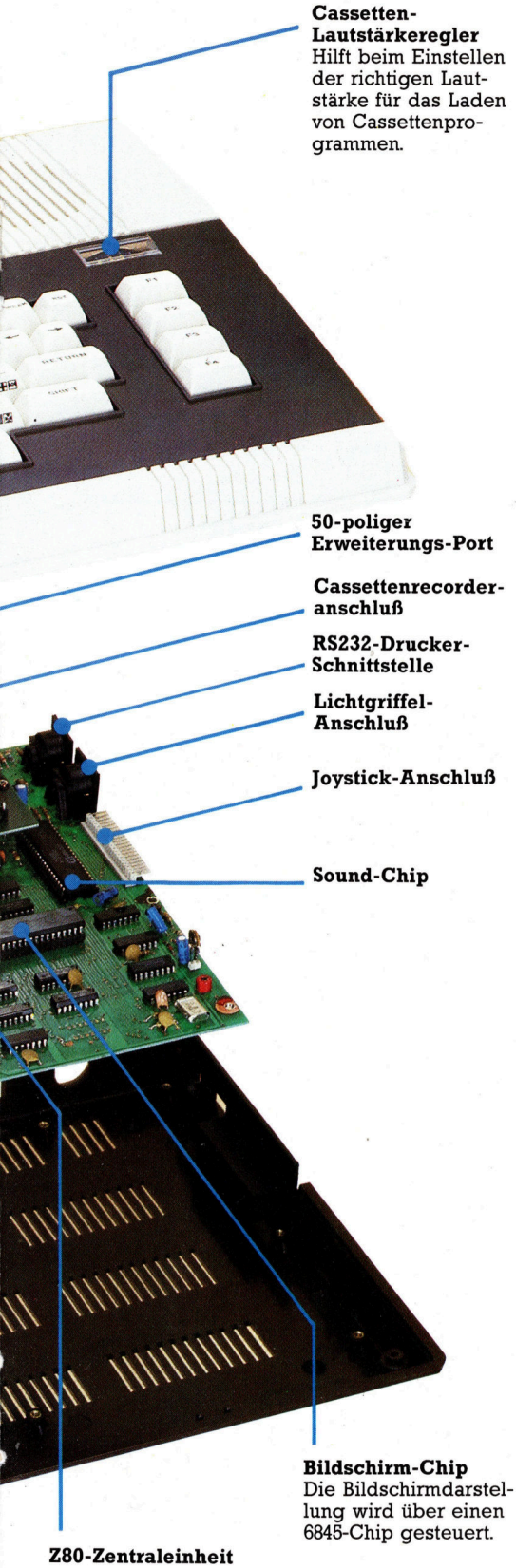
Interessante Joysticks

Der Rechner hat mehrere Schnittstellen: ein RS232-Port für Drucker und Modems, einen 50-poligen Erweiterungsport für den Anschluß von Diskettenstationen, einen Composite-Video-Ausgang, einen Audio-Ausgang, einen Lichtgriffel-Anschluß sowie einen Joystick-Port. An Peripherie stehen zur Verfügung: ein Centronics-Drucker-Interface, Joysticks, ein Prestel-Steckmodul (wozu ein Modem erforderlich ist) und Diskettenlaufwerke. Die Doppel-Joysticks haben eine integrierte Zehnertastatur, sind aber schwer zu benutzen. Für die Steuerung ist erheblicher Kraftaufwand nötig. Läßt man den Joystick los, kehrt er nicht selbständig in die Ausgangsposition zurück.

Eaca, die Firma, die den Colour Genie herstellt, bietet selbst keine Diskettenstation für den Rechner an. Man kann jedoch die Floppy von einem britischen Unternehmen beziehen. Es wird ein als QDOS bezeichnetes Betriebssystem verwendet, das dem TRS-DOS von Tandy ähnlich ist.

Das Colour Genie ist mit einem Lautstärkeregler ausgestattet, womit Ladeprobleme bei Cassetten reduziert werden. Der Benutzer justiert lediglich die Lautstärke, bis die Nadel genau in der Skalenmitte steht. Der Ladevorgang läuft dann exakt ab. Zusätzlich kann ein sogenannter „Daten-Stabilisator" zwischen Cassettenrecorder und Interface geschaltet werden. Dadurch wird das Eingangssignal





beim Bandbetrieb „gereinigt".

Das Colour Genie wird mit zwei Handbüchern geliefert – einer allgemeinen Anleitung für Anfänger und einem BASIC-Handbuch. Beide sind leicht verständlich geschrieben, es mangelt aber an Einzelheiten. Fortgeschrittene Anwender können gegen einen Aufpreis ein technisches Handbuch beziehen.

Trotz seines etwas antiquierten Erscheinungsbildes bietet das Colour Genie als Heimcomputer viel fürs Geld.

Martian Raider

Es gibt wenig Software für das Colour Genie, doch die Qualität der verfügbaren Programme ist sehr gut. Hauptsächlich stehen Spiele zur Verfügung, wobei es sich um übersetzte Versionen von Spielen handelt, die für populäre Rechner geschrieben wurden.

Colour Genie

ABMESSUNGEN
90 × 280 × 340 mm

ZENTRALEINHEIT
Z80, 2,2MHz

SPEICHERKAPAZITÄT
32 K RAM, 16 K ROM

BILDSCHIRM-DARSTELLUNG
Bis zu 25 Zeilen mit 40 Zeichen Text, Grafik bis zu 160 × 102 Pixeln, 8 Farben im Text-Mode, 4 Farben im Grafik-Mode.

SCHNITTSTELLEN
Joysticks (2), RS232-Schnittstelle, Lichtgriffelanschluß, Erweiterungsanschluß.

PROGRAMMIERSPRACHE
BASIC (mitgeliefert)

TASTATUR
62 Schreibmaschinentasten einschließlich vier Funktionstasten.

HANDBÜCHER
Der Rechner wird mit einer Einführung für den Einsteiger und einem Grundhandbuch geliefert.

STÄRKEN
Der Colour Genie ist ein guter „Familien"-Rechner. Er ist robust gebaut, verwendet Microsoft-BASIC, bietet recht gute Grafik und guten Klang, der über den Fernseher ausgegeben wird.

SCHWÄCHEN
Die Konstruktion des Genie ist überholt. Der Prozessor ist zu langsam, und es gibt nur wenig Software für das System.

Colour Genie Joysticks
Die Joysticks des Colour Genie sind attraktiv, aber teuer und im Umgang problematisch. Für die Bewegung ist sehr viel Kraft erforderlich, und sie kehren nicht automatisch in die Ausgangsposition zurück. Ungewöhnlich ist die numerische Tastatur.

Im Einsatz

In der letzten Folge wurde gezeigt, daß PASCAL keine Funktionsanweisungen zuläßt, da Funktionen Werte berechnen und ihre Aufgaben sich von denen der Prozeduren unterscheiden. In diesem Artikel sehen wir uns an, wie sie eingesetzt werden.

Funktionen lassen sich zwar wie Prozeduren mit Namen aufrufen, können jedoch nur in Gleichungen eingesetzt werden. Während ein Prozeduraufruf den Ablauf eines Unterprogramms auslöst, liefert der Aufruf einer Funktion den Ergebniswert einer Berechnung. Dieser Wert kann ein einfacher Typ sein (beispielsweise eine reale Zahl), ein Skalar oder ein „Pointer". Der Name einer Funktion unterscheidet sich von dem einer Prozedur nur durch das Wort FUNCTION und die Typenbezeichnung für den Ergebniswert. Die Funktion „odd" beispielsweise könnte man auch selbst definieren:

```
FUNCTION Odd (Nummer : integer) : boolean;
  BEGIN
    Odd := Nummer MOD 2 > 0
  END; (* Odd *)
```

Da das Ergebnis einer Funktion in ihrem Namen gespeichert wird, muß der Funktionsname den Typ des Ergebniswertes enthalten. Funktionsnamen lassen sich daher als Variablen ansehen, die zwar nie initialisiert wurden, deren Wert jedoch berechnet wird, wenn der Name in einer Gleichung auftaucht. Funktionsnamen können nur auf der linken Seite einer Zuordnung auftreten. Wäre die gleiche Funktion nachlässig programmiert, wie:

```
IF Nummer MOD 2 > 0 THEN
  Odd := true
```

(der Fall „ELSE Odd := false" fehlt) besteht die Möglichkeit, daß das Ergebnis nicht definiert ist.

Wie bei Prozedurparametern wird bei Aufruf einer Funktion der Wert des aktuellen Parameters an lokale Variablen (hier die Ganzzahl „Nummer") übergeben. So führt folgende Anweisung

```
WriteLn ( Odd ( sqr ( N DIV 100) ) )
```

(die immer „false" ergibt) vier Abläufe aus:

1. der Ausdruck „N DIV 100" wird berechnet;
2. das Ganzzahlergebnis wird an „sqr" als aktueller Parameter übergeben;
3. das Quadrat dieses Parameters wird an „Odd" übergeben;
4. das Ergebnis wird auf boolesche Weise bewertet und an die WriteLn-Prozedur wiederum als Wertparameter übergeben.

Der Wert N wird durch die Ausführung der Anweisung natürlich nicht verändert, da ein Funktionsergebnis ist immer nur „Funktion" der übergebenen Parameterwerte sein kann. Da PASCAL nicht die eigentlichen Variablen, sondern nur Kopien der Variablenwerte an Funktionen und Prozeduren weitergibt, ist sichergestellt, daß die ursprünglichen Parameter sich nicht ändern, selbst wenn ihre Werte im Inneren einer Funktion oder Prozedur umgewandelt werden.

Globale Konstanten

Zwar kann unter PASCAL direkt auf globale Daten zurückgegriffen werden, doch empfehlen wir, davon so wenig wie möglich Gebrauch zu machen. Alle Datenübergaben bei Prozedur- oder Funktionsaufrufen sollten von Parameterlisten gesteuert werden, selbst wenn die Daten innerhalb des Unterprogrammbereichs verfügbar sind. Die einzige Ausnahme zu dieser generellen Regel sind globale Konstanten, die in PASCAL per Definition nicht verändert werden können. Beachten Sie aber, daß konstante Werte, die als Parameter übergeben werden, zu lokalen Variablen werden und daher nicht länger geschützt sind.

```
FUNCTION Klein ( Zeichen : char ) : char;
  (* wandelt als Argument uebergebene
  Grossbuchstaben in Kleinbuchstaben um *)
  CONST
    Offset = 32; (* ASCII ord('a') — ord('A' )*)
  BEGIN
    IF Zeichen IN [ 'A'. .'Z']
      THEN
        Klein := chr ( ord ( Zeichen )
                 + Offset )
      ELSE
        Klein := Zeichen
  END; (* Klein *)
```

Es kann der Fall eintreten, daß Prozeduren Werte von globalen Variablen doch verändern müssen, da ihre Aufgabe sonst hinfällig wäre. Ein gutes Beispiel dafür ist die Prozedur „read". Wenn read (N) die eingegebene Ganzzahl nur an eine lokale Variable weitergeben würde, hätte das für uns wenig Wert. Wir müssen daher die Adresse des Variablenparameters (und nicht nur den Variablenwert) übergeben, damit die Prozedur sich direkt darauf beziehen kann und nicht nur eine Kopie bearbeitet. Dieser Mechanismus sollte jedoch nur mit Prozeduren eingesetzt werden.

Die Übergabe eines Variablenparameters wird in der Parameterliste der Prozedur durch

das reservierte Wort „VAR" vor dem Variablennamen angezeigt. Der Aufbau der Parameterliste ist mit der VAR-Deklaration eines Blocks identisch, allerdings brauchen im Prozedurkopf nur die Variablennamen angegeben zu werden, die von der Prozedur verändert zurückgegeben werden sollen:

```
PROCEDURE Ablauf ( VAR Zaehler :
    ZaehlerListe );
```

Um diese vielseitig einsetzbare Technik deutlich zu machen, werden wir ein Programm entwickeln, das über die Tastatur Namen einliest und zu jedem Namen einen Geldbetrag stellt, den diese Person schuldet. Zur Vereinfachung nehmen wir einen String pro Namen und Ganzzahlen als Geldbeträge. Nachdem das Programm getestet wurde, können Sie Adressen, Telefonnummern etc. hinzufügen. Als Ergebnis wollen wir eine Liste erzeugen – in alphabetischer Reihenfolge oder nach Beträgen geordnet. Als Datenstruktur bietet sich eine Liste von „Records" an, deren Felder sich später leicht erweitern lassen. Wir können dabei jeden Record einzeln ansprechen, aber auch jedes einzelne Feld als Sortierschlüssel verwenden. Zunächst die Datendefinition:

```
CONST
    StringLaenge   = 20;
    ListenLaenge   = 50;
TYPE
    Cardinal       = 0. .MaxInt;
    StringGroesse  = 1. .StringLaenge;
    String         = PACKED ARRAY
                     [ StringGroesse ] OF char;
    Daten          = RECORD
                       Name     : string;
                       Schulden : Cardinal;
                     (* weitere Felder moeglich *)
                       END; (* Daten *)
    Grenze         = 1. .ListenLaenge;
    RecordListe    = ARRAY [Grenze] OF Daten;
```

Wir können zwar nur fünfzig Namen mit nicht mehr als 20 Zeichen eingeben, diese Angaben lassen sich jedoch leicht ändern.

Aus folgenden Gründen haben wir die Unterbereiche und Strukturen mit TYPE definiert:

1. Sicherheit – ein Index vom Typ „Grenze" kann nie die festgelegten Grenzen eines Arrays überschreiten.

2. Fehlersuche – sollte während des Testens ein Bereichsfehler auftreten, läßt sich die Ursache leicht feststellen.

3. Effektivität – lokale Variablen sparen Speicher und vermeiden überflüssige Doppelvergaben.

4. Notwendigkeit – PASCAL „besteht" darauf, daß als Parameter nur Namen und keine Bezeichnungen wie 0. .255 eingegeben werden.

Als nächstes müssen wir nun die Variablen deklarieren und einen Algorithmus aufbauen. Diesen Ablauf behandeln wir in der nächsten Folge. Entwickeln Sie als Übung ihren eigenen Programmalgorithmus, und stellen Sie die entsprechenden Prozeduren zusammen.

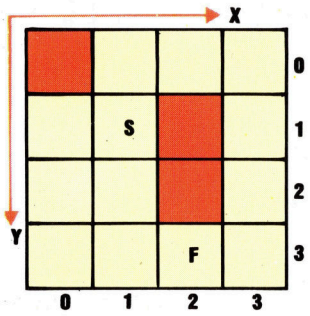
Geprüfte Eingaben

Das Programm „Karten-Leser" ist eine sichere Methode, positive Zahlen einzulesen. Dabei werden die Daten als Zeichen eingegeben. Die Umwandlung in den entsprechenden Zahlenwert geschieht durch die Subtraktion des Ziffernwertes von der Ziffer „0". Da die Zeichen 0. .9 lückenlos definiert sind, funktioniert das Programm bei allen Zeichensätzen. Beachten Sie, daß das Programm nur zwei globale Variablen besitzt.

Da die Daten für die Zeichenverarbeitung und die Funktion „Wert" nur von der Prozedur „Karte-Einlesen" benötigt werden, wurden diese Komponenten auch nur für diese Prozedur definiert.

In dem Programm gibt es nur einen einzigen möglichen Fehler – bei der Eingabe eines Wertes größer als „MaxInt" stürzt der Zuordnungsbefehl der WHILE-Schleife ab.

```
PROGRAM          Kartenleser          ( input, output );

TYPE
        Cardinal          = 0. . MaxInt;

VAR
        PosNum            : Cardinal;
        InOrdnung         : boolean;

  (* ———————————— Ebene 1 ———————————— *)

PROCEDURE KarteEinlesen  ( VAR N   : Cardinal;
                           VAR OK  : boolean );
  CONST
        Leerzeichen       = ' ';
  TYPE
        Einfach           = 0 . . 9;

  VAR
        Symbol            : char;
        Ziffern           : SET OF char;

  (* ———————————— Ebene 2 ———————————— *)

  FUNCTION Wert ( Ziffer : char )                  : Einfach;
        (* liefert den Zahlenwert einer
           Ziffer jedes Zeichensatzes *)
      BEGIN
        Wert := ord ( Zeichen ) — ord ('0' )

      END; (* Wert *)
  (* ———————————— Ebene 2 ———————————— *)
                  (* Zurueck zu Ebene 1 *)
  BEGIN (* KarteEinlesen *)

      REPEAT
        read ( Symbol )

      UNTIL Symbol > Leerzeichen;

      Zeichen := [ '0' . . '9' ];

      OK    := Symbol IN Zeichen;

      IF OK THEN
        BEGIN
          N := 0;

          WHILE Symbol IN Zeichen DO
            BEGIN (*berechnen auf Basis 10 *)
              N := 10 * N + Wert ( Symbol );
              read ( Symbol )
            END (* Begrenzungszeichen oder 'EoLn' *)
          END    (* hinter der Zahl beseitigen *)

  END; (* KarteEinlesen *)
  (* ———————————— Ebene 1 ———————————— *)
BEGIN (* KartenLeser — Ebene 0    Hauptprogramm *)

  page ( output );
  WriteLn;
  WriteLn ( ' Dieses Programm liest positive' );
  WriteLn ( ' Ganzzahlen ( im Bereich 0 . . ',
                                MaxInt : 1, ')' );
  WriteLn ( 'Das Programmende wird von der' );
  WriteLn ( 'Fehlererkennung gesteuert.' );
  WriteLn;
  write ( 'Geben Sie eine Zahl ein :  ' );
  KarteEinlesen ( PosNum, InOrdnung );

  WHILE InOrdnung DO
    BEGIN
      WriteLn;
      write ( 'Die eingegebene Zahl war : ' );
      WriteLn ( PosNum : 1);
      Write ( 'Zahl ? ' );
      KarteEinlesen ( PosNum, InOrdnung )

    END;

  WriteLn ( '——— FEHLER ERKANNT ———' : 40 )

END.
```



Im Labyrinth

In früheren Abschnitten unseres Selbstbau-Kurses haben wir bereits die Soft- und Hardware zur Steuerung eines zweimotorigen Fahrzeugs entwickelt. Dazu soll jetzt ein „intelligentes" Programm entstehen, das den kürzesten Weg durch ein Labyrinth findet.

0	1	2	3
4	5 S	6	7
8	9	10	11
12	13	14 F	15

Das Labyrinth-Programm wandelt die Koordinaten-Darstellung um. Sie wird aus DATA-Zeilen in ein zweidimensionales Feld übernommen, das auch die Koordinaten von Start- und Zielpunkt enthält. Um den richtigen Weg zu finden, wird jedes einzelne Quadrat als Verästelung in einem Baum betrachtet. Dabei werden nicht die ursprünglichen Koordinaten verwendet, sondern eine Numerierung in festgelegter Reihenfolge, die in der oberen linken Ecke beginnt.

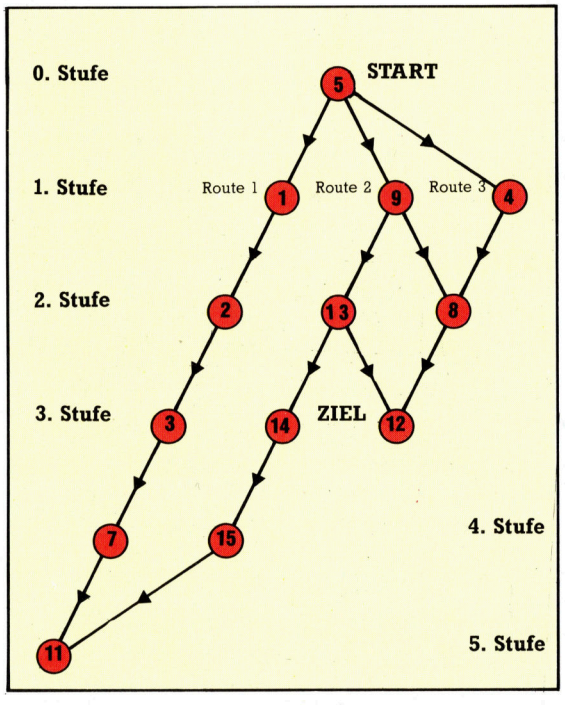
Um ein Labyrinth zu entwerfen, brauchen wir zuerst einmal eine geeignete Fläche. Das kann etwa eine Tischplatte oder der Fußboden sein. Das Gebiet wird nun in Quadrate eingeteilt, deren Größe von den Abmessungen des Fahrzeugs abhängt. Eine Drehung um 360 Grad sollte innerhalb jedes Abschnitts problemlos möglich sein. Entwerfen Sie zuerst ein Netz aus Quadraten, und markieren Sie die einzelnen Wege durch Bücher, Tassen oder kurze Holzstücke, so daß daraus ein einfaches Labyrinth entsteht.

Für das Programm müssen Sie die Größe des Labyrinths und die Lage der freien und belegten Quadrate angeben. Am besten geht das mit einem Binärcode: Die 1 bedeutet, daß ein Quadrat teilweise oder ganz von einem Gegenstand ausgefüllt ist, 0 heißt, das Quadrat ist frei. Damit diese Angaben nicht bei jedem Programmablauf erneut eingegeben werden müssen, werden sie als DATA-Anweisungen festgehalten. Die letzten vier Daten geben die Koordinaten des Start- und Zielpunktes an. Wir legen den Ausgangspunkt des Koordinatensystems in die obere linke Ecke. Damit ist die oberste Zeile Reihe 0 und die linke Seite die Spalte 0. Unser Labyrinth wird mit diesen DATA-Anweisungen beschrieben:

```
DATA 4,4: REM LABYRINTH
DATA 1,0,0,0,0,0,1,0
DATA 0,0,1,0,0,0,0,0
DATA 1,1: REM STARTKOORDINATEN
DATA 2,3: REM ZIELKOORDINATEN
```

Besonders schwierig ist es nicht, den Weg durch das Labyrinth zu finden. Wir können ein Programm entwickeln, das eine Route entwirft, die am Start beginnt und Sackgassen bzw. falsche Wege wieder verläßt. Das Fahrzeug wird dann so lange umherfahren, bis es zufällig am Ziel ankommt. Der Weg (ohne falsches Abbiegen in Sackgassen), der sich dabei ergibt, muß nicht unbedingt der kürzeste sein. Um diesen zu erreichen, brauchen wir ein Programm, das alle möglichen Wege zwischen Start und Ziel prüft und den besten herausfindet. Als „optimaler Weg" gilt in diesem Fall eine möglichst niedrige Zahl durchquerter Quadrate.

Die Überprüfung jeder einzelnen Route kann durch Einführung einer Technik vereinfacht werden, welche die Beziehungen zwischen den Quadraten berücksichtigt. Dafür bietet es sich an, die Daten in Form eines Baumes zu strukturieren. Vom Start als der „Baumwurzel" ausgehend, ergibt sich eine Gruppe von benachbarten Quadraten. Aus dieser ersten Verästelung läßt sich eine zweite ableiten und so fort. Zu jedem Labyrinth kann ein solcher Baum gezeichnet werden. Dazu werden alle Quadrate vom Start ausgehend von links nach rechts in der Reihenfolge Nord, Ost, Süd, West (vom Ausgangspunkt her gesehen) durchnumeriert.



Um den Weg aus dem Labyrinth zu optimieren, muß zuerst ein Baum konstruiert werden, der die Beziehung der einzelnen Quadrate zueinander beschreibt. Jede Gabelung des Baumes wird nacheinander bis zur letzten Verästelung durchlaufen. Gabelungen der ersten Stufe sind dem Start-Quadrat benachbart, die nächsten Gabelungen gehören zur zweiten Stufe und so fort. Verglichen mit der Programmierung in BASIC ist das Zeichnen des Baumes geradezu ein Kinderspiel.

Verästelungen ausprobieren

Wenn kein Schritt wiederholt wird, gibt es genau fünf Wege durch unser einfaches Labyrinth. Drei Möglichkeiten sind oben abgebildet, sowohl als Weg durch den Baum als auch im tatsächlichen Labyrinth. Daraus läßt sich erkennen, daß Route 2 die kürzeste ist. Ein Computer dagegen arbeitet den Baum so lange linear Ast für Ast durch, bis er in eine Sackgasse oder ans Ziel kommt. Für den zweiten Fall müssen die Daten des richtigen Weges gespeichert werden, im ersten Fall muß der eingeschlagene Weg als Sackgasse markiert sein, bevor der Rechner seine Suche vom Startpunkt erneut beginnt. Das Programm startet immer wieder von vorn, bis alle Verästelungen ausprobiert worden sind.

BASIC ist für derartige Suchalgorithmen nicht optimal – das Programm wird Ihnen daher vielleicht etwas umständich vorkommen. Sprachen wie LOGO oder ALGOL eignen sich besser für diese Zwecke. Das Programm muß zwei Aufgaben ausführen: die Erstellung des Baumes aus den Daten des Labyrinths und die Programmierung von vier Zeigern, welche die jeweils benachbarten Quadrate angeben. Das Zeigersystem kann am besten mit einem zweidimensionalen Feld TR(N,D) realisiert werden, in dem N die Nummer des Quadrats und D eine der Richtungen 1 bis 4 darstellt. In unserem Labyrinth wäre TR(9,1) 5 – das Quadrat, das nördlich von Quadrat 9 liegt. Ist eine Richtung versperrt bzw. eine Grenze des Labyrinths, wird ihr ein spezieller Wert – etwa (–1) – zugeordnet.

Beim „Abarbeiten" des Baumes wird die gewählte Route in einem Pseudo-Stack gespeichert, der aus einem eindimensionalen Feld aufgebaut ist. Die Variable D definiert das nächste erreichbare Element des Stacks. Der jeweils kürzeste gefundene Weg wird außerdem in einem weiteren eindimensionalen Feld festgehalten, dessen erstes Element die Anzahl der benötigten Schritte angibt.

Nachdem das Programm den Baum vollständig durchlaufen hat, verfügt es über die Beschreibung der optimalen Route in Form einer Reihe von Quadrat-Nummern. Steht unser Fahrzeug nun in nördlicher Richtung auf dem Startfeld, kann die Fahrtrichtung leicht durch Differenzbildung aus den im Speicherfeld aufeinander folgenden Quadrat-Nummern berechnet werden. In unserem Beispiel heißt eine Differenz von +4 „nach Norden", eine Differenz von –4 „nach Süden". Vor dem Weiterfahren zum nächsten Quadrat ist der Drehwinkel des Fahrzeugs zu berechnen. Da hier einfache Gleichstrommotoren für den Antrieb benutzt werden, ergeben sich Drehwinkel und Fahrtstrecken aus dem Zeitintervall, für das eine bestimmte Motoren-Kombination eingeschaltet wird. Sie müssen daher vor der praktischen Erprobung die Zeit ermitteln, die für eine Drehung um 90 Grad bzw. zum Vorrücken um ein Quadrat nötig ist. Diese Informationen werden in den Variablen AF und FF gespeichert, für den Acorn B Hundertstelsekunden, beim Commodore 64 Sechzigstelsekunden.

Wege aus dem Irrgarten

```
900 REM ******************
910 REM ******************
920 REM **                **
930 REM ** CBM 64 MAZE   **
950 REM **                **
960 REM ******************
970 REM ******************
980 :
990 REM **** MAIN CALLING PROGRAM ****
1000 GOSUB 1600:REM READ MAZE DATA
1100 GOSUB3700:REM PRINT MAZE
1200 GOSUB5400:REM CONSTRUCT TREE
1210 GOSUB7700:REM TRAVERSE TREE
1215 IF S(0)=9999 THEN PRINT "NO SOLUTION":END
1220 GOSUB8200:REM DIRECT VEHICLE
1400 END
1500 :
1600 REM **** READ MAZE DATA/INITIALISE ****
1700 READ SX,SY
1800 DIM MZ(SX,SY),TR(SX*SY,4),DR(4),RT(SX*SY),
     LN(SX*SY),CN(SX*SY)
1900 FOR Y=0 TO SX-1
2000 FOR X=0 TO SX-1
2100 READ A:MZ(X,Y)=A
2200 NEXT X,Y
2300 :
2400 READ XS,YS,XF,YF
2500 DATA 4,4
2600 DATA 1,0,0,0,0,0,1,0
2700 DATA 0,0,1,0,0,0,0,0
2800 DATA 1,1:REM START COORDS
2900 DATA 2,3:REM FINISH COORDS
3000 :
3100 DR(1)=-SX:DR(2)=1:DR(3)=SX:DR(4)=-1
3110 ID(1)=3:ID(2)=4:ID(3)=1:ID(4)=2
3120 SR(0)=9999:REM INIT SHORTEST ROUTE
3130 DDR=56579:DATREG=56577:POKE DDR,255
3140 AF=30:FF=45:REM ANGLE AND FWRD TIME FACTORS
3200 FOR I=1 TO 25:CD$=CD$+CHR$(17):NEXT I
3500 RETURN
3600 :
3700 REM **** PRINT MAZE ****
3800 PRINTCHR$(147):REM CLEAR SCREEN
3900 FOR X=0 TO SX-1
4000 FOR Y=0 TO SY-1
4100 GOSUB 4900:REM POSITION CURSOR
4200 PRINT CHR$(32+MZ(X,Y)*134)
4300 NEXT Y,X
4400 :
4500 X=XS:Y=YS:GOSUB4900:PRINT"S"
4600 X=XF:Y=YF:GOSUB4900:PRINT"F"
4700 RETURN
4800 :
4900 REM **** POSITION CURSOR AT X,Y ****
5000 PRINT CHR$(19);:PRINTTAB(X)LEFT$(CD$,Y);:RETURN
5300 :
5400 REM **** CONSTRUCT TREE ****
5500 :
5600 REM ** INITIALISE WITH TERMINATORS ****
5700 FOR P=0 TO SX*SY-1:FORI=1TO4:TR(P,I)=-1:NEXT I,P
6100 :
6110 REM ** CALCULATE START & FINISH ****
6120 X=XS:Y=YS:GOSUB9200:S=N
6130 X=XF:Y=YF:GOSUB9200:F=N
6140 :
6200 REM ** CONSTRUCT **
6300 LC=1:CC=0:REM INIT STACK PTRS
6400 LN(LC)=S:REM START POINT
6450 CN=LN(LC):REM GET CURR NODE OFF STACK
6500 DF=0
6600 FOR D=1 TO 4
6700 N=CN+DR(D):GOSUB 8900:REM CONVERT TO X,Y
6800 IF(X<0 OR X>SX-1 OR Y<0 OR Y>SY-1)THEN 7300
6900 IF MZ(X,Y)=1 THEN 7300
7000 IF(D=2 AND N/SX=INT(N/SX))THEN7300
7100 IF(D=4 AND CN/SX=INT(CN/SX))THEN7300
7200 IF TR(N,ID(D))=CN THEN 7300
7210 TR(CN,D)=N:DF=1
7220 CC=CC+1:CN(CC)=N:REM PUSH ONTO CURRENT STACK
7300 NEXT D
7310 IF(DF=0 AND LC=1) THEN RETURN:REM TERMINAL NODE
7320 :
7330 LC=LC-1:REM DEC LAST STACK PTR
7340 IF LC>0 THEN 6450:REM NEXT NODE
7345 :
7350 REM ** COPY CURR. STACK TO LAST STACK **
7360 FOR I=1 TO CC:LN(I)=CN(I):NEXT I
7390 LC=CC:CC=0:GOTO6450:REM NEXT NODE
7600 :
7700 REM **** TRAVERSE TREE ****
7720 C=0:RN=S:CN=RN:EF=0
7730 C=C+1:RT(C)=CN
7740 IF CN=F THEN GOSUB8100:GOSUB8000:IF EF=0 THEN 7730
7745 IF EF=1 THEN RETURN
7750 DF=0
7760 FOR D=1 TO 4
7770 IF TR(CN,D)<>-1 THEN CN=TR(CN,D):DF=1:DR=D:D=4
7780 NEXT D
7790 IF DF=0 THEN GOSUB8000
7800 IF EF=0THEN 7730
7810 IF EF=1 THEN RETURN
7820 :
8000 REM **** RESTART AT ROOT ****
8010 TR(RT(C-1),DR)=-1
8020 CN=RN:C=0
8030 IF(TR(CN,1)ANDTR(CN,2)ANDTR(CN,3)ANDTR(CN,4))=-1 THEN EF=1
8040 RETURN
8050 :
8100 REM **** SAVE ARRAY ****
8110 IF C>SR(0) THEN RETURN:REM IS NEW ROUTE SHORTER?
8120 SR(0)=C
8130 FOR I=1 TO C:SR(I)=RT(I):NEXT I:RETURN
8140 :
8200 REM **** DIRECT VEHICLE ****
8205 PD=1:REM ASSUME INITIAL
     DIRECTION NORTH
8210 FOR C=1 TO SR(0)-1
8220 DF=SR(C+1)-SR(C)
8222 :
8225 REM ** FIND REQUIRED DIRECTION **
8230 FOR I=1 TO 4
8240 IF DF=DR(I) THEN D=I:I=4
8250 NEXT I
8260 :
8265 DR=D-PD:PD=D
8270 H=INT(4+DR/4):R=(4+DR)-4*H
8275 :
8277 REM ** DO TURN **
8280 FOR I=1 TO R
8290 POKE DATREG=9:REM CLOCKWISE TURN
8300 T=TI
8310 IF (TI-T)<AF THEN 8310:REM WAIT
8320 POKE DATREG=0:REM OFF
8330 NEXT I
8340 :
8350 REM ** FORWARD **
8360 POKE DATREG=5
8370 T=TI
8380 IF (TI-T)<FF THEN 8380
8390 POKE DATREG=0
8400 NEXT C
8410 :
8420 RETURN
8430 :
8900 REM **** CONVERT N TO X,Y ****
9000 Y=INT(N/SX):X=N-SX*Y:RETURN
9200 :
9210 REM **** CONVERT X,Y TO N ****
9220 N=Y*SX+X:RETURN
```

Für den Acorn B
Diese Zeilen müssen Sie ändern:

```
3130 DDR=&FE62:DATREG=&FE60

8290 ?DATREG=9
8300 TIME=0
8310 REPEAT UNTIL TIME>=AF
8320 ?DATREG=0

8360 ?DATREG=5
8370 TIME=0
8380 REPEAT UNTIL TIME>=FF
8390 ?DATREG=0
```

Shadowfire

Ihnen bleiben genau 100 Minuten, um einen Botschafter, der wichtige Pläne mit sich führt, von dem Raumschiff zu retten, auf dem man ihn gefangenhält. Ein Team von sechs Wesen hilft Ihnen dabei.

Selbst gelegentliche Beobachter der Szene werden bemerken, daß Computerspielprogramme immer ausgefeilter werden. In den neueren Programmen sind viele der Elemente aus Reaktions- und Strategiespielen kombiniert. So entstehen Unterhaltungsprogramme, die wesentlich länger Spannung vermitteln als die fünf Minuten währenden Spielhallenspiele und herausfordernder sind als die „hirnzermarternden" Abenteuerprogramme.

Hintergrund bei „Shadowfire" ist, daß der abtrünnige General Zoff Botschafter Kryxix nebst den Plänen für ein neues Raumschiff entführt hat. Sie, der Spieler, wurden ausgesucht, den Botschafter zu befreien, bevor er zur Übergabe der Pläne gezwungen werden kann. Zur Durchführung Ihrer Aufgabe haben Sie genau 100 Minuten. Unterstützt werden Sie bei dieser Rettungsaktion von sechs Wesen, die über unterschiedliche Stärken und Schwächen verfügen.

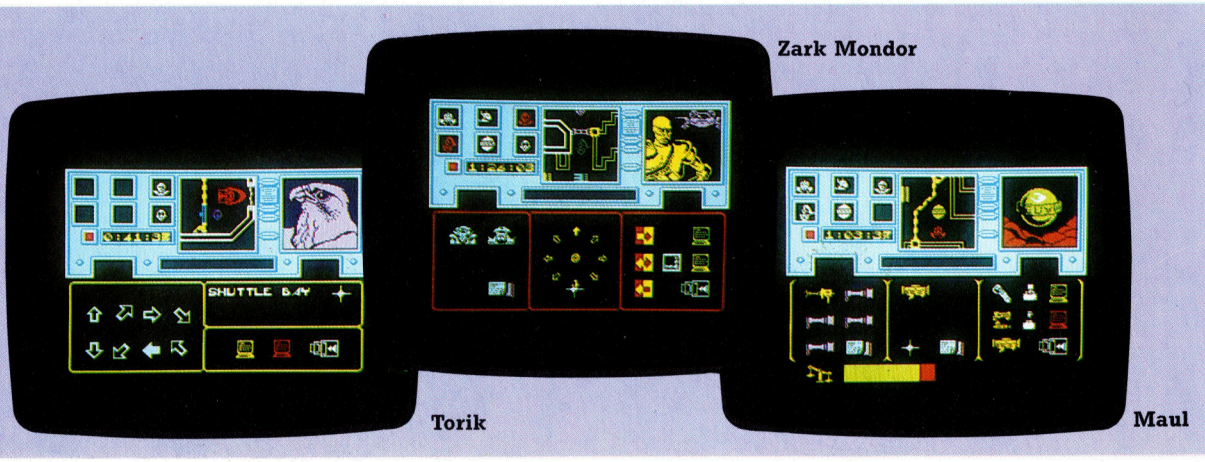
Bei Shadowfire findet ein einzigartiges Verfahren zur Bewegung der Charaktere Anwendung, durch das es ihnen möglich ist, Gegenstände aufzunehmen und zu kämpfen. Anders als bei normalen Abenteuer-Programmen, bei denen der Spieler Befehle wie „Go North" oder „Take Laser" eintippen muß, werden hier alle Funktionen über Piktogramme und einen frei beweglichen Cursor ausgeführt – in etwa dem Betriebssystem entsprechend, das beim Apple-Macintosh Anwendung findet. Um etwa Zark, den Teamchef, mit Handgranaten auszurüsten, wird zunächst das Zark-Programm ausgewählt. Daraufhin werden seine Stärke, Ausstrahlung und andere Eigenschaften in Form von Symbolen auf dem Bildschirm gezeigt. Auf der rechten Bildschirmseite befinden sich drei „Monitor"-Piktogramme, die Bewegung, Kampfstatus und den Objekt-Screen darstellen.

Wählt man nun mittels Cursor (gesteuert wahlweise über Tastatur, Joystick oder Lichtgriffel) den Objekt-Schirm, verändert sich wiederum dieser und zeigt die Gegenstände, die sich in der Nähe des gewählten Charakters befinden, und eine Anzahl von „Aktions"-Piktogrammen. Durch Wahl des „Nimm auf"-Piktogramms und anschließende Steuerung des Cursors auf das Granaten-Piktogramm ist Zark nunmehr mit Granaten ausgerüstet.

Nach Bewaffnung und Teleportation auf das Raumschiff kann das Team nun mit der Suche nach General Zoff und Botschafter Kryxix beginnen. Das Raumschiff besteht aus einer Vielzahl von Räumen und Korridoren, worin sich Waffen oder Schlüssel befinden, mit denen sich Türen öffnen lassen. In anderen halten sich gegnerische Wachen auf, die erst überwältigt werden müssen, bevor man weiterkann. Hat man den Schlüssel nicht, ruft man Sevrina herbei, die alle Schlösser „knacken" kann.

Was Shadowfire so interessant macht, ist die Cursor-Steuerung. Der Spieler kann mit dem Piktogramm-Wahlsystem schneller auf Situationen reagieren, als es mit einer Befehlseingabe über Tastatur möglich wäre.

Shadowfire: Für C 64 und ZX Spectrum (beide Versionen auf einer Cassette)
Hersteller: Beyond Software, Competition House, Farndon Road, Market Harborough, LE16 9NR
Autoren: Steven Cain, Dave Colcough, Karen Davies, Graham Everett, John Gibson, Fred Gray, John Heap, Ally Noble und Colin Parrott
Joystick: Wahlweise
Format: Cassette



Wir zeigen hier drei der Charaktere, die der Spieler in Shadowfire in verschiedenen Situationen dirigiert. Links sieht man Torik im Bewegungsmodus, wobei die unterschiedlichen Richtungen gezeigt werden, in die er gehen kann. Auf dem mittleren Screen wird Zark Mondor gerade angegriffen. Die für ihn möglichen Bewegungsrichtungen und seine Gegner werden am Bildschirmrand dargestellt. Maul, ein Kampf-Droid, wird mit einem Objekt-Screen gezeigt. Der mittlere Teil am unteren Bildschirmrand zeigt, welche Gegenstände er im Augenblick mit sich führt und welche Objekte er aufnehmen kann.

Fast schon antik

Die Atari-810-Diskettenstation ist schon seit langer Zeit auf dem Markt. Da das Gerät jedoch über umfassende Diskettenbefehle verfügt, erläutern wir hier die wichtigsten Befehle und Routinen.

Die 810 verarbeitet einseitige 5 1/4-Zoll-Disketten mit einfacher Schreibdichte. Die Station ist mit dem Computer über eine spezielle parallele Schnittstelle verbunden. Bis zu vier 810 können „in Reihe" geschaltet werden. Die Schalterstellungen auf der Rückseite geben die Stationsnummer (1–4) an. Obwohl die 810 über einen eigenen Microprozessor verfügt, ist sie kein intelligentes Laufwerk, da ein Teil des Atari-Diskettenbetriebssystems erst ins RAM geladen werden muß, bevor die Station angesprochen werden kann.

Das Atari-DOS ist im Lieferumfang der Diskettenstationen enthalten. Es besteht aus drei miteinander in Beziehung stehenden Dateien: DOS.SYS enthält das Dateiverwaltungssystem FMS (File Managing System) und die im RAM gespeicherten Diskettenbefehle. DUP.SYS ist eine Hilfsdatei mit dem DOS-Menü und einigen DOS-Befehlen. AUTORUN.SYS enthält eine Datei, die auf Befehl automatisch in das RAM geladen wird und das DOS-Menü und im RAM gespeicherte Teile des DOS aufruft.

Um mit eingesteckter BASIC-Cartridge auf das DOS zugreifen zu können, muß vor dem Anschalten des Computers das Laufwerk eingeschaltet und die Masterdiskette eingelegt sein. Der Computer „bootet" (lädt) nun einen Teil von DOS.SYS ins RAM. Für den Aufruf des DOS-Menüs mit fünfzehn Bearbeitungsmöglichkeiten müssen Sie DOS eingeben und „Return" drücken. Ist keine Cartridge eingesetzt, wird das DOS-Menü automatisch von der Boot-Routine aufgerufen.

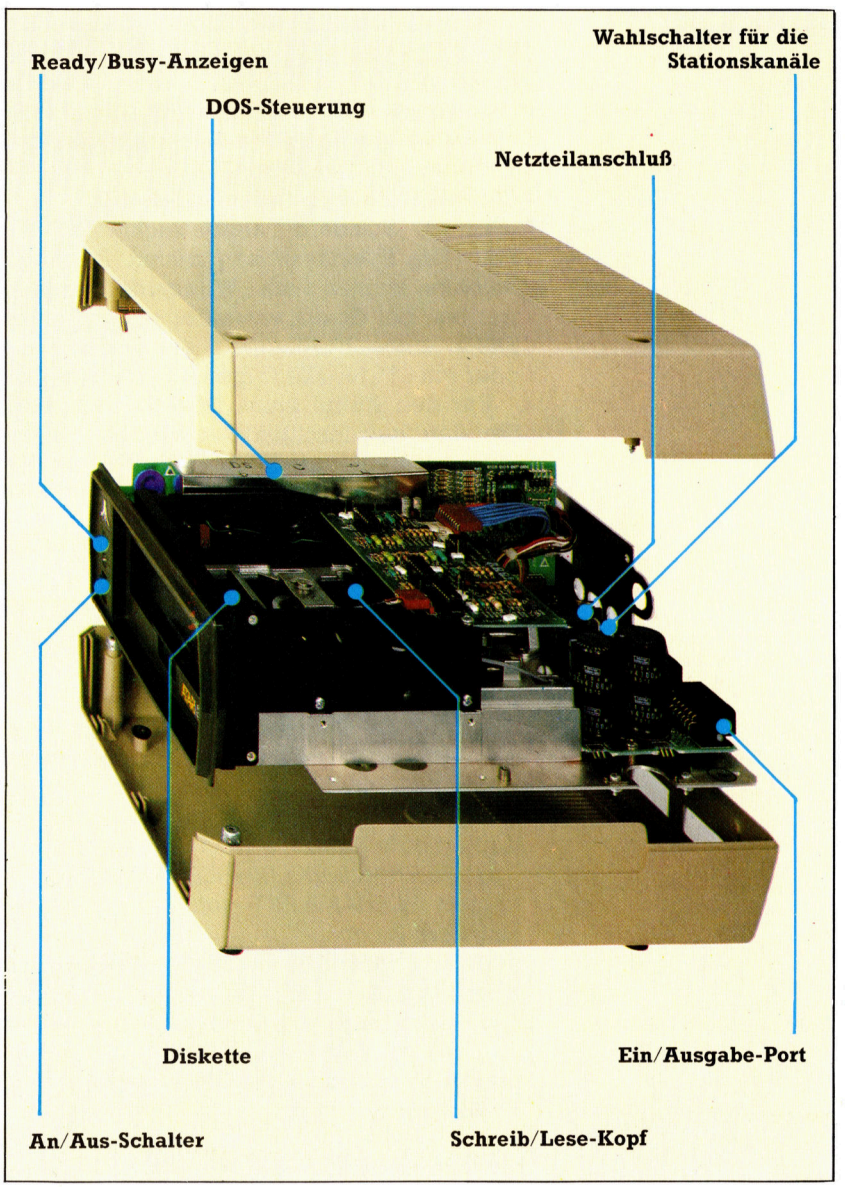
Da jeder Schreibbefehl des DOS automatisch die geschriebenen Daten verifiziert, beträgt die Übertragungsrate nur 2,4 KByte pro Sekunde. Für höhere Geschwindigkeiten kann die automatische Verifizierung mit dem Befehl POKE 1913,80 außer Kraft gesetzt werden. Die Übertragungsrate liegt dann bei 4,8 KByte pro Sekunde. POKE 1913,87 schaltet die Verifizierung wieder ein.

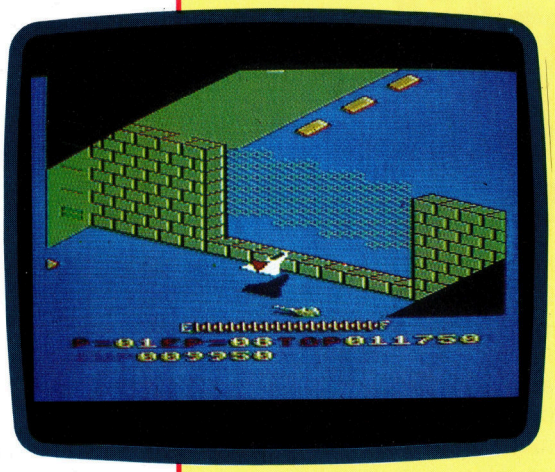
Der Befehl „FORMAT DISK" formatiert die im angegebenen Laufwerk liegende Diskette mit 40 Spuren, von denen jede einzelne in 18 Sektoren zu je 128 Bytes unterteilt ist. Drei Sektoren-Bytes sind für das FMS reserviert, acht Sektoren der Diskette werden vom DOS als Inhaltsverzeichnis gebraucht, und ein Sektor enthält die Belegungstabelle. Insgesamt stehen 707 Sektoren * 125 Bytes oder 88.375 Bytes zur Verfügung.

Mit den Standardbefehlen LOAD, SAVE und entsprechenden BASIC-Kommandos lassen sich Programme und Daten auf Diskette speichern. Dateien können auch byteweise sequentiell oder wahlfrei eingelesen werden. Das Inhaltsverzeichnis und die Belegungstabelle werden beim Schreiben oder Ändern von Dateien automatisch aktualisiert.

Obwohl das Atari-DOS vielseitig und gut durchdacht ist, ist die Diskettenstation 810 heute überholt, und für die neuen Atari-Computer sind leistungsfähigere Diskettenstationen sowie erweiterte Betriebssysteme längst verfügbar.

Größter Nachteil der Atari-810-Diskettenstation ist außer der seriellen Schnittstelle, die den Diskettenzugriff sehr verlangsamt, die geringe Speicherkapazität von 86 KByte pro Diskette. Das gute Betriebssystem gleicht diese Nachteile jedoch etwas aus.





Zaxxon ist eins der erfolgreichen Spiele, die für Atari-Computer angeboten werden. Der Spieler ist Pilot eines Düsenjägers, kämpft mit gegnerischen Flugzeugen und führt Bodenangriffe durch.

Diskettenbefehle

Die Befehle für die Atari-810 haben folgendes Standardformat:

BEFEHL "DN:FILENAME.EXT"

BEFEHL ist der DOS-Befehl, N die Laufwerksnummer (1–4), FILENAME der Dateiname (bis zu acht Zeichen, das erste Zeichen muß ein Buchstabe sein). Die Namenserweiterung .EXT ist nicht notwendig (sie kann den Typ der gespeicherten Daten angeben).

Diese Standard-Dateibezeichnung wird FSP (File Specification) genannt. In einigen Befehlen können auch „Wildcards" verwandt werden, bei denen entweder ein „?" an die Stelle eines bestimmten Buchstabens tritt oder ein „*" für alle folgenden Zeichen steht.

Für den Aufruf eines DOS-Befehls geben Sie den gewünschten Buchstaben ein und drücken die Return-Taste. FNM steht bei den nachfolgenden Erläuterungen für D1:FILENAME.EXT.

A. DISK DIRECTORY – Inhaltsverzeichnis
Stellt eine Liste aller Dateien dar, die die Diskette in Laufwerk 1 enthält, außerdem die Erweiterung und Sektorenanzahl jeder einzelnen Datei.

B. RUN CARTRIDGE – Cartridge aufrufen
Übergibt die Steuerung des Computers an die eingesetzte Cartridge (normalerweise BASIC).

C. COPY FILE – Datei kopieren
Kopiert eine Datei.

D1:FILENAME.EXT,D2:FILENAME.EXT

kopiert FILENAME.EXT von Laufwerk 1 auf Laufwerk 2.

D1:FILENAME.EXT,D1:FILENAMlegt auf der gleichen Diskette die Sicherheitskopie (Bakkup) einer Datei an. Der neue Dateiname muß sich von dem Namen der Ursprungsdatei unterscheiden – in diesem Fall wurde die Erweiterung geändert.

D. DELETE FILE(S) – Datei(en) löschen
FNM löscht die angegebene(n) Datei(en). Zum Löschen aller Dateien kann „*" statt der Angabe aller Dateinamen und Erweiterungen verwandt werden.

E. RENAME FILE – Datei umbenennen
Ändert den Namen der angegebenen Datei. Beispiel:

D1:ALTNAME, NEUNAME

Auch hier lassen sich mit Hilfe der Wildcards die Erweiterungen einer ganzen Dateigruppe ändern.

F. LOCK FILE – Datei sperren
FNM schützt eine Datei (außer bei Neuformatierung) vor Überschreiben oder Löschen. Im Inhaltsverzeichnis erscheint vor gesperrten Dateinamen ein „*".

G. UNLOCK FILE – Datei freigeben
FNM gibt die angegebene Datei oder – bei Wildcards – alle entsprechenden Dateien frei.

H. WRITE DOS FILES – DOS-Dateien auf Diskette schreiben
Folgen Sie den dargestellten Anweisungen, um das DOS auf eine formatierte Diskette zu schreiben.

I. FORMAT DISK – Diskette formatieren
Folgen Sie den Anweisungen, um eine Diskette zu formatieren.

J. DUPLICATE DISK – Diskette kopieren
Folgen Sie den Anweisungen, um eine ganze Diskette entweder von einem Laufwerk auf ein anderes oder – über eine Zwischenspeicherung im RAM – auf eine andere Diskette im gleichen Laufwerk zu kopieren.

K. BINARY SAVE – Binär speichern
FNM,SSSS,EEEE speichert den Inhalt des angegebenen Speicherbereiches (normalerweise ein Maschinencodeprogramm). SSSS ist Anfangsadresse und EEEE Endadresse im vierstelligen Hexadezimalformat.

L. BINARY LOAD – Binär laden
FNM lädt eine Datei, die mit BINARY SAVE gespeichert wurde, wieder in die ursprünglichen Speicherstellen zurück.

M. RUN AT ADDRESS – Von Adresse an ausführen
Geben Sie nach Erscheinen des Prompts die Hexadezimaladresse eines mit BINARY LOAD geladenen Programms an. (RETURN) führt das Programm aus.

N. CREATE MEM.SAV – Datei MEM.SAV anlegen
Folgen Sie für den Aufbau der Datei MEM.SAV den dargestellten Anweisungen. Beim Aufruf des DOS-Menüs speichert das DOS automatisch den Inhalt des Speicherbereiches, der von dem Menü überschrieben wird. MEM.SAV wird bei Aufruf von RUN CARTRIDGE zurückgeladen.

O. DUPLICATE FILE – Datei kopieren
Folgen Sie den Anweisungen, um eine Datei von einer Diskette auf eine andere zu kopieren. Wildcards sind möglich.
Außerdem gibt es noch folgende Befehle zur Steuerung von Programm- und Datendateien:
SAVE LOAD LIST ENTER RUN OPEN# CLOSE# PRINT# INPUT# NOTE# POINT# PUT#GET# STATUS# XIO

Programmdateien:

SAVE FSP – FSP speichern
Schreibt das angegebene Programm im „Tokenformat" auf Diskette.

LOAD FSP – FSP laden
Liest das angegebene Programm im Tokenformat von der Speicheruntergrenze an aufwärts in den Arbeitsspeicher.

LIST FSP,LN1,LN2 – Programm im ATASCII-Format speichern
Speichert ein BASIC-Programm in ATASCII (Ataris Version von ASCII). Sind keine Zeilennummern (LN1 und LN2) angegeben, wird das gesamte Programm gespeichert. LN1 und LN2 bezeichnen Anfang und Ende des Programmteils, der gespeichert werden soll. Zusammen mit ENTER kann der Befehl Programme verbinden.

ENTER FSP – FSP einlesen
Liest eine Datei, die zuvor mit LIST gespeichert wurde, in den Arbeitsspeicher ein und verbindet sie mit dem dort vorhandenen Programm. Bei doppelten Zeilennummern überschreiben die neuen Zeilen die vorhandenen.

RUN FSP – FSP laden und starten
Lädt das angegebene Programm im Tokenformat in den Arbeitsspeicher und startet es.

Datendateien:

OPEN# – Datei öffnen
Steuert den Zugang zu speziellen Kommunikationskanälen (I/O Control Blocks – IOCB genannt) und verbindet sie mit dem entsprechenden Gerät – hier eine Datei auf einer Diskettenstation:

```
OPEN#IOCB,AC1,AC2,FSP
```

IOCB bezeichnet den Ein- und Ausgabekanal (1–5); AC1 den Hilfscode 1 (gibt den Typ des E/A-Vorgangs gemäß einer Tabelle des DOS-Handbuchs an); AC2 bezeichnet das Gerät (0 für Diskettenstation).
Die folgenden Befehle beziehen sich auf IOCBs, die wie beschrieben mit OPEN eröffnet wurden.

CLOSE#IOCB – Datei schließen
Schaltet die für einen bestimmten IOCB gesetzten E/A-Bedingungen ab. Auf geschlossene Dateien ist kein Zugriff möglich.

PRINT# – in Datei schreiben
Schreibt numerische (X,Y) oder Stringdaten (A$) auf den angegebenen IOCB. Beispiel:

```
PRINT#,X,Y oder PRINT#,IOCB,A$
```

INPUT# – von Datei einlesen. Liest von dem angegebenen IOCB numerische oder Stringdaten. Beispiel:

```
INPUT#IOCB,X,Y oder INPUT#,IOCB,A$
```

NOTE# – Angabe der Speicherinformation
Wird vor dem Speichern mit PRINT# gesetzt. Gibt an, in welchem Sektor und auf welcher Bytenummer das nächste Byte auf der Diskette abgelegt werden soll. Die Ergebnisliste kann als Tabelle in einer weiteren Datei abgelegt werden und so als Index für wahlfreien Zugriff dienen. Mit POINT# lassen sich einzelne Bytes lesen. Beispiel:

```
NOTE#IOCB,A,B
```

A ist Nummer des Sektors (1–719) und B die Bytenummer (0–124).

POINT# – Byte laden
Lädt ein Datenbyte in den Arbeitsspeicher, das zuvor mit NOTE# angesprochen wurde. Beispiel:

```
POINT#IOCB,A,B
```

PUT# – Byte schreiben
Schreibt ein einzelnes Byte in den angegebenen IOCB. Beispiel:

```
PUT#IOCB,N
```

N = 1 bis 255.

GET# – Byte lesen
Liest ein einzelnes Byte, das mit PUT# gespeichert wurde. Beispiel:

```
GET#IOCB,N
```

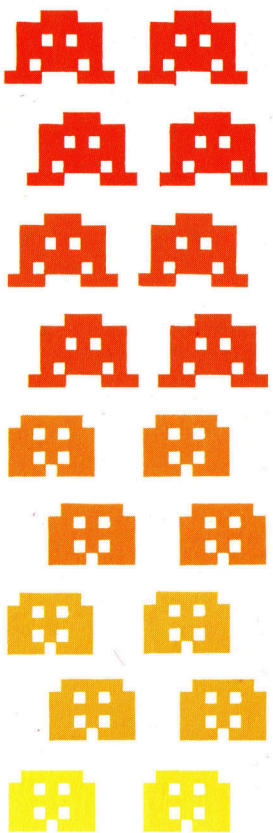
STATUS#IOCB,ERROR – Fehlernummer abfragen
Übergibt der angegebenen Variablen (hier: ERROR) die aktuelle Fehlernummer für den letzten E/A-Vorgang mit dem IOCB. Die Fehlernummer kann dann in der Fehlertabelle des Atari-DOS-Handbuchs nachgeschlagen werden.

XIO CN,#IOCB,AC1,AC2,FSB
Ermöglicht den Abruf von Diskettenbefehlen über Befehlsnummern (CN) statt über das DOS-Menü. Das Atari-DOS-Handbuch enthält eine Liste mit Befehlsnummern und den zugehörigen Befehlen.

In Blue Max „sitzt" der Spieler am Steuer eines Doppeldeckers aus dem Ersten Weltkrieg. Obwohl das Spiel nicht mit einem Flugsimulator zu vergleichen ist, bezieht es seinen Reiz aus der qualitativ hohen grafischen Gestaltung. Blue Max gibt es auf Diskette, Cassette oder als Cartridge.

Immer mit der Ruhe!

Der Hauptvorteil des Maschinencodes ist seine hohe Ausführgeschwindigkeit. Da der Code für manche Aufgaben jedoch zu schnell ist, werden Zeitverzögerungen eingebaut. Wir untersuchen einige Methoden zur Steuerung des Z80 und des 6502.



In der Assemblersprache des 6502 gibt es mehrere Möglichkeiten, Verzögerungsschleifen zu programmieren. Am einfachsten ist das Laden eines Indexregisters, das dann von einer Schleife so lange dekrementiert wird, bis es Null erreicht hat:

V-Schleife	Zeitbedarf für jeden Ablauf
LDY #$07	Zwei Zyklen
DEY	Zwei Zyklen
BNE LOOP	Zwei Zyklen (bei Verzweigung auf die gleiche Seite 3 Zyklen, zu einer anderen 4 Zyklen)

Jede Maschinencodeanweisung benötigt eine bestimmte Anzahl Taktzyklen für die Ausführung. Informationen darüber sind normalerweise bei der Erläuterung der einzelnen Befehle zu finden. So braucht der Befehl DEY zwei Zyklen und LDY im unmittelbaren Adressierungsmodus ebenfalls zwei. Da jeder Zyklus eine Microsekunde dauert, können wir die „Echtzeit" berechnen, die die Verzögerungsschleife benötigt. Die Gesamtzahl der Zyklen errechnet sich folgendermaßen:

1. Der Befehl LDY #$07 braucht zwei Zyklen.
2. Das Programm durchläuft die Schleife siebenmal. Da der BNE-Befehl bei jeder Rückverzweigung drei Zyklen benötigt, dauern DEY und BNE (2+3)*7=35 Zyklen.
3. Der letzte BNE-Befehl verzweigt nicht zurück und braucht daher nur zwei Zyklen.

Die Gesamtzahl der Zyklen beträgt 2+35−1=36. Die Ausführung der Verzögerungsschleife dauert daher 36 Microsekunden.

Beim Einsatz von Maschinencodeschleifen für „Echtzeit"-Verzögerungen (Verzögerungen, die sich in Sekunden oder Microsekunden messen lassen) entstehen jedoch mehrere Probleme. Während der Ausführung eines Maschinencodeprogramms unterbricht der Prozessor seine Aktivität in regelmäßigen Abständen, um andere Teile des Systems zu „bedienen". So überprüft er die Tastatur auf Eingaben, aktualisiert die interne Uhr etc. Diese Unterbrechungen heißen „Interrupts". Auf dem 6502 treten zwei Arten von Interrupts auf: NMI (nicht maskierbare Interrupts) und IRQ (Interrupt-Anfragen). Der erste Interrupt-Typ kann nicht gestoppt werden. IRQ-Interrupts sind jedoch für das Funktionieren des Prozessors nicht unbedingt notwendig und lassen sich sperren.

IRQ-Interrupts werden „maskiert", indem ein bestimmtes Bit des Prozessor-Status-Registers mit dem Befehl SEI auf Eins gesetzt wird. Der CLI-Befehl setzt das gleiche Bit zurück und macht IRQ-Interrupts wieder möglich. Wenn wir nun die IRQ-Interrupts vor Aufruf der Verzögerungsschleife maskieren, können wir die Ablaufzeit präzisieren. Treten während der Schleifenausführung jedoch nicht maskierbare Interrupts auf, läßt sich die Ablaufzeit nicht mehr exakt definieren. Zur Maskierung der Interrupts sollte unsere Schleife auf folgende Art und Weise verändert werden:

Befehl	Funktion	Zeitbedarf
SEI	IRQ ausschalten	Zwei Zyklen
LDY #$07		36 Zyklen
DEY		
BNE LOOP	IRQ einschalten	
CLI		Zwei Zyklen

Die Maskierung der IRQs verlängert die Routine um vier Zyklen, so daß die gesamte Verzögerung nun 40 Microsekunden beträgt – vorausgesetzt, daß keine nicht maskierbaren Interputs auftreten.

Ein weiteres Problem von Verzögerungsschleifen ist ihre „Auflösung" – das heißt, das Intervall zwischen zwei Zählerwerten. In unserem Beispiel haben wir das Y-Register mit dem Wert Sieben geladen. Bei dem Wert Sechs beträgt die Verzögerungszeit jedoch 35 Microsekunden (2+2+(2+3)*6−1+2), bei Fünf 30 Microsekunden und so weiter bis zur minimalen Auflösung von fünf Microsekunden.

Das Programm läßt sich nun noch durch NOP-Befehle außerhalb der Schleife „verfeinern". Ein NOP-Befehl dauert zwei Zyklen. Dabei führt der Prozessor keine Operation aus (No OPeration). Für eine Verzögerung von 44 Microsekunden fügen wir daher vor (oder nach) der Schleife einfach zwei NOP-Befehle ein:

SEI	Zwei Zyklen
LDY #$07	Zwei Zyklen
NOP	Zwei Zyklen
NOP	Zwei Zyklen
DEY BNE LOOP	34 Zyklen
CLI	Zwei Zyklen

Diese Art der Verzögerung ist an eine Obergrenze gebunden, die vom maximalen Wert des Y abhängt. Da das Y-Register acht Bits enthält, liegt die Obergrenze bei 255 und die größtmögliche Verzögerung bei 1280 Microsekunden (2+2+(2+3)*255—1+2) oder etwa einer Tausendstelsekunde. Die Verzögerung kann zwar durch weitere NOP-Befehle im Inneren der Schleife noch etwas verlängert werden — für längere Zeiten müssen wir jedoch andere Methoden verwenden.

Lange Verzögerungen werden oft durch verschachtelte Schleifen oder durch Dekrementierung einer größeren Zahl programmiert. Bei jeder dieser Methoden sollte die Standardauflösung berechnet werden.

```
DELAY SEI
      LDX #$04    ;X—Reg = Zaehler Aussenschleife
LOOP1 LDY #$FF ;Y—Reg = Zaehler Innenschleife
LOOP2 DEY
      BNE LOOP2   ;Ende Innenschleife
      DEX
      BNE LOOP1   ;Ende Aussenschleife
      CLI
```

Die innere Schleife braucht 1276 Microsekunden (2+(2+3)*255—1). Die äußere Schleife steuert die innere und führt DEX und BNE viermal aus. Die Gesamtzeit beträgt: 2+2+(1 276+2+3)*4—1+2=5129 Microsekunden.

Die Befehle des Z80-Maschinencodes haben unterschiedliche Ausführzeiten, die in „T-Zuständen" gemessen werden. Da außerdem der Z80 auf jeder Maschine mit anderer Geschwindigkeit läuft, muß für die Berechnung der Echtzeitdauer jedes Befehls die Anzahl der T-Zustände durch die Taktfrequenz des Micros geteilt werden. So braucht beispielsweise ein Befehl mit einer Dauer von vier T-Zuständen bei einer Taktfrequenz von zwei Megahertz insgesamt zwei Microsekunden.

Für sehr kleine Verzögerungen läßt sich der NOP-Befehl einsetzen, der auf einem 2-MHz-Micro eine Verzögerung von zwei Microsekunden erreicht. Mehrere NOPs nacheinander verlängern diese Zeit zwar, größere Verzögerungen müssen jedoch über Dummy-Routinen programmiert werden. Die folgenden Befehle erzeugen eine Verzögerung von 27 T-Zuständen:

```
CALL DELAY
RET
```

In diesem Beispiel braucht der CALL-Befehl 17 T-Zustände, und RET benötigt zehn. Bei einer Prozessorgeschwindigkeit von zwei Megahertz beträgt die Verzögerung daher 13,5 Microsekunden. Mit NOP-Befehlen am Anfang der Routine kann diese Zeit aber noch geringfügig verlängert werden.

Noch längere Verzögerungen lassen sich nur über Schleifen programmieren. Das folgende Beispiel lädt ein Register, das über eine Schleife dekrementiert wird. Diese Routine erreicht eine Verzögerung von 99 T-Zuständen (oder 49,5 Microsekunden bei 2 MHz):

BEFEHL	ZEITBEDARF
CALL DELAY	17
(DELAY LOOP)	
LD B,5	7
DEC B	4
JR NZ,LOOP	12 oder 7 wenn „true"

Die Verzögerungsschleife besteht aus drei Befehlen, die mit LD B,5 anfangen. Wie bei dem 6502 verändert sich die Ausführzeit dieser Routine durch den Wert, mit dem das Register geladen wird. Die Taktzyklen für die Ausführung dieses Codes betragen:

C = 24 + (N * 16) — 5

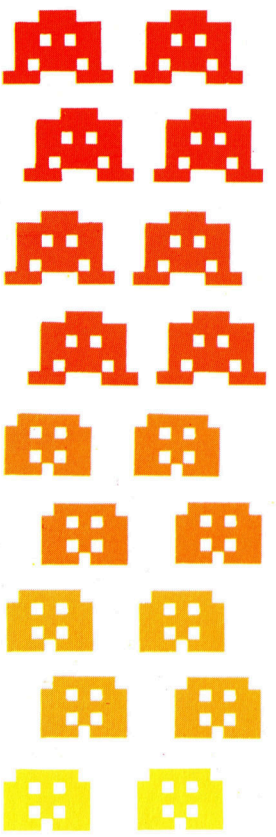
wobei N den Wert darstellt, der in das B-Register geladen wird.

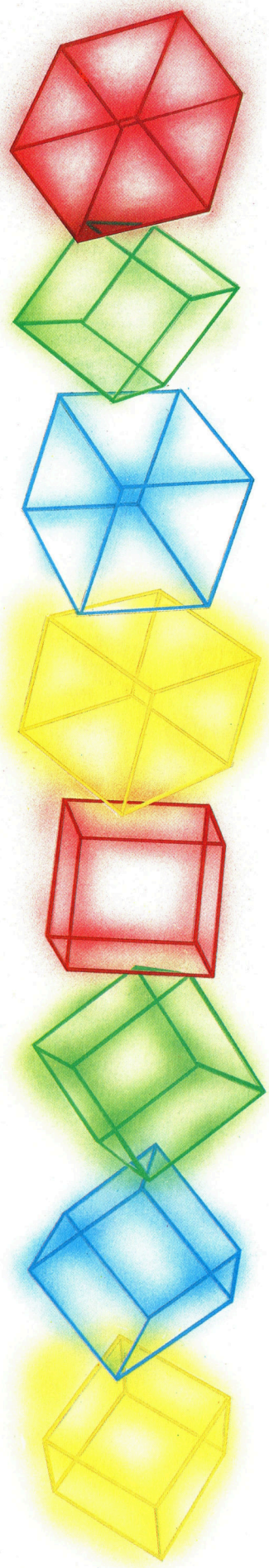
Auch hier können verschachtelte Schleifenzähler eingesetzt werden. Dabei muß jedoch folgendes berücksichtigt werden: Zunächst müssen vor Ausführung einer derartigen Routine alle Register auf den Stapel geschoben werden, damit ihr Inhalt erhalten bleibt. Weiterhin gibt es auf einigen Maschinen Hardware-Interrupts, die den Zeitablauf durcheinanderbringen. Die folgende Routine setzt verschachtelte Schleifen ein (DI und EI schalten die maskierbaren Interrupts aus und wieder ein):

Befehl	Funktion	Zeitbedarf
DI	Interrupt ausschalten	4
PUSH DE	Registerinhalt sichern	11
LD D,nn	Innerer Zähler	7
LD E,nn	Äußerer Zähler	7
CALL OLOOP	Zu äußerem Zähler springen	17
POP DE	Registerinhalt zurückladen	10
EI	Interrupt einschalten	4
(OLOOP)		
DEC E	Außenschleife dekrementieren	4
RET Z	Ende falls Null	11 oder 15
(ILOOP)		
DEC D	Innenschleife dekrementieren	4
JP Z,OLOOP	Springe falls D = Null	10
JP ILOOP	Sonst weiter mit Innenschleife	10

Eine Erhöhung von E verlängert die Verzögerung. Die Routine endet, wenn das E-Register beim Dekrementieren Null enthält. Erreicht der Zähler der inneren Schleife den Wert Null, während der der äußeren Schleife über Eins liegt, wird der innere Schleifenzähler automatisch mit 255 initialisiert. Bevor die Steuerung an die äußere Schleife zurückgegeben wird, muß die innere Schleife bis Null heruntergezählt sein.

In Spielprogrammen sind Verzögerungen des Maschinencodes immer dann notwendig, wenn sich auf dem Bildschirm ein Objekt bewegt, auf das der Spieler reagieren muß. Ein klassisches Beispiel dafür ist das Spiel „Space Invaders". Ohne Zeitverzögerung würden die Bewegungen der eindringenden Aliens zu schnell ablaufen. Durch sorgfältig programmierte Bewegungsverzögerungen wird eine Spielsteuerungüberhaupt erst möglich.





Blickpunkte

Wir setzen unseren BASIC-Kurs mit einem Programm zum Zeichnen und Drehen dreidimensionaler Objekte fort. Die in diesem Programm für Spectrum und Acorn B verwendeten geometrischen Prinzipien zum Darstellen von Punkten bilden eine ideale Basis zur Entwicklung einfacher animierter Grafiken.

Das hier gezeigte Programm verwendet einige geometrische Grundregeln zur perspektivischen Darstellung von Objekten, die man von allen Seiten und aus jeder Entfernung betrachten kann. Alle Daten für die Objekte werden in DATA-Anweisungen gespeichert. Sie enthalten dreidimensionale Koordinaten für jeden Endpunkt einer Linie des Objektes.

Das Problem der Umwandlung der dreidimensionalen Koordinaten in zweidimensionale Werte läßt sich mit einfachen, aber umfangreichen mathematischen Berechnungen lösen. Die X- und Y-Koordinaten jedes Punktes werden durch einen Faktor dividiert, der die Entfernung des Objektes zum Betrachter repräsentiert. Außerdem wird die resultierende Koordinate durch einen dem Koordinatensystem des Computers angepaßten Faktor verkleinert. Durch Veränderung des Entfernungsfaktors kann das Objekt vergrößert oder verkleinert werden.

Eine dritte Konstante kann zur Änderung der Perspektive verwendet werden. Erhöht man diesen Wert, wird die perspektivische Darstellung des Objektes verbreitert.

Zusätzlich gestattet das Programm, das Objekt zu drehen, so daß es aus jedem Winkel betrachtet werden kann. Dies erreicht man durch einfache Trigonometrie. Die Achsen werden um den gewünschten Winkel gedreht, so daß bei der Darstellung des Objektes auf dem Bildschirm der Eindruck einer Drehung entsteht.

Obwohl das Programm relativ viele Effekte ermöglicht, ist es recht einfach aufgebaut. Die Kontrolle des Fluchtpunktes erfolgt über die Zahlentasten 1 bis 8. Daraus ergibt sich: eine Bewegung des Fluchtpunktes nach links, rechts, hoch, hinunter, nach vorne und nach hinten; eine Vergrößerung des perspektivischen Effekts; eine Verkleinerung des perspektivischen Effekts.

Die aktuellen dreidimensionalen Koordinaten werden in den drei Arrays X, Y und Z gespeichert. Die Modifikationen zur perspektivischen Umwandlung dieser Arrays und der Konstanten werden durch verschiedene Unterroutinen ausgeführt. Nach jedem Tastendruck wird das Bild auf dem Schirm gelöscht. Anschließend werden die Änderungen durch Aufruf der entsprechenden Unterroutine ausgeführt, und das Objekt wird neu gezeichnet.

Die perspektivische Umwandlung wird in der Unterroutine zum Zeichnen des Objektes durchgeführt. Dabei wird jeder Satz dreidimensionaler Koordinaten in zweidimensionale Koordinaten umgewandelt und auf dem Bildschirm dargestellt.

Das Erstellen eigener Objekt-Daten für dieses Programm ist eine langwierige Aufgabe. Die Daten werden in Anweisungen am Ende des Programms abgelegt, wobei jeweils vier Werte einen Punkt des Objekts kennzeichnen. Die Gesamtanzahl der Punkte wird in der ersten Zeile des Programms festgelegt. Die im Listing enthaltenen Daten bilden einen Würfel.

X-, Y- und Z-Koordinaten

Die vier Werte eines Punktes setzen sich wie folgt zusammen: Punkt oder Linie, X-Koordinate, Y-Koordinate, Z-Koordinate. Die Werte können einfach ermittelt werden, indem man die Seiten des Objektes mißt. Verwenden Sie einen imaginären Stift und steuern Sie jeden Eckpunkt des Objektes an. Wird der Stift dabei zu einem Punkt geführt, ohne eine Linie zu zeichnen, wird eine 4 als erster Wert notiert. Der Wert 5 gibt an, daß vom vorherigen Punkt eine Linie zu diesem gezeichnet werden muß.

Der Koordinatenursprung (0,0) befindet sich in der Mitte des Bildschirmes. Am besten nehmen Sie diesen Punkt auch als Mittelpunkt für Ihr Objekt. Die X-Achse ist die horizontale Achse mit positiven Werten in aufsteigender Reihenfolge. Die Z-Achse bewirkt den „Tiefen"-Effekt. Der positive Bereich dieser Achse führt „in" den Bildschirm.

Halten Sie die X-, Y-, und Z-Werte so klein wie möglich. Bei der Einstellung des perspektivischen Effekts muß berücksichtigt werden, daß eine Objektbreite von 10 den Bildschirm ausfüllt. Eine Änderung wäre möglich, indem man den Faktor in der Umwandlungsroutine ändert. Am Anfang sollten Sie mit einfachen Objekten etwas experimentieren. Verwenden Sie jedoch nicht zu viele Punkte. Mit etwas Erfahrung können Sie dann auch kompliziertere Objekte umwandeln.

Das hier gezeigte Programm könnte noch erweitert werden, um zusätzliche Effekte zu ermöglichen. So wäre eine Routine zum Bewe-

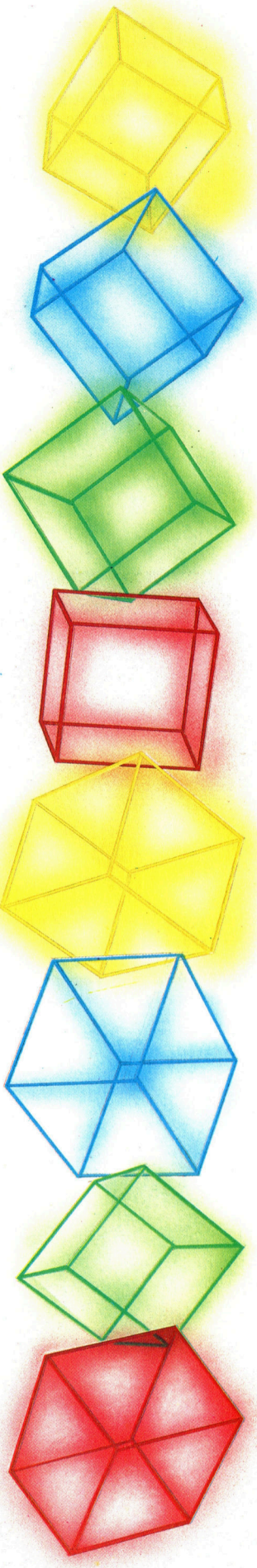
gen des Objektes relativ zum Koordinatenursprung denkbar. Ferner könnten Sie eine Routine entwickeln, die diejenigen Linien des Objektes entfernt, die normalerweise nicht sichtbar sind. Dies ist jedoch ein schwieriges Unterfangen, das komplizierte mathematische Berechnungen erfordert und den Programmlauf sehr verlangsamen würde. Doch selbst wenn Sie keinerlei Änderungen am Programm vornehmen, sondern es in der hier gezeigten Form verwenden, werden Sie sehr interessante Ergebnisse erzielen.

Spectrum-Version

```
 10 LET N=16
 20 DIM P(50)
 21 DIM X(50)
 22 DIM Y(50)
 23 DIM Z(50)
 24 DIM A(50)
 25 DIM B(50)
 40 LET D=10: LET P=0.5
 50 LET SI=SIN 0.09: LET CO=COS 0.09
 60 FOR I=1 TO N
 70 READ P(I),X(I),Y(I),Z(I)
 80 NEXT I
 90:
200 INVERSE 0: GO SUB 300
210 IF INKEY$<>"" THEN  GO TO 210
211 IF INKEY$="" THEN  GO TO 211
212 LET I$=INKEY$
230 INVERSE 1: GO SUB 300
240 IF I$="1" THEN  GO SUB 1000
241 IF I$="2" THEN  GO SUB 2000
242 IF I$="3" THEN  GO SUB 3000
243 IF I$="4" THEN  GO SUB 4000
244 IF I$="5" THEN  GO SUB 5000
245 IF I$="6" THEN  GO SUB 6000
246 IF I$="7" THEN  GO SUB 7000
247 IF I$="8" THEN  GO SUB 8000
250 GO TO 200
260:
300 FOR I=1 TO N
310 LET A(I)=X(I)*300/(P*Z(I)+D): LET B(I)=Y(
I)*300/(P*Z(I)+D)
320 NEXT I
330 FOR I=1 TO N
340 IF P(I)=4 THEN  PLOT A(I)+128,B(I)+85
345 IF P(I)=5 THEN  DRAW A(I)-A(I-1),B(I)-B(I
-1)
350 NEXT I
360 RETURN
370:
1000 FOR I=1 TO N
1010 LET X=X(I)*CO-Z(I)*SI
1020 LET Z=Z(I)*CO+X(I)*SI
1030 LET X(I)=X: LET Z(I)=Z
1040 NEXT I
1050 RETURN
1060:
2000 FOR I=1 TO N
2010 LET X=X(I)*CO+Z(I)*SI
2020 LET Z=Z(I)*CO-X(I)*SI
2030 LET X(I)=X: LET Z(I)=Z
2040 NEXT I
2050 RETURN
2060:
3000 FOR I=1 TO N
3010 LET Y=Y(I)*CO+Z(I)*SI
3020 LET Z=Z(I)*CO-Y(I)*SI
3030 LET Y(I)=Y: LET Z(I)=Z
3040 NEXT I
3050 RETURN
3060:
4000 FOR I=1 TO N
4010 LET Y=Y(I)*CO-Z(I)*SI
4020 LET Z=Z(I)*CO+Y(I)*SI
4030 LET Y(I)=Y: LET Z(I)=Z
4040 NEXT I
4050 RETURN
4060:
5000 LET D=D*0.9
5010 RETURN
5020:
6000 LET D=D/0.9
6010 RETURN
6020:
7000 LET P=P/0.9
7010 RETURN
7020:
8000 LET P=P*0.9
8010 RETURN
8020:
9000 DATA 4,1,1,1, 5,1,1,-1, 5,-1,1,-1, 5,-1,1
,1,5,1,1,1
9010 DATA 5,1,-1,1, 5,1,-1,-1, 5,-1,-1,-1, 5,-
1,-1,1, 5,1,-1,1
9020 DATA 4,1,-1,-1, 5,1,1,-1, 5,-1,-1,-1, 5,-
1,1,-1
9030 DATA 4,-1,1,1, 5,-1,-1,1
```

Acorn-Version

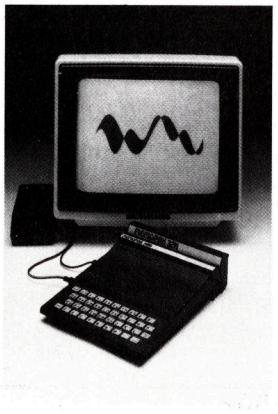
```
 10 N=16
 20 DIM P(50),X(50),Y(50),Z(50),A(50),B(50)
 30 MODE0:VDU29,640;512;5
 40 D=10:P=0.5
 50 SI=SIN(0.09):CO=COS(0.09)
 60 FOR I=1 TO N
 70   READ P(I),X(I),Y(I),Z(I)
 80   NEXT I
 90 :
200 GCOL0,3:GOSUB 300
210 I$=GET$
220 V=VAL(I$)
230 GCOL0,0:GOSUB 300
240 ON V GOSUB 1000,2000,3000,4000,5000,6000
,7000,8000 ELSE 150
250 GOTO 200
260 :
300 FOR I=1TON
310   A(I)=X(I)*1000/(P*Z(I)+D):B(I)=Y(I)*
1000/(P*Z(I)+D)
320   NEXT I
330 FOR I=1 TO N
340   PLOT P(I),A(I),B(I)
350   NEXT I
360 RETURN
370 :
1000 FOR I=1 TO N
1010   X=X(I)*CO-Z(I)*SI
1020   Z=Z(I)*CO+X(I)*SI
1030   X(I)=X:Z(I)=Z
1040   NEXT I
1050 RETURN
1060 :
2000 FOR I=1 TO N
2010   X=X(I)*CO+Z(I)*SI
2020   Z=Z(I)*CO-X(I)*SI
2030   X(I)=X:Z(I)=Z
2040   NEXT I
2050 RETURN
2060 :
3000 FOR I=1 TO N
3010   Y=Y(I)*CO+Z(I)*SI
3020   Z=Z(I)*CO-Y(I)*SI
3030   Y(I)=Y:Z(I)=Z
3040   NEXT I
3050 RETURN
3060 :
4000 FOR I=1 TO N
4010   Y=Y(I)*CO-Z(I)*SI
4020   Z=Z(I)*CO+Y(I)*SI
4030   Y(I)=Y:Z(I)=Z
4040   NEXT I
4050 RETURN
4060 :
5000 D=D*.9
5010 RETURN
5020 :
6000 D=D/.9
6010 RETURN
6020 :
7000 P=P/.9
7010 RETURN
7020 :
8000 P=P*.9
8010 RETURN
8020 :
10000 DATA 4,1,1,1, 5,1,1,-1, 5,-1,1,-1, 5,-1,
1,1, 5,1,1,1
10010 DATA 5,1,-1,1, 5,1,-1,-1, 5,-1,-1,-1, 5,
-1,-1,1, 5,1,-1,1
10020 DATA 4,1,-1,-1, 5,1,1,-1, 5,-1,-1,-1, 5,
-1,1,-1
10030 DATA 4,-1,1,1, 5,-1,-1,1
```



Nach dem Erfolg seiner ZX81-Produkte wandte sich Memotech neuen Produkten zu. Eines dieser Reihe ist der 80-Zeichen-Matrix-Drucker, der DMX 80. Neuestes Produkt des Hauses ist der MTX 512. Er kann Daten auf Cassette oder mit der hier gezeigten Diskettenstation speichern.

Die Technokraten

Memotech ist ein englisches Unternehmen, das sich als Hersteller von Erweiterungen und Peripherien für den Sinclair ZX81 einen Namen gemacht hat. Vor einiger Zeit begann das Haus mit der Produktion eigener Microcomputer – der MTX-Serie.



Memotech hatte sich anfangs auf Produkte für den Sinclair ZX81 spezialisiert, so etwa das hier gezeigte Memopak.

Die Firmengründung von Memotech kam als Ergebnis des ungeheuren Interesses für Sinclair Researchs erste Microcomputer zustande. Trotz der Popularität des ZX80 und ZX81 wurde der Nachteil des kleinen Speichers bald deutlich. Daraus wuchs ein gewaltiger Bedarf an Speichererweiterungen.

Beide Gründer des Unternehmens waren als Lehrer an der Oxford-Universität tätig: Geoff Boyd lehrte Metallurgie am Wilson College und Richard Branton unterrichtete Mathematik am Christ Church College. Die beiden begegneten sich erstmals 1981 während einer Computerausstellung an der Universität und beschlossen, gemeinsam an Speichererweiterungen für den ZX81 zu arbeiten. Erstes Produkt war die 16-K-Erweiterungskarte. Darauf folgte später eine ganze Serie sogenannter „Memopacks", von 32-K- und 64-K-Speichererweiterungen über das hochauflösende Grafikpack (HRG), ein Spreadsheet (Memocalc) sowie eine Textverarbeitung (Memotext) bis hin zu Centronics- und RS232-Schnittstellen und einer Tastatur.

Als Sinclair Research 1982 den Spectrum auf den Markt brachte, entschied sich Memotech dafür, keine Erweiterungen für den neuen Rechner herzustellen. Stattdessen besann man sich auf die gemachten Erfahrungen und konzentrierte sich ganz auf Entwicklung und Bau eines eigenen Rechners. Tim Spencer, Verkaufs- und Marketing-Chef von Memotech erläutert dazu: „Wir glaubten, daß es den ZX81 nicht mehr lange geben würde, also entschieden wir, unseren eigenen Computer zu bauen. Die dazu erforderliche Technologie besaßen wir ja. Doch der ZX81 ist länger im Markt geblieben, als wir vermuteten, und unsere Speichererweiterungen verkaufen sich noch immer gut."

Memotech schätzt, daß weltweit über 250 000 Memopack-Einheiten verkauft wurden. Sie werden ebenso wie die MTX-Computer in der Unternehmenszentrale in Witney hergestellt.

Offiziell wurde die MTX-Serie im Februar 1984 eingeführt. Seitdem sind über 25 000 Rechner des Typs verkauft worden. Der MTX wird in zwei Versionen angeboten: MTX 500 mit 32 KByte und MTX 512 mit 64 KByte. Die Rechner basieren auf dem Z80-A-Microprozessor und haben 16 Farben. Zudem verfügen sie über einen Assembler/Disassembler.

Kooperation mit Fremdfirmen

Der Rechner kann erweitert werden, um so die Möglichkeiten des HRX-Grafik-Pakets von Memotech zu nutzen. In der Grundversion des MTX 500 kann der Benutzer Diskettenstationen sowie drei Grafik-Kontrolleinheiten anschließen: Mit diesem System lassen sich Animationen erzeugen, ist die Komposition von Bildern ebenso möglich wie Grafik-Design und richtiges Setzen von Texten. Das System kostet etwa 18 000 Mark.

Auf die Frage nach der MTX-Philosophie antwortet Tim Spencer: „Wir wenden uns an den ernsthafteren Anwender zu Hause und setzen auf den Büromarkt. Diese Rechner sind nicht für den Spielemarkt konzipiert, wenngleich man natürlich darauf auch die bekannten Spiele laufen lassen kann."

Aufgrund seiner CP/M-Kompatibilität hat der MTX eine große Software-Unterstützung. Das Unternehmen weiß um den Mangel an Software auf Cassetten-Basis, die das System auch für den weniger ernsthaften Anwender interessanter machen würde. Derzeit stehen etwa 40 verschiedene Cassetten für den MTX zur Verfügung. Memotech hofft auf die Entwicklung weiterer Programme. „In den letzten Monaten", so Tim Spencer, „haben wir viel getan. Wir kooperieren mit Continental Software, und PSS schreibt für uns."

Fachwörter von A bis Z

File Maintainance = Dateipflege
In Dateien gespeicherte Informationen müssen bestimmten Vorgängen unterzogen werden, damit ihre Einsatzfähigkeit erhalten bleibt. Dazu gehört das Anlegen von Sicherheitskopien, das Löschen ungültiger oder nicht belegter Datensätze und das Aktualisieren der gespeicherten Informationen. Diese als Dateipflege bezeichneten Abläufe sind für kommerzielle Daten ebenso wichtig wie für die Programmentwicklung oder eine Datenbank mit einem persönlichen Adressenverzeichnis.

File Protection = Dateischutz
Eine Datei auf Cassette oder Diskette ist im allgemeinen jedem zugänglich, der den Dateinamen kennt. Wenn eine Datei vertrauliche Informationen enthält, etwa die Firmenbuchhaltung oder den Quellcode kommerzieller Software, dann sind Vorkehrungen gegen unberechtigte Zugriffe erforderlich. Am einfachsten ist es, die Datenträger an einem Ort aufzubewahren, der für Unbefugte nicht zugänglich ist. Die Wirksamkeit derartiger Maßnahmen ist aber begrenzt, weil das Material ja für die rechtmäßige Nutzung zugänglich bleiben muß und somit versehentlich in falsche Hände geraten könnte. Bei der Erstellung einer vertraulichen Datei sollte deshalb gleich ein eingebauter Dateischutz vorgesehen werden.

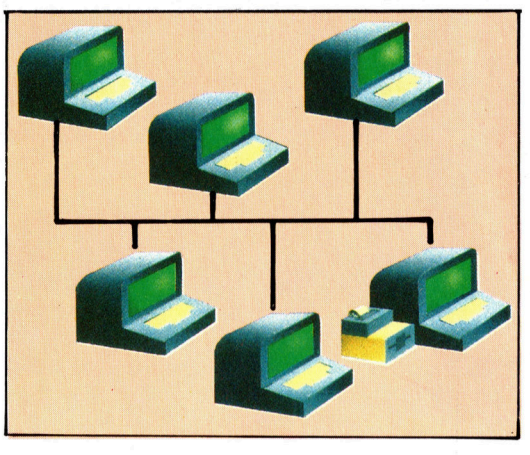
Bei Großanlagen für viele Benutzer fängt die Sicherheit damit an, daß sich jeder mit einer zugeteilten Benutzernummer und einem selbstdefinierten Paßwort „einloggen" muß. Die Benutzernummer bestimmt die „Systempriorität" für den Benutzer, und dazu gehört der Grad der Zugangsberechtigung. Dabei wird beispielsweise jedem Anwender ein bestimmter Directory-Abschnitt zugewiesen, und ein Zugriff auf andere Dateien ist ohne das korrekte Paßwort nicht möglich. Mitarbeiter des Rechenzentrums wie Operateure und Systemprogrammierer haben oft allgemeingültige Benutzernummern – eine Art Generalschlüssel – und damit zu sämtlichen Dateien Zugang.

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.

File Server = File Server
In einem Computer-Netzwerk wird manchmal ein Rechner nur dafür abgestellt, allen anderen Geräten die benötigten Dateien verfügbar zu machen. Zur Erleichterung dieser Aufgabe werden alle Daten von und zu Terminals, Druckern, Diskettenlaufwerken und der sonstigen Peripherie wie unabhängige Dateien behandelt. Der für die Steuerung zuständige Rechner heißt „File Server". Er hat lediglich die Aufgabe, den Datenverkehr zwischen den Netzwerkstationen und den Speichereinheiten zu regeln.

Der File Server wird aktiv, sobald ein Benutzer eine bestimmte Information anfordert – etwa ein Datenfile oder ein Programm auf Diskette oder eine Verbindung mit dem Drucker oder anderweitiger Peripherie. Ist der Kanal für den Benutzer verfügbar, übermittelt der File Server die Daten. Die Datei wird vom Benutzer gegebenenfalls verändert und dann wieder dem File Server überstellt, der sie in der aktualisierten Form zurückspeichert.

File Transfer = Dateiaustausch
Eine Datei wird gewöhnlich im Rechner erstellt und dann in einem peripheren Speicher abgelegt. Die weitere Übertragung auf andere Speicher oder an andere Netzwerkkomponenten, auch an ein entferntes System, stellt einen typischen Fall von Dateiaustausch dar. Schwierigkeiten bei diesem Vorgang bereitet wieder einmal die Kompatibilität. Jedes Gerät hat seine eigenen Spezifikationen hinsichtlich Übertragungsrate, Parität und Steuersignalen. Somit können die Erfordernisse sehr unterschiedlich sein, so daß der Dateiaustausch nicht immer problemlos abläuft.



Beim „Econet" werden bis zu 254 Acorn-Rechner zu einem Netzwerk mit Busstruktur verknüpft. Einem der Rechner wird dann die Rolle des File Server zugewiesen.

Es hat verschiedene Anläufe seitens der Hersteller und diverser Organisationen zu einer Standardisierung gegeben, aber die Probleme sind dadurch eher größer geworden. Das Wachstum der nationalen und internationalen Telefondatennetze erzwingt in letzter Zeit aber doch einheitliche Normen. Innerhalb eines Netzwerks besteht die einfachste Lösung zumeist darin, den ganzen Austausch über denselben Steuerrechner, nämlich den File Server, abzuwickeln, der eine Datei von einer Station übernimmt und dann im entsprechenden Format an eine andere Netzwerkstation weiterleitet.

Bildnachweise
1149: Allan Adler
1150: Ian Dobie
1151: Liz Dixon, Associated Press
1153: Steve Cross
1154, 1155, 1156, 1166: Liz Dixon
1157, 1167, 1169: Ian McKinnell
1158: David Higham
1161, 1162, 1163: Chris Stevens
1170: Soft
1174, 1175: Kevin Jones

+ Vorschau +++ Vorschau +

Vorschau +++ Vorschau +

computer kurs

Der Epson PX-8

Tragbare Rechner können entweder Koffergröße haben oder in die Tasche passen. Dazwischen liegen die „Lap-Helds“:

User-Port-System

In unserem Selbstbau-Kurs wurden bisher die einzelnen Komponenten beschrieben. Diesmal finden Sie eine Zusammenfassung.

Textausgabe

In dieser BASIC-Folge beschäftigen wir uns mit einem Hilfsprogramm, mit dem der auszugebene Text formatiert wird.

Per Anhalter

durch die Galaxis fährt man mit dem Spiel „Hitch Hiker's Guide To The Galaxy“ von Intercom.

Künstliche Intelligenz

ist einer der aufregendsten Teilbereiche der gegenwärtigen Computerarbeit. Wir bringen viele Informationen.