Computer verständlich

FALKEN VERLAG


FALKEN
VERLAG

Vergleich der BASIC-Dialekte

Im folgenden werden die unterschiedlichen Kommandos der verschiedenen BASIC-Dialekte von Sinclair ZX81, Sinclair Spectrum, Apple II und Commodore C64 und des BASIC, das wir in diesem Buch benutzt haben, gegenübergestellt.

Unser BASIC	Sinclair ZX81	Sinclair Spectrum	Apple II	Commodore C64
CLS	CLS	CLS	HOME	PRINT CHR (147)
PRINT (ZEILE, SPALTE)	PRINT AT ZEILE, SPALTE	PRINT AT ZEILE, SPALTE	VTAB ZEILE: HTAB SPALTE	POKE 211, ZEILE POKE 214, SPALTE SYS 58640 (für C64) SYS 58643 (für VC 20)
RND	RND	RND	RND (ZAHL)	RND (ZAHL)
LOG	LN	LN	LOG	LOG
ASC	CODE	CODE	ASC	ASC

Computer verständlich

Susan Curran und Ray Curnow

Computerspiele, Grafik & Musik

Übersetzt und bearbeitet von
Wolfgang Rudolph

FALKEN VERLAG

Von den gleichen Autoren erscheinen im Falken-Verlag: »Einführung in die Programmiersprache BASIC« (Bd. 4303) und »Lernen mit dem Computer« (Bd. 4304).

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Curran, Susan:
Computerspiele, Grafik und Musik / von Susan Curran u. Ray Curnow. [Übers.: Wolfgang Rudolph]. – Niedernhausen/Ts.: Falken-Verlag, 1984
(Computer verständlich) (Falken-Sachbuch)
Einheitssacht.: Games, graphics and sounds «dt.»
ISBN 3-8068-4305-8
NE: Curnow, Ray:

ISBN 3 8068 4305 8

Titelbild: Computer Grafic Design, Stuttgart
Illustrationen: Hayward and Martin Ltd., Radius, David Whelan, The Communication Studio
Übersetzung: Wolfgang Rudolph, Köln

Satz: LibroSatz, Kriftel bei Frankfurt
Druck: Offset-Team Zumbrink, Bad Salzuflen

817 2635 4453 5271

Inhalt

Einleitung

Mehr und mehr Menschen wagen den Sprung und kaufen sich einen Heimcomputer. Wozu brauchen sie ihn? In der überwältigenden Mehrzahl gebrauchen sie ihn zum Spielen!

Vielleicht ist das nicht unbedingt die Art und Weise, wie Sie Ihren eigenen Heimcomputer benutzen wollen. Mit dieser Einstellung stehen Sie nicht allein da. Die meisten Leute sagen, wenn sie gefragt werden, warum sie sich einen Heimcomputer kauften, daß sie sich oder ihren Kindern etwas über Computer beibringen wollen. Schätzungsweise 80% aller Käufer geben Ausbildung als Hauptgrund für den Kauf eines Computers an. So kann man also sagen, daß Spiele erzieherisch wirken.

Natürlich machen Spiele auch Spaß. Gleichgültig, welche Spiele Sie am liebsten spielen – schnelle Spiele oder langsame, gedankenanregende Spiele, die sich über Stunden hinziehen, Action- oder Strategiespiele, Wortspiele oder Spiele mit Mathematik –, Sie werden bestimmt einige Spiele finden, die Sie anziehend finden (und wahrscheinlich ein paar, von denen Sie sich kaum noch losreißen können).

Einige Spiele sind speziell zum Lernen konzipiert, um etwa Kindern oder Erwachsenen Geometrie, Aussprache, Fremdsprachen, Geschichte, Geographie und anderes beizubringen. Andere Spiele scheinen auf den ersten Blick nicht so lehrreich zu sein; dennoch können sie Ihnen eine Menge beibringen. Mit »Space-Invaders« beispielsweise trainieren Sie die Koordination Ihrer Bewegungen; Sie üben sich darin, Strategien zu planen (und herauszufinden, ob sie funktionieren!). Das wahrscheinlich Wichtigste ist jedoch, daß Sie sich daran gewöhnen, mit einem Computer umzugehen.

Es ist deshalb keine schlechte Idee, sich zusammen mit dem neuen Computer auch gleich ein paar Spiele zu kaufen. Sie helfen Ihnen, sich an die Tastatur und an die Betriebssystemkommandos zu gewöhnen, und später gewähren sie Ihnen eine wohlverdiente Pause beim Abrechnen des Haushaltsgeldes oder was auch immer Sie machen.

Wir möchten Ihnen mit diesem Buch eine Einführung in die ganze aufregende Welt der Computerunterhaltung geben: Welche Arten von Spielen sind überhaupt erhältlich? Was können Sie von diesen cellophanumhüllten Kassetten erwarten? Welche Spiele bringen den besten Gegenwert für ihren Preis? Wir werden Ihnen all diese Fragen beantworten. Gleichzeitig wollen wir Ihnen zeigen, wie Sie etwas

über Computer lernen können, indem Sie Ihre eigenen Programme schreiben oder Programme aus Zeitschriften und Büchern Ihrem System anpassen. Um Ihnen den Start zu erleichtern, werden wir Ihnen ein paar Programme mitliefern.

Welcher Computer ist erforderlich?

Wenn Sie einen Computer besitzen oder sich bereits entschlossen haben, welchen Computer Sie kaufen wollen, so ist das in Ordnung. Alle Personal Computer können auch für Spiele benutzt werden. Falls Sie sich jedoch noch nicht entschlossen haben, nennen wir Ihnen einige Kriterien, nach denen Sie sich richten sollten. Wir werden Ihnen auch etwas über Peripheriegeräte sagen. Das sind besondere Zusatzgeräte wie etwa Joysticks oder Lichtgriffel, mit denen das Spielen mit Ihrem Computer noch mehr Spaß macht.

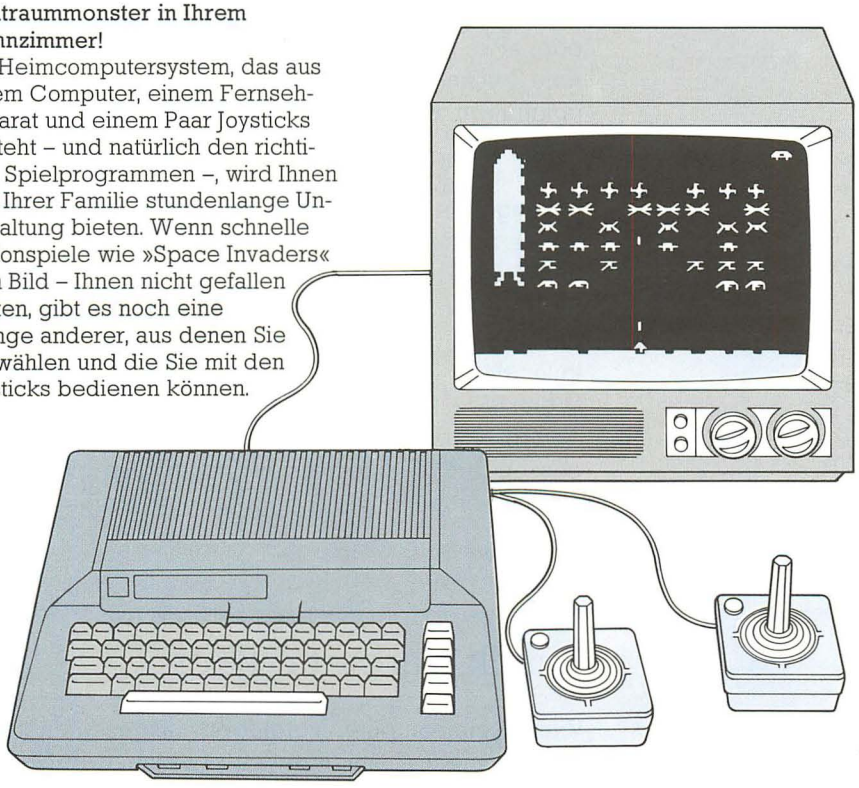
Wir mußten uns für einen bestimmten Computer entscheiden, für den wir unsere Programme geschrieben haben. Dafür gibt es 2 Hauptgründe: Zum einen hat jeder Heimcomputer seine eigenen Spiele, Grafik und Töne, und wir konnten in einem Buch dieses Umfangs einfach nicht beschreiben, wie die Programme für die ganzen verschiedenen Geräte geschrieben werden müßten. Zum anderen befindet sich der Heimcomputermarkt in ständigem Fluß, so daß jede Geräteauswahl nur begrenzten Wert hätte und daher unbefriedigend wäre. Wenn Sie sich in unsere Programmierstrategien vertiefen und sich genau ansehen, wie wir Spiele für eine bestimmte Maschine aufbauen und schreiben, werden Sie nachvollziehen können, was und warum wir etwas gemacht haben.

Wir haben unsere Programme in Microsoft-BASIC geschrieben, der verständlichsten und unkompliziertesten Form dieser Sprache. Verwendet wird diese Sprache auf dem Tandy (Radio Shack) Color Computer, dem Dragon 32, dem Video Genie und vielen anderen bekannten Heimcomputern. Die Programme sind für den Dragon ausgelegt; sie laufen aber mit wenigen oder gar keinen Änderungen auf Geräten wie dem Tandy Color Computer mit Extended BASIC. Für den Fall, daß Sie ein anderes Gerät

haben, bringen wir genügend Erläuterungen, damit die Programme ohne große Probleme an Ihren Computer angepaßt werden können.

Wissen Sie bereits etwas über Computer?

Wahrscheinlich nicht! Eventuell haben Sie auf der Schule oder Hochschule ein wenig gelernt. Sie haben sich vielleicht auch ein paar Computerzeitschriften gekauft (und sie unter Umständen als schwer verständlich empfunden). Oder Sie sind vielleicht wirklich noch ganz am Anfang. Gleichgültig auf welchem Wissensstand Sie sich befinden – Sie sollten in diesem Buch eine Menge an Interessantem und Unterhaltsamem finden. Wenn Sie noch Anfänger sind, finden Sie Informationen über das Auswählen und Be-



Weltraummonster in Ihrem Wohnzimmer!
Ein Heimcomputersystem, das aus einem Computer, einem Fernsehapparat und einem Paar Joysticks besteht – und natürlich den richtigen Spielprogrammen –, wird Ihnen und Ihrer Familie stundenlange Unterhaltung bieten. Wenn schnelle Actionspiele wie »Space Invaders« – im Bild – Ihnen nicht gefallen sollten, gibt es noch eine Menge anderer, aus denen Sie auswählen und die Sie mit den Joysticks bedienen können.

nutzen von fertigen Spielen, die sich auf Compact-Cassetten oder in ROM-Cassetten befinden oder die Sie aus Zeitschriften abtippen müssen. Wenn Sie schon etwas über das Programmieren in BASIC, der populärsten Sprache für Heimcomputer, wissen, werden Sie lernen, wie Sie Ihr Wissen einsetzen können, um Spiel- und Grafikprogramme zu schreiben.
Wir werden Ihnen hier nicht die Grundlagen von BASIC beibringen, denn es gibt zuviel anderes, mit dem wir Sie sonst noch bekannt machen wollen. Die meisten unserer Programme sind jedoch einfach geschrieben, und wenn Sie sich BASIC in Ihrem Computerhandbuch oder einem Einführungsbuch etwas angeschaut haben, sollten Sie keine Probleme haben, unseren Programmlistings und den allgemeinen Vorschlägen für Programmentwicklung und -erstellung zu folgen.

Spiele auf dem Heimcomputer

Wenn Sie es gewohnt sind, Videospiele wie »Asteroiden« oder »Pac-Man« zu spielen, werden Sie feststellen, daß es eine Zeitlang dauert, sich an deren Vettern auf den Heimcomputern zu gewöhnen. Die Grundideen sind die gleichen, aber die Einzelheiten variieren, da einige Heimcomputer nicht so viel Speicherkapazität besitzen wie die Videogeräte. Außerdem sind die meisten nicht nur für Spiele, sondern auch für kleinere Geschäftsanwendungen und Ausbildungsaufgaben konzipiert. Die besten Spiele für Heimcomputer machen jedoch genausoviel Spaß wie die Originalversionen, wenn sie auch nicht ganz so spektakulär sind.
Wenn Sie ein Experte für Schach, Backgammon oder andere Brettspiele sind, werden Sie sicher auch eine für Sie passende Computerversion finden. Das gleiche gilt für Abenteuerspiele wie »Dungeons and Dragons«.

Grafik, Farbe und Ton

Viele Heimcomputer vermögen ausgezeichnete Bilder, Diagramme und Liniengrafiken zu erzeugen. Die meisten können das auch in Farbe, wenn sie mit einem Farbfernseher verbunden sind. Wir werden Grafik nicht nur des-

halb besprechen, weil viele Spiele davon Gebrauch machen, sondern weil sie für sich allein schon interessant ist. Einige der aufgeführten Programme sind keine Spielprogramme, sondern Grafikprogramme, die Ihnen die Art von Bildern veranschaulichen sollen, die Sie mit einfachen BASIC-Kommandos erreichen können.
Auch die Tonerzeugung ist wichtig. Die meisten Computer können ein paar einfache Pieptöne erzeugen, um Spiele zu beleben und realistischer zu gestalten. Einige können etwas mehr und erlauben es, erkennbare Melodien zu erzeugen. Wir werden erklären, wie das alles gemacht wird und wie Sie diese Möglichkeiten für sich selbst in einer Vielzahl von Programmen einsetzen können.
Jetzt aber genug der Einleitung! Lassen Sie uns nun beginnen und sehen, was Computer wirklich für Sie tun können!

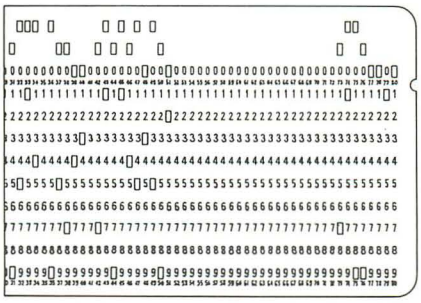
Die Entwicklung der Computerspiele

Wie haben die Computerspiele angefangen? »Computerspezis« würden sagen: mit den allerersten Computern in den 40er Jahren. Für Leute, die mit Computern arbeiten, ist die Beschäftigung mit dem Computer ein Spiel, auch wenn sie einem ernsthaften Zweck dient. Genauso betrachten Erfinder die Entwicklung von Computern als Spiel. Seit es große Computer zum Spielen gibt, haben die Programmierer mit ihren Maschinen gespielt und das seit 30 Jahren – oder auch länger. Das ist abhängig von der Definition, was ein Computer ist. Wann auch immer es ein paar freie Sekunden Computerzeit gab – und diese frühen Computer waren so teuer und ausgelastet, daß ein paar Sekunden das äußerste waren, worauf man hoffen konnte –, schoben die Programmierer und Operatoren ein oder zwei »Spaß«-Programme ein, um die Sache ein wenig zu beleben.

Was waren das für »Spaß«-Programme? Die Favoriten waren wahrscheinlich Computerzeichnungen von Snoopy, nackten Damen, der Mona Lisa und andere Illustrationen, die erstellt werden können, indem man die Buchstaben des Alphabets in verschiedenen Kombinationen ausdruckt. Die Programmierer gaben einen Stapel Lochkarten mit Anweisungen für den Computer ein, und heraus kamen die Bilder auf dem Zeilendrucker des Computers. Ein Beispiel dieser frühen »Computerkunst« – ein Portrait von Charles Babbage – finden Sie auf der nächsten Seite.

Dahinter steckt keine besonders ausgetüftelte Programmierung; es ist nur eine Menge schwerer Arbeit. Der Künstler mußte Zeichen für Zeichen bestimmen, was er vom Computer gedruckt haben wollte. Dann gab er der Maschine die nötigen Instruktionen, um mit dieser Arbeit zurechtzukommen.

Zeichnungen waren besonders geeignet, da sie der Geschwindigkeit der damaligen Computergeneration angepaßt waren. Wenn man einen Stapel Lochkarten eingeben mußte (anstatt auf einer Tastatur tippen zu können), und dann warten mußte, bis der Computer sie verarbeitet und die Resultate auf dem Zeilendrucker ausgegeben hatte, konnte man kaum erwarten, damit schnelle Actionspiele wie »Asteroiden« oder »Breakout« zu spielen!



Computerkunst
Oben links: Ein Kupferstich, welcher Charles Babbage (1791–1871) zeigt, einen der Pioniere der EDV. Oben rechts: Die entsprechende Version eines Zeilendruckers. Das Zeilendruckerbild wurde von einem Großrechner unter Benutzung eines Satzes Lochkarten (eine davon ist links zu sehen) zur Programmierung erstellt. Es ist kein wirkliches Computerbild – so etwas gibt es nicht –, sondern eine Menge übereinandergedruckter Buchstaben.

Schach

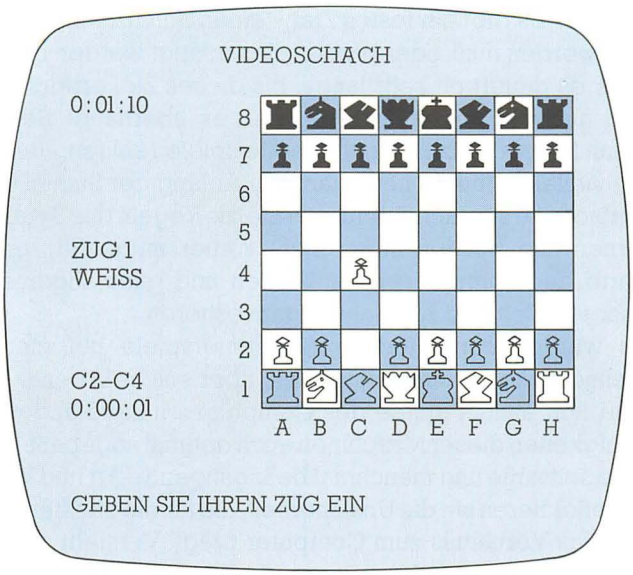
Schachprogramme wurden ebenfalls sehr früh entwickelt, gegen Ende der 40er und zu Beginn der 50er Jahre. Sie erforderten eine Menge Überlegungen und konnten nicht gerade in ein paar freien Momenten erstellt werden. Einen Computer zum Schachspielen zu bringen war eine lange, umständliche Angelegenheit. Die meisten Untersuchungen wurden als regelrechte Forschungsprogramme an Universitäten durchgeführt. Die Entwickler opferten einen großen Teil ihrer Zeit dem Schachspiel, da es ein gutes Beispiel für komplizierte Denkprozesse ist. Indem sie einem Computer Schach beibrachten, lernten sie, wie man dem Computer ganz allgemein intelligentes Verhalten beibringt.

Es dauerte lange, bis Computer überhaupt einigermaßen gut Schach spielen konnten. Das vermutlich erste Computerschachspiel lief im Jahre 1956 auf einem Computer namens Maniac 1 in Los Alamos/New Mexico. Um die Aufgabe zu vereinfachen, benutzte die Maschine ein Feld mit nur 36 anstatt der normalen 64 Quadrate. Der Computer benötigte für jeden seiner Züge 12 Minuten, und seine Spielstärke wurde mit der eines Menschen verglichen, der etwa 20 Partien gespielt hat – ein blutiger Anfänger, in der Tat!

Im folgenden Jahr wurde ein leistungsfähigerer Computer, ein IBM 704, programmiert, das erste »echte« Schachspiel auf einem normalen Brett bis zum Ende durchzuspielen. Das Programm war mehr als zehnmal länger als sein Vorgänger, aber das Resultat war dennoch nicht besser als etwa das, was ein durchschnittlicher Freizeitschachspieler zu leisten vermag.

Auch heute können die besten Spieler selbst die ausgetüfteltsten Schachprogramme schlagen. Nichtsdestotrotz hat sich die Spieltheorie, das heißt die Ausarbeitung von Strategien, um ein Spiel zu planen und zu gewinnen, stetig entwickelt. Heutzutage können für ein paar Mark Schachprogramme für die billigsten Heimcomputer gekauft werden, die wesentlich leistungsfähiger sind als die früheren, teuer entwickelten Versuche. Außerdem sind sie auch noch gut: Die meisten haben verschiedene Spielstärken, so daß sie sowohl für Anfänger als auch für Experten würdige Spielpartner darstellen können.

Andere traditionelle Spiele, wie zum Beispiel Dame, wurden in früheren Untersuchungen ebenfalls benutzt. Da sie nicht so kompliziert sind wie Schach, kann ein gut pro-



Computerschach
Die heutigen Schachprogramme enthalten eine Unmenge Schachtheorie, mit der der Computer seine Züge berechnet, einschließlich zahlreicher zugreicher Spieleröffnungen. In diesem Bild hat der Computer, der Weiß spielt, sich innerhalb einer Sekunde entschlossen, das Spiel durch die Bewegung eines Bauern von C2 nach C4 zu eröffnen. Dies ist die Englische Eröffnung.

grammierter Computer immer den besten Zug herausfinden und wird seinen menschlichen Gegenspieler jedesmal schlagen – außer wenn der Programmierer Mitleid mit dem Spieler hat und den Computer instruiert, gelegentlich Fehler zu machen...

Abenteuerspiele

Der wirkliche Durchbruch kam jedoch in dem Moment, als Programmierer anfingen, neue Spiele speziell für Computer zu erfinden, anstatt althergebrachte Spiele anzupassen. Diese neue Spielegeneration machte von der unbegrenzten Geduld eines Computers Gebrauch und von seiner phänomenalen Fähigkeit, sich Informationen zu merken und auf jede nur denkbare Art zu kombinieren. Diese Spiele wurden Abenteuerspiele oder »Adventure Games« genannt.
In solch einem Abenteuerspiel reist der Spieler durch eine Welt voller Abenteuer, die der Computer erschaffen hat. Man kann dem Computer sagen, wo man hingehen möchte – nach Norden, Süden, Osten oder Westen – und was man tun möchte. Nach jeder Bewegung beschreibt der Computer, wo man sich befindet und was jeweils geschieht. Ein Spieler kann seltsame Wesen treffen, Zaubersprüche lernen, einen Schatz finden oder angegriffen und ausgeraubt

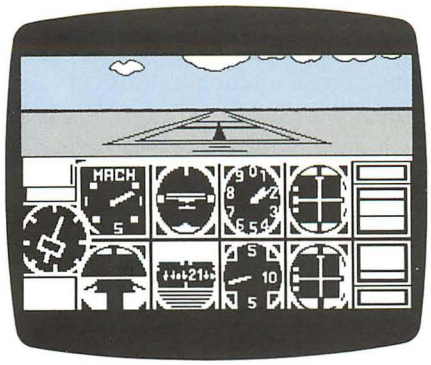
werden. Es gibt ein festes Ziel – einen Schatz, der gefunden werden muß, oder Wissen, das erlangt werden muß – aber es dauert oft sehr lange, bis dieses Ziel erreicht ist und nur wenige Spieler erreichen es überhaupt. Selbst wenn Sie ein ganzes Wochenende spielen sollten, stehen Sie vielleicht immer noch ganz am Anfang der Suche. Um Fortschritte zu machen, müssen Sie die Regeln des Spieles lernen (sie werden Ihnen nicht vorher mitgeteilt), eine Karte der Computerwelt aufstellen und ganz allgemein sehr viel Zeit und Anstrengung investieren.

Es wurden computerlose Abenteuerspiele mit einem menschlichen Führer entwickelt, aber solche Abenteuer sind nun einmal Spiele des Computerzeitalters und den Fähigkeiten dieser Maschinen auch optimal angepaßt. Auf eine seltsame und manchmal beängstigende Art und Weise reflektieren sie die Unsicherheit, die für die meisten von uns das Verhältnis zum Computer prägt. Versteht er uns wirklich, steckt wirklich keine Seele in der Maschine? Wird er unser Leben bestimmen? Auf jeden Fall gibt es viele, die geradezu süchtig nach solchen Spielen sind.

Eines der ersten Abenteuerspiele – es hieß einfach »Adventure« – wurde Ende der 70er Jahre von Computerexperten in den USA entwickelt. Tracy Kidders »The Soul of a New Machine« gibt einen lebhaften Eindruck von den »Mitternachtsprogrammierern«, die auch nach der Arbeit noch blieben, um »Adventure« zu spielen, und wie dieses Spiel als Feuerprobe diente, um zu sehen, ob ein neuer Computer korrekt arbeitete.

Ernsthafte Computeranwendungen sind oft angepaßt worden, um auch der Unterhaltung zu dienen. Gute Beispiele sind beispielsweise Computersimulationen für das Fahren eines Autos, das Fliegen eines Flugzeugs und sogar die Kontrolle eines Raumschiffs. Viele Übungsprogramme enthalten ausgereifte Computersimulationen, die dem Übenden einen Eindruck geben sollen, was alles möglich ist, ohne jedoch dabei das Leben oder teure Ausrüstung zu riskieren. Gleichzeitig sind billigere Simulationen für reine Unterhaltungszwecke erhältlich. Sie können auf dem Mond landen oder ein Rennauto fahren, und das alles in Ihrem Wohnzimmer, wobei Sie dann von der Forschung profitieren, die in die »richtigen« Versionen gesteckt wurde.

CAD/CAM (computerunterstützte Planung und Fertigung) hatte genauso seine Ableger. Die Computersysteme, die heute dazu benutzt werden, komplizierte mechanische Teile oder Gebäude herzustellen, oder sogar, um die Maschinen zu kontrollieren, die sie herstellen, sind eine beträchtliche Investition. Selbstverständlich sind die Zei-



»Flugsimulator«
Das Hauptbild sieht meistens etwa so aus: Oben hat man einen vereinfachten Blick aus dem Cockpit, unten ist das Armaturenbrett. Die verschiedenen Skalen und Anzeigen reagieren auf Ihre Steuerung wie in einem richtigen Flugzeug. Um das Flugzeug zu fliegen, geben Sie Ihre Befehle mit der Tastatur ein (oder mit einem Joystick). Sie können normalerweise durch die Landebahn und andere Einzelheiten gut erkennen, wo Sie sich befinden. Im schlimmsten Falle gibt es eine spektakuläre Bruchlandung auf dem Bildschirm, die Sie genießen können – in absoluter Sicherheit!

chenprogramme, die Sie für Ihren Computer kaufen können, nicht annähernd so hochentwickelt wie die heutigen Spitzensysteme, aber sie sind gut vergleichbar mit den Systemen, die noch vor wenigen Jahren teures Geld gekostet haben. Ihr Heimcomputer hat sowohl die Rechenleistung als auch die Speicherkapazität, die bis vor kurzer Zeit nur die größten Maschinen besaßen.

Videospiele

So wurde vieles von dem, was in den heutigen Computerspielen steckt, auf Großrechnern vorbereitet, lange bevor Heimcomputer existierten. Es gibt jedoch auch noch andere Vorfahren der Computerspiele. Einer davon ist das Videospiel.
Flipper, also Spielgeräte mit mechanischen Federn und elektrischen Sensoren, gibt es bereits seit Jahrzehnten. In den frühen 70er Jahren konnte man das Auftauchen ihrer elektronischen Gegenspieler beobachten. Diese neuen Spiele hatten einen Bildschirm wie ein Fernsehapparat, und der Spieler benutzte einen sogenannten Joystick, um die Bewegung eines Schlägers zu steuern oder einen Lichtblitz abzufeuern.
Der Vater der Videomaschinen war ein Amerikaner namens Nolan Bushnell. Er entwickelte den Prototyp eines Videospiels namens »Computer Space« bereits im Jahre 1971. Verglichen mit den heutigen ausgereiften Spielen war es extrem einfach. Bushnell zog mit 2 anderen klassischen Spielen nach: Videotennis, bei dem 2 Spieler einen Ball über den Schirm hin- und herschlagen, und »Breakout«, in dem eine Mauer Stein für Stein abgeschossen wer-

den muß. Bushnell gründete dann Atari, eine der größten Videospiel-Gesellschaften.
Zuerst fand man diese einfachen Videospiele in Gaststätten und Spielhallen. Später wurden Versionen für den Hausgebrauch entwickelt. Das erste Fernsehspiel wurde von Magnavox 1972 hergestellt. Es war eine trickreiche Angelegenheit mit farbigen Schablonen mit Ausschnitten für die einzelnen Spielfelder. Der Spieler mußte für die einzelnen Spiele die Folien auf den Bildschirm heften. Der Computer produzierte dann einen einzelnen weißen Lichtpunkt, der auf die verschiedenen Ziele »feuern« konnte. Fernsehtennis war einfacher und hatte größeren Erfolg. 1975 produzierte Atari eine neue Version namens »Pong«. Der Pong-Computer konnte lediglich »Pong« spielen, also Ping-Pong, jedoch seine Nachfolger konnten bereits eine Vielzahl von Spielen mit Ball und Schlägern anbieten.
Ende der 70er Jahre waren Fernsehspiele die ganz große Mode. Es waren viele Geräte auf dem Markt – kleine Computer, die allerdings nur diejenigen Spiele spielen konnten, für die sie programmiert waren. Sie waren nicht imstande, Programme auszuführen, die der Benutzer in einer Sprache wie BASIC geschrieben hatte. Wir nennen sie Spezialcomputer, im Gegensatz zu den Vielzweckcomputern wie den heutigen Heimcomputern.
Während die Fernsehspiele begannen, die frühen Videospiele zu imitieren, bewegten sich diese sozusagen zu höheren Sphären. Das tollste von allen war wahrscheinlich »Space Invaders«, das Spiel, das die japanische Firma Taito 1978 auf eine ahnungslose Menschheit losließ.
»Breakout« war nie ein ernsthafter Konkurrent für die Flippergeräte. Der kleine fernsehartige Schirm konnte nicht mit den hellen, leuchtenden Anzeigen konkurrieren. »Space Invaders« war da schon etwas ganz anderes. Der Bildschirm strahlte intensiver und natürlich farbig, und sogar das Gehäuse war attraktiv und zog die Aufmerksamkeit auf sich. Und bald zogen »Space Invaders« und andere derartige Spiele – wie »Defender«, »Galaxian«, »Asteroide« oder »Pac-Man« – um die ganze Welt.
Diese großen Videospiele werden auch durch Computer mit Festprogrammen kontrolliert, die jedoch wesentlich leistungsfähiger als jene in den ersten Fernsehspielen sind. Erst heute, einige Jahre nach dem Erscheinen von »Space Invaders«, gibt es die ersten Heimcomputerspiele, die sich in der Qualität mit diesen frühen Videospielen messen können. In der Zwischenzeit haben sich die Videospiele natürlich zu ganz neuen Höhen von Komplexität und Vielfalt aufgeschwungen, die die Heimcomputer nicht erreichen können – noch nicht.

Tragbare Computerspiele
Wie die Videospiele sind die meisten tragbaren Spiele »festgelegte« Computer: Das heißt, sie sind dazu da, ein Spiel zu spielen, und zu sonst nichts. Ein Mikrocomputer kann sehr viel mehr – und außerdem noch viele verschiedene Spiele.

Während »Space Invaders« die Spielhallen der ganzen Welt im Sturm eroberte, erschien eine immer größere Vielzahl von festprogrammierten Spielen für den Hausgebrauch. Viele davon waren keine Fernsehspiele, sondern kleine tragbare Geräte mit eigenen Anzeigen in der Art der ersten Videospiele. Auch in anderen Anwendungsbereichen wurde Computerleistung zum Spaß und zur Unterhaltung eingesetzt. Es gab mit Computern ausgerüstete Roboterspielzeuge, die mit den Kindern plappern konnten, und andere Computer, die ihnen beim Buchstabieren helfen konnten.
Die ersten tragbaren Geräte beherrschten nur jeweils ein einziges Spiel; heute können einige dieser Geräte Kassetten aufnehmen, die sie in die Lage versetzen, eine Vielzahl von Spielen zu spielen. Jede Kassette enthält das Programm für ein spezielles Spiel fest in ihren Speicherzellen verankert. Alle diese Spiele wurden durch die »Explosion« der Computertechnologie ermöglicht, welche dann, und das ist für uns besonders wichtig, zum Erscheinen der ersten Heimcomputer führte.

Heimcomputer

Die allerersten preiswerten Computer – wie der Altair, der 1975 in den USA erschien – waren keine richtigen Heimcomputer im heutigen Sinn. Sie waren Hobbymaschinen, zusammengebaut von Elektronikfreaks in ihrer Freizeit, für die das wichtigste war, ihre Geräte überhaupt zum Arbeiten zu bewegen. Ein Großteil des Vergnügens lag für sie im Zusammenbauen und Basteln an den Teilen. Es gab nicht viele fertige Spiele für diese selbstgemachten Meisterstücke; es waren reine Selbstzweckprojekte einiger Bastler.

Selbst als 1980/81 die überaus populären Computer Sinclair ZX80 und ZX81 erschienen, wurden sie ebenso als Bausatz wie als Fertiggeräte verkauft. Viele der ersten Käufer wollten herausfinden, wie ein Computer funktioniert, und nicht Spiele darauf laufen lassen. Nur langsam gingen die Stückzahlen der Bausätze zurück, bis sie aufgegeben und nur noch fertige Geräte verkauft wurden.

Mit der Entwicklung des Heimcomputermarktes entschlossen sich immer mehr Leute, die überhaupt nichts von Elektronik oder Programmieren verstanden, einen Heimcomputer zu kaufen. Sie wollten nicht den ganzen Abend mit dem Lötkolben hantieren, sondern spielen! Die Industrie stellt sich immer mehr auf diese Tatsache ein. Man geht heute davon aus, daß der Kauf eines Computers nur der Anfang ist. Wenn Sie sich einmal ein Gerät gekauft haben, werden Sie wahrscheinlich mindestens noch einmal so viel für Spiele ausgeben.

Zuerst gab es die Spiele nur auf Tonbandkassetten. Sie waren recht preiswert, aber umständlich zu benutzen. Heutzutage kommen mehr und mehr fertige Programme in Einschubkassetten auf den Markt. Der Einschub enthält Speicherbausteine, die bereits mit den fertigen Programmen beschrieben sind. Es handelt sich um ROM-Speicher (Read-Only-Memory), die vom Benutzer nicht beschrieben werden können, anders als die käuflichen Speichererweiterungen. Der Benutzer muß lediglich noch die Kassette einschieben und das Gerät einschalten.

Kompaktkassetten und ROM-Kassetten
Computerspiele sind normalerweise in beiden Formen erhältlich. ROM-Kassetten sind besser, aber sie können auch viel teurer sein.

Spiele und Grafikprogramme

Warum kaufen sich die Leute Heimcomputer mit so vielen Einsatzmöglichkeiten, anstatt sich eine der immer ausgereifteren Spielmaschinen zuzulegen? Schließlich geht ein Teil des Geldes, das für den Heimcomputer ausgegeben wird, in Elemente, die seine Universalität erst ermöglichen, wie etwa die Tastatur. Wenn Sie wirklich nur spielen wollen, ist ein reines Fernsehspiel sicher der bessere Kauf. Andererseits eröffnet der Heimcomputer völlig andere Anwendungsbereiche. Sie können sich verschiedene Fertigprogramme zulegen, beispielsweise für die Verwaltung des Haushaltsgeldes, Geschäftsprogramme oder Lernprogramme. Und selbstverständlich können Sie für alles Ihre eigenen Programme schreiben, von der Steuerung einer Modelleisenbahn bis zur Erledigung der Hausaufgaben in Erdkunde.

Wir werden uns um das Schreiben von Spiel- und Grafikprogrammen später in diesem Buch kümmern. Für viele Leute ist das Schreiben von Programmen mindestens ebenso befriedigend wie das Benutzen der fertigen Programme. Jedoch nur dann, wenn Sie ein wirklicher Experte im Programmieren sind, können Sie erwarten, daß Ihre selbstgeschriebenen Spiele so gut sind wie die gekauften. Die schnellsten Actionspiele sind außerdem nicht im leicht erlernbaren BASIC geschrieben, sondern in der sogenannten »Maschinensprache« des Computers – einer Folge von Nullen und Einsen, die nicht so leicht zu handhaben ist.

Sie können diese Lücke zwischen Einschubkassetten und Tonbandkassetten einerseits und Ihren Eigenprodukten andererseits überbrücken, indem Sie Computerzeitschriften und -bücher kaufen und die mitgelieferten Programmlisten ausprobieren. Wenn Sie die Zeit haben, die Listings einzutippen, können Sie gegenüber gekauften Spielen eine Menge Geld sparen. Wir werden diese Möglichkeit später ebenfalls noch ausführlicher bsprechen.

Ihr Heimcomputer ist eine sehr leistungsfähige Maschine, mit guten Programmen, auf der Sie, vorausgesetzt Sie verstehen sie, eine Vielzahl von aufregenden und phantasieanregenden Spielen laufen lassen können. Sie dürften enttäuscht sein, wenn Sie erwarten, ihn in einen großen Videospielautomaten zu verwandeln: Diese festprogrammierten Geräte sind den Vielzweckcomputern mit Sicherheit immer einen Schritt voraus. Trotzdem: Es gibt mehr als genug, was Sie zufriedenstellen wird: Spielen – und dazu auch Lernen!

Die verschiedenen Arten von Computerspielen

Welche Arten von Computerspielen können Sie kaufen oder selbst programmieren? In diesem Kapitel werden wir uns die Möglichkeiten ansehen, die Spiele bieten, und Wege aufzeigen, wie man Spiele erkennt, die das können, wonach man sucht.

Bekannte Spiele

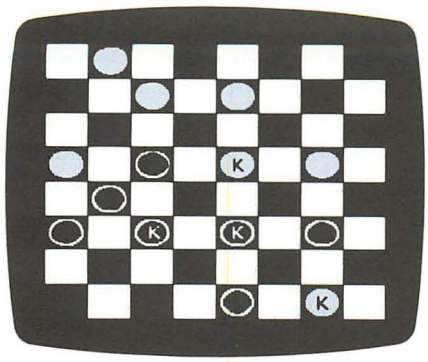
Viele Computerspiele sind natürlich alte Bekannte, die für den Computer aufpoliert wurden. Brettspiele, Kartenspiele und Schreibspiele sind alle an den kleinen Bildschirm angepaßt worden. Es ist gar nicht so unwahrscheinlich, daß für Ihr Lieblingsspiel – egal, ob es nun Pfeilwerfen oder Backgammon ist –, irgendwo irgendjemand bereits ein Computerprogramm geschrieben hat.
Warum soll man überhaupt ein Spiel wie Schach oder Hangman auf dem Computer spielen, wenn man es zu Hause mit wenig oder gar keinen Utensilien spielen kann? Hier sind einige der Vorteile:

Der Computer ist immer spielbereit

Außer bei Solitaire und Patiencen (und selbst dafür gibt es Computerspiele) braucht man bei den meisten Spielen mindestens zwei Spieler. Wenn gerade niemand da ist, um mit Ihnen Ihr Lieblingsspiel zu spielen, ist es schön zu wissen, daß der Computer immer einsatzbereit und willig ist. Er hat immer genauso lange Lust zu spielen wie Sie, selbst wenn er gerade eine Pechsträhne hat.

Der Computer führt die ganzen unangenehmen Arbeiten für Sie aus

Niemand will Punkte notieren? Kein Problem: Der Computer erledigt das; außerdem mischt er die Karten und verteilt sie.



Bekannte Spiele
So gut ein Computer beim Schachspiel ist, bei Dame (links) ist er noch viel besser. Weil Dame ein einfacheres Spiel ist, gibt es weniger Möglichkeiten zu berechnen, so daß der Computer hier noch viel besser seine Fähigkeit der Vorausberechnung zum Tragen bringen kann. Ein gutes Spiel, um die Seiten zu wechseln, wenn man auf verlorenem Posten steht!

Es gibt viele kommerzielle Versionen von »Hangman« (links). Sie können aber auch Ihre eigene Variante schreiben, wenn Sie wollen. Normalerweise hat »Hangman« etwas mit Worteraten zu tun, aber einige der für Computer entwickelten Versionen benutzen Ziffern und Zahlen und sollen mathematische Fähigkeiten verbessern.

Der Computer betrügt nicht

Ein Abenteuerspiel kann Sie außer Fassung bringen, ein Schachprogramm mag schonungslos Ihre Spielschwächen aufdecken und Sie aufregen; aber Sie können sicher sein, daß ein Programm Sie niemals betrügt. Es hält die Regeln ein und warnt Sie vielleicht sogar, wenn Sie versuchen, sie zu brechen.

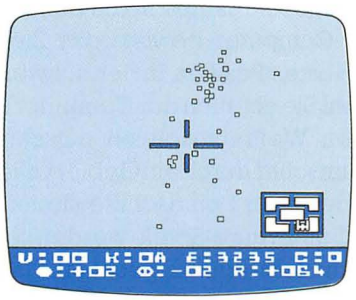
Der Computer ist ein guter Lehrer

Er sorgt nicht nur dafür, daß Sie die Schachregeln korrekt einhalten: Viele Spiele schlagen Ihnen sogar Züge vor, wenn Sie ratlos sind. Wenn Sie sich sozusagen selbst eine Grube gegraben haben, können Sie oft einfach die Seiten tauschen und sich freuen, zur Abwechslung einmal auf der Gewinnerseite zu stehen!
Obwohl es eine Menge Computerspiele gibt, die ein Pendant ohne Computer haben oder aus diesen entwickelt wurden, führen nur wenige Spiele die »Hitparade« bezüglich Popularität und Verbreitung an. Das hängt damit zusammen, daß manche Spiele einfacher zu programmieren sind als andere (Sie werden beispielsweise nur wenige

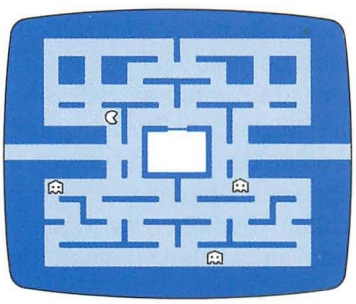
Bridgespiele für Computer finden, dafür jedoch viele Versionen des einfacheren und logisch aufgebauten Spieles »Gomuku«), teilweise aber auch damit, daß manche auf dem Computer einfacher zu spielen sind. Es ist jedoch schwer zu verstehen, warum irgend jemand Pfeilwerfen mit seinem Computer spielen will (obwohl es eine ganz gute Version davon gibt, wenn Sie wirklich Wert darauf legen sollten).
Hier sind einige der etablierten Spiele, die jetzt als Computerprogramme erhältlich sind:
Schach Braucht keine Erklärung
Dame Einfacher als Schach – aber eine gute Methode, einen regnerischen Nachmittag zu vergessen!
Backgammon Der Computer wird mit Ihnen in der üblichen Weise wie um einen Einsatz spielen, aber manche Leute meinen dennoch, daß das Spiel ohne echten Geldeinsatz viel von seinem Reiz verliert.
Blackjack In noch höherem Maße ein Glücksspiel. Es ist besser, wenn 2 oder mehr Spieler mitmachen, wobei der Computer als Bankhalter und Schiedsrichter fungiert.
TicTacToe Sowohl die konventionelle als auch eine moderne dreidimensionale Version sind erhältlich.
Hangman Die guten Versionen haben einen sehr großen Wortvorrat sowie verschiedene Schwierigkeitsgrade und können von Ihnen sogar erweitert werden.
Würfeln oder *Domino* Viele Versionen sind erhältlich.
Mastermind Eines der klassischen Ratespiele.
Nimm-Spiel Ein logisches Spiel, das schwerer ist als es scheint.
Go und *Othello* Brettspiele (und dazu eine Menge nicht so bekannter Brettspiele)
Sie können Programme für einige der einfacheren Spiele weiter hinten im Buch finden.

»Hand-und-Auge«-Spiele

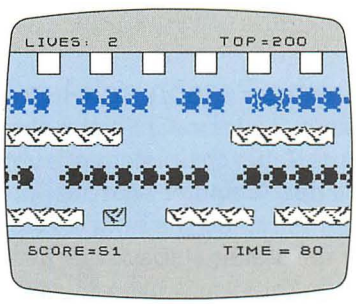
Fernsehtennis, Ping-pong, »Space Invaders« und die vielen anderen Spiele, die wir im letzten Kapitel kurz erwähnten, fallen unter die Kategorie der Spiele für Hand/Auge-Koordination. Das bedeutet nicht, daß Strategie nicht darin vorkommt – in den besseren ist sie sogar sehr wichtig –, aber das Entscheidende dabei ist, die Aktionen so zu koordinieren, daß man einen Ball trifft oder ein Raumschiff abschießt.



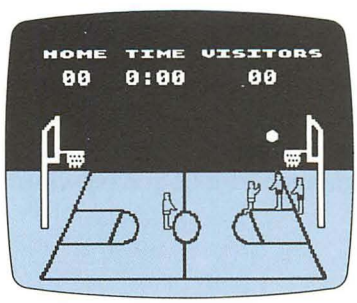
Star Raiders
In einem typischen Raumfahrtspiel bewegen sich die Außerirdischen in recht gut vorhersagbaren Mustern, so daß sich der Spieler nur auf die richtige Stelle bewegen und feuern muß. Aber manchmal schießen die Fremden zurück – oft blindlings; aber in manchen besseren Versionen nehmen sie Sie aufs Korn, um Vergeltung zu üben!



Pac-Man
Es gibt viele verschiedene Versionen dieses Spiels (sämtlich mit verschiedenen Namen). Die Grundidee ist, den Pac-Man durch das Labyrinth zu führen, Früchte und Energietabletten zu essen und den Monstern auszuweichen, die das Labyrinth bewohnen, bis er bereit ist, sie zu verschlingen.



Frogger
Hier sollen Sie dem Frosch helfen, die Sicherheit der Lilienbuchten zu erreichen, wobei er verschiedene natürliche Gefahren und die Fallen der menschlichen Zivilisation umgehen muß. Sie kontrollieren die Bewegungen des Frosches, sonst nichts.



Basketball
Nicht gerade die »Harlem Globetrotters«, aber trotzdem ein aufregendes Spiel! Die Bewegung auf dem Schirm wird mit Joysticks kontrolliert, und Sie können gegen den Computer oder einen menschlichen Gegner spielen. Basketball ist nur eines von vielen Ballspielen, die auf dem Schirm gespielt werden, auch wenn an diesem Tag kein Spiel im Fernsehen läuft.

Die einfachsten Koordinationsspiele sind wirklich *sehr* einfach. Entweder steht das Ziel still, und Sie müssen Ihren Schläger oder Ihre Abschußrampe in die richtige Stellung bringen, oder der Computer bewegt das Ziel, und Sie schießen aus einer festen Position. In den schwereren Versionen führen sowohl Sie als auch der Computer Bewegungen aus. In einigen Weltraumspielen müssen Sie beispielsweise Ihr Raumschiff durch ein Gebiet voller sich bewegender Asteroiden oder Feindschiffe steuern.
Die einfachen Hand-und-Auge-Spiele werden sehr schnell langweilig. Wenn man es einmal geschafft hat, läßt das Interesse schnell nach. Deshalb bieten die meisten außer den allereinfachsten hausgemachten Versionen mehrere Schwierigkeitsgrade: Das Spiel wird schneller, der Schläger kleiner, die Hindernisse größer oder zahlreicher. Es ist gut, ein Spiel zu wählen, bei dem vor Beginn der Schwierigkeitsgrad gewählt werden kann, so daß Sie sich nicht jedesmal durch die leichteren Ebenen durcharbeiten müssen, ehe Sie auf echte Herausforderungen treffen. Die Fähigkeiten der einzelnen Spieler variieren sehr stark; wenn die Möglichkeit besteht, zwischen mehreren Schwierigkeitsgraden zu wählen, können Sie leichter einen für Sie passenden finden.
Wodurch unterscheiden sich die einzelnen Spiele? Hier sind einige Stichpunkte, auf die Sie achten sollten:

Kann mehr als eine Person spielen?

Gibt es Optionen, einen, 2 oder mehr Spieler auszuwählen, oder ist nur eine feste Zahl von Spielern erlaubt? Falls es ein Spiel ist, bei dem der Computer zwar das Spielfeld vorgibt, aber nicht gegen Sie spielt, wie etwa in »Breakout« (in dem der Computer die Mauer aufbaut, die Bälle einwirft, aber nicht versucht, sie zurückzuschlagen), wäre es schön, zumindest gelegentlich einen Mitspieler zu finden. Die eigene Bestleistung zu übertreffen macht nicht annähernd so viel Spaß, wie jemanden anderen zu schlagen. Sie können sich jederzeit abwechseln; aber noch besser ist es, gleichzeitig zu spielen und beispielsweise zu sehen, wer den Asteroiden zuerst trifft. Viele Computerspiele erlauben das. Einige Spiele stellen Ihnen zur Auswahl, entweder gegen den Computer oder gegen einen Mitspieler zu kämpfen.

Ist das Spiel jedesmal gleich?

Einige Leute finden es langweilig, wenn beispielsweise die fliegende Untertasse in »Space Invaders« jedes zweite

Mal 15 Punkte bringt. Wenn sie solche Muster einmal erkannt haben, verlieren sie das Interesse. Andere sind glücklich, ein vorhersagbares Muster zu haben und werden nervös, wenn sie keines finden!

Wie sieht die Anzeige aus?

Ein Großteil des Interesses für diese Spiele wird von der Anzeige auf dem Bildschirm geweckt. Finden Sie das Hauptobjekt des Spieles anziehend? (Dieselben grundlegenden Ideen können in vielen verschiedenen Spielen auftauchen – Weltraumspiele, mystische Drachenspiele, Cowboys und Indianer und so fort). Ist die Anzeige gut durchdacht? Passieren interessante Dinge, wenn Sie Ihre Sache gut machen? Oder sogar, wenn Sie ein schlechtes Spiel liefern? Bei manchen Versionen von »Meteorsturm« läßt sich ein gewisses perverses Vergnügen feststellen, die Meteore nicht zu zerschießen. Die Explosionen, die sie dann selbst auslösen, sind wesentlich eindrucksvoller!
Einige dieser Merkmale hängen von der Art des gewählten Spieles ab, andere natürlich von der speziellen Version, die Sie besitzen. Wir werden die Hauptunterschiede zwischen guten und schlechten Programmen ausführlicher im Kapitel 8 besprechen, wenn wir uns der fertiggeschriebenen Software zuwenden.

Können Sie sich die Zeit einteilen?

In einigen Spielen können Sie wählen, wann Sie Ihren Zug eingeben. »Labyrinth« ist eines davon. Sie bewegen sich in Ihrem eigenen Tempo, und es gibt Momente, in denen Sie eine Pause einlegen können, ohne daß etwas Katastrophales passiert. »Breakout« ist das genaue Gegenteil. Ununterbrochen bewegt sich der Ball in Richtung Schläger, und Sie müssen ihn rechtzeitig treffen. Einige Menschen mögen schnelle Action, andere mögen Koordinationsspiele, aber ziehen es vor, alles in Ruhe zu machen.
Hier ist eine Liste der Hand-und-Auge-Spiele, die zur Zeit erhältlich sind, jeweils mit einer kurzen Beschreibung. Die Heimcomputerversionen sind, wie oben schon erwähnt, nicht identisch mit ihren großen Videospielverwandten. Außerdem haben sie, wenn sie von verschiedenen Firmen entwickelt wurden, unterschiedliche Namen.
Space Invaders Das klassische Videospiel. Viele andere sind dann nachgefolgt: »Galaxian«, »Defender« usw. Der Spieler hat normalerweise eine oder mehrere Raumbasen am unteren Bildschirmrand und verteidigt diese gegen die

böswilligerweise angreifenden Fremden, die sich in Wellen nach unten vorarbeiten.

Asteroiden Wieder ein Raumthema, aber diesmal nicht ganz so blutrünstig. Sie müssen eine Menge Asteroiden in die Luft jagen, bevor sie Ihr eigenes Schiff in Trümmer verwandeln. Manche Spiele liefern zur Abwechslung gleich noch ein paar Feindschiffe mit; trotzdem ist der Bildschirm meist ruhiger als bei den Invader-Spielen.

Pac-Man Eine ganze Serie bizarrer Spiele mit Cartoonfiguren, die sich mampfend durch ihre Miniwelt bewegen, Pluspunkte sammeln und Gefahren vermeiden. Auch einfachere Versionen sind häufig anzutreffen. Eine davon ist das Schlangenspiel, bei dem Sie eine Schlange über den Schirm bewegen, und sie veranlassen Zahlen oder Zeichen zu fressen.

Frogger Es mag unverschämt klingen, dies ein Ökologiespiel zu nennen. Dennoch ist es ein friedlicheres Spiel als viele der anderen Videospiele. Ein Frosch muß nach Hause geführt werden, wobei Hindernisse wie stark befahrene Straßen oder schnellfließende Flüsse zu überwinden sind.

Breakout Eine Familie von Ball-und-Schläger-Spielen, in denen eine Mauer abgerissen werden muß, indem Stein für Stein mit dem Ball getroffen wird. Ein einfaches Spiel zum Selbstprogrammieren. (Unsere Version findet sich weiter hinten im Buch.)

Fußball und Tennis Es gibt Unmengen von Variationen, wie solche Ballspiele auf dem Bildschirm imitiert werden. Meistens kontrollieren Sie einen oder mehrere Spieler und spielen entweder gegen den Computer oder einen Mitspieler, der die andere Mannschaft kontrolliert. Andere computerisierte Ballspiele sind Basketball, Squash, Flipper, Golf und Billard.

Simulationsspiele

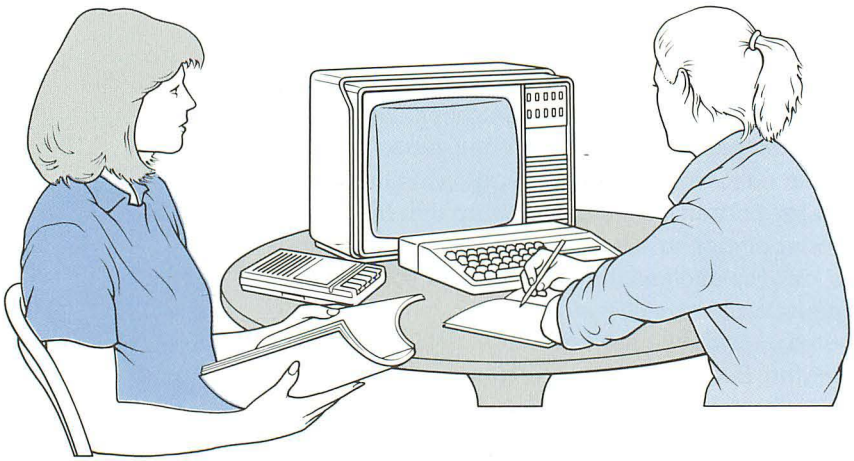
In einem gewissen Sinn sind natürlich alle Videospiele, die wirkliche Ereignisse auf dem Schirm imitieren, Simulationen. Wir wollen diese Bezeichnung jedoch auf Spiele der Kategorie »Was wäre wenn« beschränken, wie »Life« oder »Kingdom«.

Diese Spiele simulieren eine Situation aus dem »wirklichen Leben« – manchmal komplex, manchmal sehr einfach –, die der Spieler kontrollieren darf. Im Spiel »Life« (Leben) darf man Mutter Natur sein und die Geburt und

das Sterben von Organismen planen. Sie können als König oder Regierungschef die Wirtschaft lenken und die Steuerpolitik planen, wie in den Spielen »Kingdom« oder »Great Britain Limited«. In »Lunar Lander« können Sie ein Raumfahrer sein, der eine Rakete auf dem Mond landet. Der Computer erwartet von Ihnen Entscheidungen und verändert abhängig davon seine Modellwerte. Werden Sie abstürzen oder sicher landen? Wird das Land Sie als Regenten akzeptieren oder stürzen? Wird Ihre Mikrowelt erblühen oder eingehen? Es ist auf jeden Fall beruhigend zu wissen, daß Sie auch nach Fehlern immer noch einmal neu anfangen können.
Bei einigen dieser Spiele liegt die Attraktion ebenfalls in den gezeigten Bildern. Einige Versionen des Lebensspiels können zum Beispiel für die Simulation des Wachstums und Vergehens einer Anzahl von Organismen sehr attraktive Muster auf den Bildschirm zaubern. In anderen Spielen können Sie erleben, wie Ihr Land erobert oder überflutet wird, wenn Sie schlecht regieren (wieder ein Fall, in dem Verlieren mehr Spaß als Gewinnen machen kann: Katastrophen lassen sich leichter illustrieren als Gewinne!). Manche Versionen benutzen jedoch nur Text. Der Computer beschreibt die Ergebnisse Ihrer Arbeit in Worten oder Zahlen, aber ohne jegliche Illustration.
Die Vermutung könnte naheliegen, daß die Textversionen deshalb unterlegen seien. Weit gefehlt! Manche von ihnen benutzen den Computerspeicher so extensiv für ihre komplexen Modelle, daß einfach kein Platz mehr für dramatische Bilder bleibt. Die Anziehungskraft mag anders geartet sein, aber für manche Leute ist sie genauso groß.

Abenteuerspiele

Wir sprachen im vorigen Kapitel bereits kurz über Abenteuerspiele. Sie sind, wenn man so will, natürliche Erweiterungen der Simulationen. Der Hauptunterschied besteht darin, daß sie nicht die wirkliche Welt imitieren, sondern ein eigenes Phantasiereich erschaffen.
Das Ziel des Spielers in einem Abenteuerspiel ist es nicht, zu versuchen, die Welt zu verbessern, sondern sich hineinzuwagen und ein Problem zu lösen oder einen Schatz zu finden. Einige einfache »Schnell«-Abenteuerspiele ähneln mehr der Beantwortung eines Rätsels: Haben Sie erst einmal die richtige Idee (was eine Weile dauern kann, da



Abenteuerspiele
Nicht alle Abenteuerspiele bieten Bilder, aber einige von denen, die es tun, offerieren wirklich aufregende! Die besten Programme sind auf Diskette. Auf Kassetten finden Sie eher comicartige Figuren und impressionistische Hintergründe. Sie sind normalerweise unbewegt; es gibt nichts zu kontrollieren. Das Wichtigste bei Abenteuerspielen ist der Text, für den normalerweise ein eigenes Fenster am unteren Bildrand vorgesehen ist, wie hier gezeigt.

Abenteuerspiele nicht allzuviel erklären) und erreichen das Ziel, gibt es nichts mehr zu tun. Die längeren und komplizierteren Spiele entwickeln den Grundgedanken fast unbegrenzt weiter. Sie sind so ausgearbeitet, daß nur eine kleine Minderheit der Spieler sie jemals beenden wird. Das Vergnügen liegt darin, sich Stück für Stück dem Ziel immer wieder ein wenig zu nähern.

Wieviel Spielzeit ist von einem Abenteuerspiel zu erwarten? Das ist etwas, was ganz enorm variieren kann. Passionierte Abenteuerer können Hunderte von Stunden mit dem gleichen Spiel glücklich sein. Ein wirklicher Experte dürfte allerdings unter Umständen ein Spiel in ein paar Stunden lösen, obwohl in den besseren Versionen gewöhnlich Zufallselemente das Interesse wachhalten. Der Anfänger schafft vielleicht noch nicht einmal den Einstieg. Warum? Weil er womöglich jahrelang auf den Einleitungs-

text starrt, ohne zu verstehen, wie er irgendetwas bewirken kann. Aber Achtung: In den meisten Abenteuerspielen passiert zumindest am Anfang gar nicht viel. Seien Sie geduldig! Lesen Sie jeden Hinweis, der im Spiel auftaucht, oder andere, die man in Besprechungen dieses Spiels findet! Geben Sie dem Computer viele Befehle – auch solche, die nicht in den Regeln erwähnt sind. Man wird das Programm kaum zum »Absturz« bringen, aber wahrscheinlich herausfinden, daß irgendeine wunderliche Instruktion genau das war, was das Programm erwartete.

Wie bei den Simulationen sind auch viele Abenteuerspiele nur mit Text versehen. Der Speicher des Computers und die Aufmerksamkeit des Spielers sind auf das Spielgeschehen gerichtet und nicht auf grafische Schnörkel. Jedoch kommen allmählich auch bebilderte Abenteuerspiele auf den Markt. Neuere Versionen auf Floppy-Disks, die mehr Speicher zur Verfügung stellen, sind wirklich ausgereift und haben für jede neue Örtlichkeit oder Situation, in die der Abenteurer gerät, ein Bild parat.

Einige Abenteuerspiele sind in »Echtzeit« programmiert, das heißt, Ereignisse können eintreten, ohne daß der Computer eine Eingabe erhalten hat (Sie befinden sich in der Burg und grübeln gerade in der Königshalle vor sich hin, als die Meldung erscheint: »Die neunköpfige Hydra ist erwacht und wird Sie gleich fressen«). Ein Bestandteil solcher Spiele, nach dem gesucht werden sollte, ist das sogenannte Pausenkommando: Dieses Kommando friert den aktuellen Spielstand ein, während Sie sich eine Tasse Kaffee bereiten können oder irgend etwas anderes. Sonst riskieren Sie, daß Sie jedesmal bei Ihrer Rückkehr neu anfangen müssen, da Sie in der Zwischenzeit getötet wurden!

Auch hier ist es eine gute Idee, sich ein Thema auszusuchen, das einem gefällt. Es gibt mythologische Abenteuer, die auf den Abenteuern von Odysseus, Jason oder anderen bekannten Figuren basieren. Es gibt Spiele, denen Bücher wie »Der kleine Hobbit« (oder berühmter: James Bond) zugrunde liegen. Es gibt historische Spiele, die im alten Rom oder im Inka-Königreich spielen. Es gibt technologische Spiele in Raumstationen und Detektivgeschichten im Stil der 30er Jahre und außerdem noch vieles andere.

Kriegsspiele stellen schon fast eine eigene Kategorie dar. Die Versionen für Spielbretter sind schon immens kompliziert, aber ihre computerisierten Äquivalente sind es noch viel mehr. Die Rekonstruktionen wirklicher Schlachten können lehrreich sein, aber Sie werden vielleicht auch feststellen, daß der Programmierer von Anfang an eine ganze Menge Wissen voraussetzt.

Lehrspiele

Wie wir bereits früher erwähnten, gibt es eine riesige Vielzahl von Lehrspielen. Einige sind es von der Grundidee her, sind aber als Spiel verpackt, um den Stoff schmackhafter zu machen. Viele benutzen einfache Grafiken, um den Spieler für eine richtige Antwort zu belohnen – eine richtige Antwort im Französischtest, und Sie dürfen eine Rakete abschießen. Andere verpacken die ganze Übung in Spielform: »Wortjagd« oder »Hangman« üben das Buchstabieren. Im »Mathematischen Irrgarten« macht das Rechnenlernen etwas mehr Spaß.

Superlative

Das Längste

Das längste uns bekannte Computerspiel ist ein Abenteuerspiel, das für den Apple II Computer geschrieben wurde: »Time Zone« von Ken und Roberta Williams. Es umfaßt 6 Disketten und enthält über 1400 verschiedene Bilder!

Die Bestleistung

Videospielexperten können so gut werden, daß ein einziges Spiel stundenlang dauert. Der Höchstpunktstand bei »Defender« wurde von einem amerikanischen Schüler notiert, der auch die ganze Nacht spielte (insgesamt 16 Stunden und 34 Minuten), um etwa 16 Millionen Punkte zu erreichen!

Computergrafik

Am Anfang dieses Kapitels wollen wir uns kurz der Frage widmen, wie der Computer Informationen für seine Bildschirmanzeige verarbeitet und speichert.
Sie wissen eventuell schon, daß der Computer Zahlen und Buchstaben (überhaupt alle Informationen, die er verarbeitet) in einen Binärcode umwandelt, so daß sie aus einer Folge von Nullen und Einsen bestehen. Kleine Heimcomputer verarbeiten normalerweise Gruppen von 8 »Bit« (Bezeichnung für ein Binärzeichen, »binary digit«), die auch als Byte bezeichnet werden. So werden der Buchstabe »A« beispielsweise als das Byte 01000001 und die Zahl 15 als 00001111 codiert.
Nach dem gleichen System speichert der Computer die Informationen für die von ihm »produzierte« Bildschirmanzeige. Als ganz einfache Vorstellung können wir einmal annehmen, daß der Computer den Bildschirm in eine Menge kleiner Punkte oder Rechtecke aufteilt, von denen jeder bzw. jedes entweder schwarz oder weiß sein kann. Der Computer merkt sich Schwarz beispielsweise als 0 und Weiß als 1 und reserviert einen Teil seines Speichers, um sich die Einzelheiten des Bildschirmaufbaus zu merken. Dies ist der Bildschirmspeicher, in dem jedes gespeicherte Byte seinen festen Platz – eine »Adresse« hat.
Nehmen wir einmal an, die Adressen begännen mit 0020 und der Computer merkte sich die Eigenschaften der Zeilen in aufeinanderfolgenden Speicherzellen, dann würde das folgende Muster

0020	0021	0022
01010101	11111111	00000000

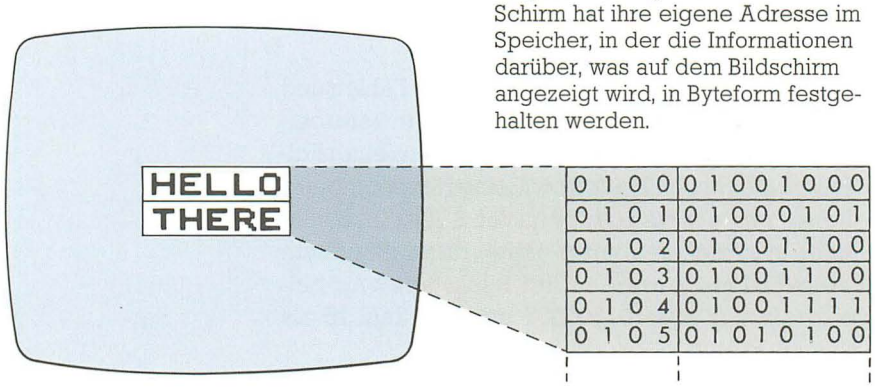
in der linken oberen Ecke des Schirms erst eine Reihe wechselnder schwarzer und weißer Punkte, dann eine weiße Reihe und dann eine schwarze Reihe bewirken.
Wenn der Computer in Farbe arbeitet (auf einem Schwarzweißfernsehgerät wären das verschiedene Grautöne), muß er sich außerdem noch die Farbe jedes Punktes merken. Mit 3 Bit könnte er sich den Code für 8 Farben merken:

000 Schwarz	100 Blau
001 Rot	101 Purpur (Magenta)
010 Grün	110 Cyan (Hellblau)
011 Gelb	111 Weiß

3 Bits sind allerdings eine ziemlich krumme Zahl für eine Maschine, die normalerweise mit Gruppen von 8 Bit arbei-

Bildschirmspeicherkarte

Der Computer erhält immer ein Speicherabbild der Bildschirmanzeige aufrecht. Jede Stelle auf dem Schirm hat ihre eigene Adresse im Speicher, in der die Informationen darüber, was auf dem Bildschirm angezeigt wird, in Byteform festgehalten werden.



tet. Es ist für den Computer einfacher, ein ganzes Byte zur Beschreibung zweier Punkte zu verwenden, wobei dann für jeden Punkt 4 Bit zur Verfügung stehen. Das zusätzliche 4. Bit könnte die Menge der Farben von 8 auf 16 erhöhen oder andere Eigenschaften wie etwa »blinkend« oder »extra hell« beschreiben.
Schauen wir uns doch mit dem eben beschriebenen Verfahren einmal ein Beispiel an! Das linke der 4 Bits soll dann den »Blink-Code« darstellen: 0 = aus, 1 = an. Dann würde das oben genannte Muster der 3 Bytes folgendes bedeuten:

2 purpurne Punkte, nicht blinkend	(01010101)
2 blinkende weiße Punkte	(11111111)
2 schwarze Punkte, nicht blinkend	(00000000)

Wunderbar, denken Sie jetzt wahrscheinlich – und tatsächlich funktioniert es auch wirklich sehr gut. Wo liegt das Problem? Ganz einfach in der Menge des benötigten Speicherplatzes. Eine typische niedrigauflösende Grafikanzeige könnte beispielsweise aus 64 Spalten und 32 Reihen von Punkten bestehen. Das sind 64 × 32 oder 2048 einzelne Punkte, die zu beschreiben sind. Auch mit 2 Punkten pro Byte werden immer noch 1 kB (1024 Bytes) Speicherplatz benötigt, um diesen Bildschirm zu beschreiben. Mit anderen Worten: Der gesamte Schreib-Lese-Speicher (RAM) eines kleinen Heimcomputers, etwa des Sinclair ZX81/ Timex 1000 würde benötigt, um nur die Informationen einer einzigen farbigen Bildschirmdarstellung zu speichern.

Natürlich wird die Sache noch schlimmer, wenn wir versuchen, die Auflösung eines Bildes zu verbessern, indem wir mehr Punkte auf den Schirm bringen. Die hochauflösende Anzeige eines Heimcomputers könnte etwa 256 Spalten und 192 Zeilen umfassen. Das sind dann insgesamt 49152 Bildpunkte. Mit 4 Bit für jeden dieser Punkte bräuchten Sie dann insgesamt 24 Kilobyte Speicher – einen Großteil des RAMs eines recht leistungsfähigen Heimcomputers.
Daraus können wir 2 Schlußfolgerungen ziehen: Erstens gibt es eine enge Verbindung zwischen der Speichergröße eines Computers und der Bildqualität, die er liefern kann – es ist sinnlos, von einem Computerwinzling überzeugende, 16farbige, bewegte Grafiken zu erwarten. Zweitens gibt es gute Gründe, warum Computerhersteller versuchen, die Bildschirmdaten in so wenig Speicher wie möglich zu packen. So können sie Ihnen einerseits die Möglichkeit geben, gute Bilder zu erzeugen, und andererseits immer noch Speicher für Sie übriglassen, um Programme eingeben und laufen lassen zu können.
Alle Computer reservieren einen Teil des Speichers für Bildschirmdaten. In manchen Computern ist er völlig vom Rest des Speichers getrennt. In anderen gibt es fließende Übergänge, so daß Sie dem Computer sagen können, wo die Grenze zwischen Arbeitsspeicher und Bildschirmspeicher liegen soll, abhängig von der Art der Anzeige, die Sie möchten. Viele Computer reservieren einen recht großen Teil ihres Speichers, in dem Sie dann – wie besprochen – Punkt für Punkt die gewünschten Anzeigen definieren können. (Wir werden später noch sehen, wie Sie per Programm diese Details in den Computerspeicher eintragen.)
Nichtsdestotrotz wenden beinahe alle Computer irgendwelche Tricks an, um möglichst viele Informationen für die Bildschirmdarstellung aus einem begrenzten Speicherplatz zu zaubern.
Der »Haken« dabei ist, daß alle diese Tricks dem Programmierer das Leben schwerer machen. Oft bedeutet das, daß Sie sich um Abläufe im Inneren des Computers kümmern müssen, wenn Sie Grafiken programmieren, etwas, das Sie bei anderen Programmen in höheren Programmiersprachen wie etwa BASIC ignorieren können. Es ergeben sich auch Einschränkungen für die Art der Bilder, die Sie erzeugen möchten. Sie können unter Umständen die gesamte Farbskala des Computers nur dann ausnutzen, wenn Sie mit einer ziemlich geringen Auflösung arbeiten. Oder Sie können vielleicht Text und Grafik auf dem Bildschirm nicht vermischen.
Die Art dieser Einschränkungen variiert von Computer zu Computer. Aus Platzgründen können wir hier nicht die ge-

nauen Eigenschaften eines jeden Modells aufführen. Statt dessen hoffen wir, daß wir durch die Verdeutlichung der grundlegenden Prinzipien Ihnen das Verständnis Ihres eigenen Computers erleichtern – oder die Auswahl, falls Sie noch keinen besitzen.

Textanzeige

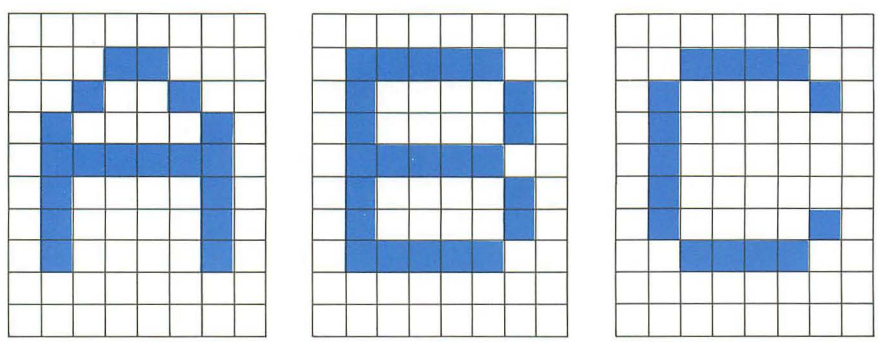
Farbige Rechtecke oder Punkte sind ja nun abgehandelt – aber was ist mit Zeichen? Wie kommt der Computer mit ihnen zurecht?

Jedes Zeichen – Buchstabe, Zahl oder irgend etwas anderes –, das auf dem Schirm erscheint, besteht aus einem Muster von Punkten. Das Punktmuster wird hochauflösend gezeichnet, so daß sich die Punkte auf einem normalen Fernsehschirm manchmal überlappen. Der Computer kann sie dennoch unterscheiden. Wenn er nun beispielsweise eine 8 × 10-Matrix benutzt, um seine Zeichen anzuzeigen, könnten A, B und C so aussehen, wie unten gezeigt.

So nehmen sich etwa die Buchstaben aus dem Zeichensatz des »NewBrain«-Computers aus, der davon 4 verschiedene besitzt. Einige sind mit einer 8 × 8-Matrix definiert, andere mit 8 × 10. Das sind auch die Werte, die bei den meisten kleinen Computern verwendet werden.

Jedesmal, wenn Sie eine Taste der Tastatur drücken oder dem Computer einen Zeichencode mit einem Kommando wie etwa PRINT CHR$(56) übergeben, muß der Computer den Code des gewünschten Zeichens zu dem benötigten

Zeichen auf einem Bildschirm



Punktmuster in Beziehung setzen, um das Zeichen auf dem Schirm erscheinen zu lassen. Der sogenannte Zeichengenerator des Computers macht das normalerweise automatisch, indem er sich auf ein gespeichertes Punktmuster im Speicher bezieht: eines für jedes Zeichen im Zeichensatz des Computers. Diese Muster sind in nicht löschbaren Speichern gespeichert (ROM).
Bei einigen Computern ist es jedoch auch möglich, den Zeichengenerator so zu steuern, daß er seine Muster aus einem von Ihnen dafür reservierten Bereich im RAM-Speicher holt. Mit dieser Methode (in den Feinheiten gibt es geringe Unterschiede) ist es möglich, den Zeichensatz ganz oder teilweise umzudefinieren oder etwa neue Zeichen hinzuzufügen. Wir werden uns das bald anschauen.
Wenn der Computer Zeichen auf den Bildschirm druckt, betrachten wir das als niedrige Auflösung. Das stimmt natürlich nur in einem gewissen Sinne. Der Computer benutzt genausoviele Punkte wie in der hochauflösenden Betriebsart, aber wir sehen den Bildschirm jetzt in eine Menge größerer Rechtecke aufgeteilt, von denen jedes aus einer Punktmatrix besteht, mit der ein Zeichen dargestellt wird.
Wie arbeitet der Computerspeicher in der niedrigen Auflösung? Für jedes dieser Rechtecke müssen 3 Informationen gespeichert werden:

- der Code für das anzuzeigende Zeichen
- die Hintergrund- oder »Papier«-Farbe
- die Vordergrund- oder »Tinten«-Farbe.

Die verschiedenen Computer verwenden unterschiedliche Organisationsmethoden, um diese Informationen im Speicher zu halten. Ganz allgemein kann man aber sagen, daß die niedrigere Auflösung weniger Speicher benötigt als die hohe Auflösung, die man für detailreiche grafische Arbeiten braucht.

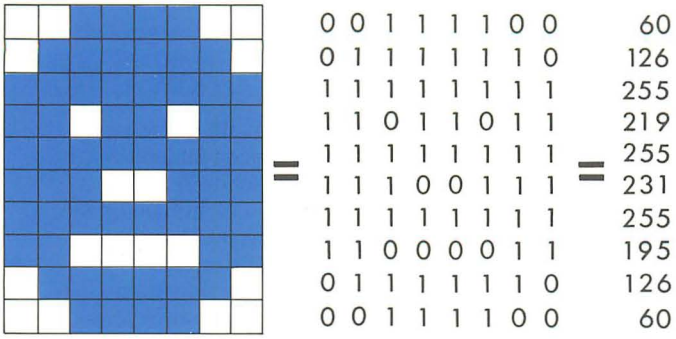
Grafische Zeichen und selbstdefinierte Zeichen

Für den Computer ist es absolut kein Problem, anstatt den Code für »A« herauszusuchen, ein Muster wie das hier gezeigte anzuzeigen. Dies ist ein grafisches Zeichen. Ein Computer besitzt normalerweise einen ganzen vordefinierten Satz solcher Zeichen, die Sie von ihm genauso leicht darstellen lassen können wie die alphanumerischen.

Manchmal können Sie die normalen Tasten der Tastatur dafür verwenden, indem Sie zuerst eine spezielle »Grafik«-Taste betätigen, manchmal müssen Sie dem Computer den Zeichencode in einem Befehl wie PRINT CHR$ übergeben. Auf jeden Fall können Sie mit diesen fertigen Zeichen den Computer recht komplizierte Muster erzeugen lassen und trotzdem mit der niedrigen Auflösung arbeiten, die normalerweise für Text vorgesehen ist.
Mit manchen Computern können Sie genauso wie mit den vordefinierten Zeichen mit selbstdefinierten Zeichen arbeiten. Vielleicht möchten Sie in der Zeichenmatrix ein kleines Gesicht oder eine Schachfigur definieren. Sie brauchen spezielle Kommandos, um dem Computer die Informationen zur Neudefinition des Zeichens zu geben. Normalerweise sagen Sie dem Computer, mit welchem »Code« Sie sich auf das Zeichen beziehen möchten, und Sie geben ihm das Muster aus 0 und 1 in jeder Reihe der Matrix. Um das kleine Gesicht, das auf der nächsten Seite gezeigt ist, zu erzeugen, müßten Sie in die Speicherzelle für die Reihe 1 den Code 00111100 (oder das dezimale oder hexadezimale Äquivalent dieser Binärzahl) schreiben und so weiter. Solche Zeichen werden benutzerdefinierte Zeichen genannt.

Mittlere und hohe Auflösung

Was heißt »hohe Auflösung«? Die Antwort ist relativ. Für den Besitzer eines Tandy Color Computers sind es 256 × 192 Punkte. Auf dem Apple II sind es 280 × 192 Punkte. Für den Besitzer des NewBrain (schwarz/weiß) sind es 640 × 250 Punkte. Für den Benutzer eines speziellen CAD-Gerätes (computerunterstützte Konstruktion) kann es sehr viel mehr sein.
In der Praxis gibt es natürlich eine Grenze für die Abbildungsqualität, die man von einem preiswerten Heimcomputer in Verbindung mit dem häuslichen Fernseher erwarten kann. Was nutzt es, ein anzuzeigendes Bild in fantastischen Details beschreiben zu können, wenn die zur Anzeige verwendete Hardware nicht mit einer vergleichbaren Genauigkeit arbeiten kann?
Genauso wie damit ein bestimmter Grad der Auflösung beschrieben wird, bezieht sich der Ausdruck »hohe Auflösung« normalerweise auch auf einen bestimmten Satz von Kommandos – in BASIC oder einer anderen Sprache wie



Das Definieren von eigenen Zeichen

LOGO – und auf eine spezielle Art, den Schirm zu benutzen. Hochauflösende Anzeigen konzentrieren sich meistens darauf, exakte Linien zu zeichnen und die Flächen, die sie umschließen, auszumalen (typische BASIC-Kommandos sind MOVE, DRAW und PAINT). Sie dienen dazu, »Geschäftsgrafik« darzustellen – beispielsweise Diagramme und Histogramme –, und um detaillierte Muster und Bilder zu zeichnen. (Ein Histogramm ist die grafische Darstellung einer Häufigkeitsverteilung in Form von Säulen.) Wir werden später im Buch dafür Beispiele finden.

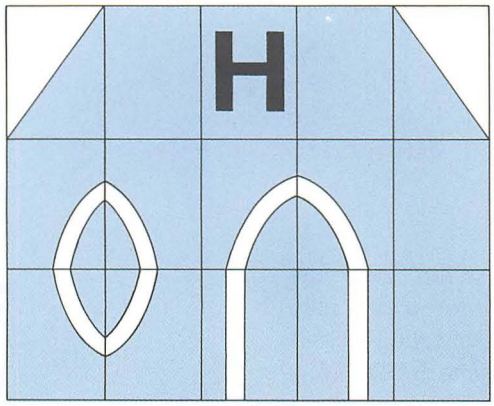
Manchmal wird der Ausdruck »mittlere Auflösung« für Anzeigen verwendet, die nicht in der normalen »Zeichen«-Art arbeiten, aber auch keine hochauflösende Grafik anbieten. Eine typische mittlere Auflösung stellt die doppelte Textdichte dar: auf dem Tandy Color Computer ist das beispielsweise 64 × 32.

Grafiken mittlerer Auflösung benutzen »Pixel«, der Ausdruck für kleine Rechtecke in Farbe, die größer als die Punkte der hohen Auflösung sind. Die Anzeigen werden aufgebaut, indem bestimmte Pixel mit bestimmten Farben versehen werden, anstatt daß wie in der hohen Auflösung Linien gezogen werden. Es gibt oft spezielle BASIC-Kommandos, mit denen Sie gewünschte Pixel setzen oder »rücksetzen« (SET und RESET) können.

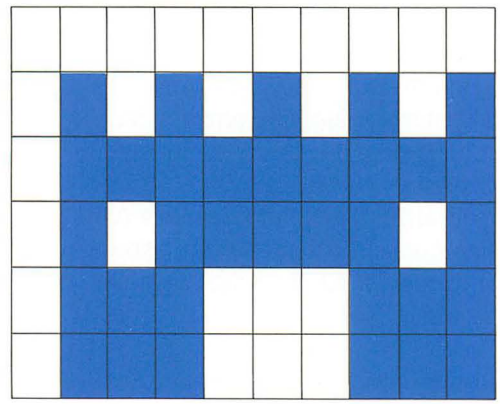
Anzeigearten und Fenstertechnik

Einige Computer bieten nur begrenzte Möglichkeiten, Text, grafische Zeichen, Pixel und hochauflösende Elemente zu mischen. Der Computer arbeitet in verschiedenen Betriebsarten, in denen der Schirm auf eine bestimmte

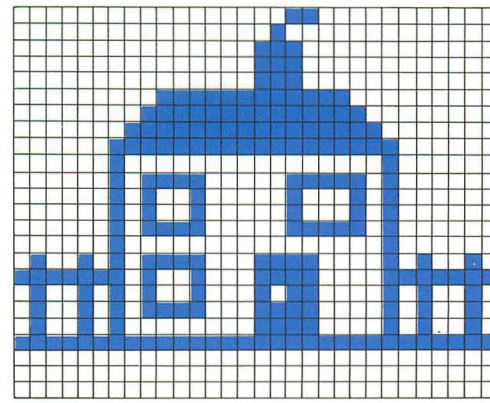
Niedrige, mittlere und hohe Auflösung



Links sieht man ein typisches niedrigaufgelöstes Bild mit Text und grafischen Zeichen. Die grafischen Zeichen unterscheiden sich von Computer zu Computer. Die im Bild ähneln denen der NewBrain-, Sharp- oder Commodore-Computer. Grafiken mittlerer Auflösung benutzen Pixel (siehe Mitte). Pixel verdoppeln die Dichte der niedrigauflösenden Rechtecke – das heißt, es gibt auf der gleichen Fläche viermal so viele, wodurch auch mehr Einzelheiten angezeigt werden können.



Mit hochauflösender Grafik lassen sich präzise Linien zeichnen und nicht nur Pixel in verschiedenen Farben setzen. Felder zwischen Linien können mit speziellen BASIC-Kommandos eingefärbt werden. So liefert die hohe Auflösung die detailreichsten Bilder, verbraucht aber auch mehr Speicher als die anderen Betriebsarten. Außerdem kann auf manchen Computern die hohe Auflösung nicht mit den anderen Betriebsarten kombiniert werden.



Weise benutzt wird und spezielle Kommandos zur Anwendung kommen. Bevor Sie Ihre Grafiken programmieren, müssen Sie dem Computer sagen, welche Betriebsart Sie wünschen.

Wir wollen uns einige Beispiele anschauen! Auf dem Tandy Color Computer mit Extended BASIC und auf dem Dragon 32 – zwei sehr ähnlichen Maschinen – sind Text- und Pixelmodus kompatibel. Sie können einen Teil des Schirms verwenden, um Meldungen auszugeben, und auf dem restlichen Schirm beliebige Pixel an- oder ausschalten und die jeweils unterschiedliche Auflösung benutzen (32 × 16 für Text und 64 × 32 für mittlere Auflösung). Sie können aber keinen dieser Modi mit hochauflösender Grafik mischen. Um die hochauflösende Grafik zu benutzen, müssen Sie dem Computer ein entsprechendes PMODE-Kommando geben. (Es gibt 4 verschiedene hochauflösende Modi, die unterschiedlich viel Speicherplatz benötigen und verschiedene Auflösungen und Farbsets anbieten.) Wenn Sie das einmal gemacht haben, müssen Sie Kommandos für hochauflösende Grafik – wie etwa DRAW oder MOVE – benutzen und können keine Zeichen mehr auf dem Bildschirm ausgeben. Auf diese Art und Weise könnten Sie ein Programm für hochauflösende Grafik im Textmodus eröffnen und Erläuterungen ausgeben, im hochauflösenden Modus das Spielbrett zeichnen und dann wieder in den Textmodus gehen, um das Ergebnis anzuzeigen.

Auf dem Commodore 64 mit doppelt soviel Speicher können Sie alle möglichen Grafikarten gleichzeitig benutzen. Ein fester Speicherbereich ist für sogenannte Punktblöcke reserviert. Sie können dem Computer mitteilen, was die Information im zugehörigen Speicherbereich bedeutet. Sie könnten eine Menge Bits zur Beschreibung der Farbe verwenden und nur ein paar für das Anzeigemuster; Sie könnten ein paar Bits verwenden, um Zeichen zu erzeugen; Sie könnten sich aber auch für eine begrenzte Farbauswahl entscheiden und die meisten Bits verwenden, um ein komplexes Punktmuster festzulegen. So hätten Sie etwa oben in der Mitte eine Überschrift, würden einen »Hintergrund« in allen 16 möglichen Farben bei bescheidener Auflösung und dann im Vordergrund eine Figur mit sehr hoher Genauigkeit zeichnen, aber nur 2 Farben in jedem Speicherblock anzeigen. Der Nachteil dieses Systems ist, daß der Computer keine speziellen Kommandos für jeden dieser Modi hat.

Auf dem BBC-Computer gibt es 7 verschiedene Betriebsarten. Sie können immer nur eine benutzen, die Sie vorher mittels des MODE-Kommandos ausgewählt haben müssen (außer man benutzt denjenigen Modus, der nach dem

Einschalten aktiv ist). Jeder Modus unterstützt Text und Grafik, aber mit unterschiedlichem Speicherbedarf, unterschiedlicher Auflösung und Farbauswahl. Nur in einem Modus gibt es grafische Zeichen, aber in allen anderen Modi sind selbstdefinierte Zeichen möglich.

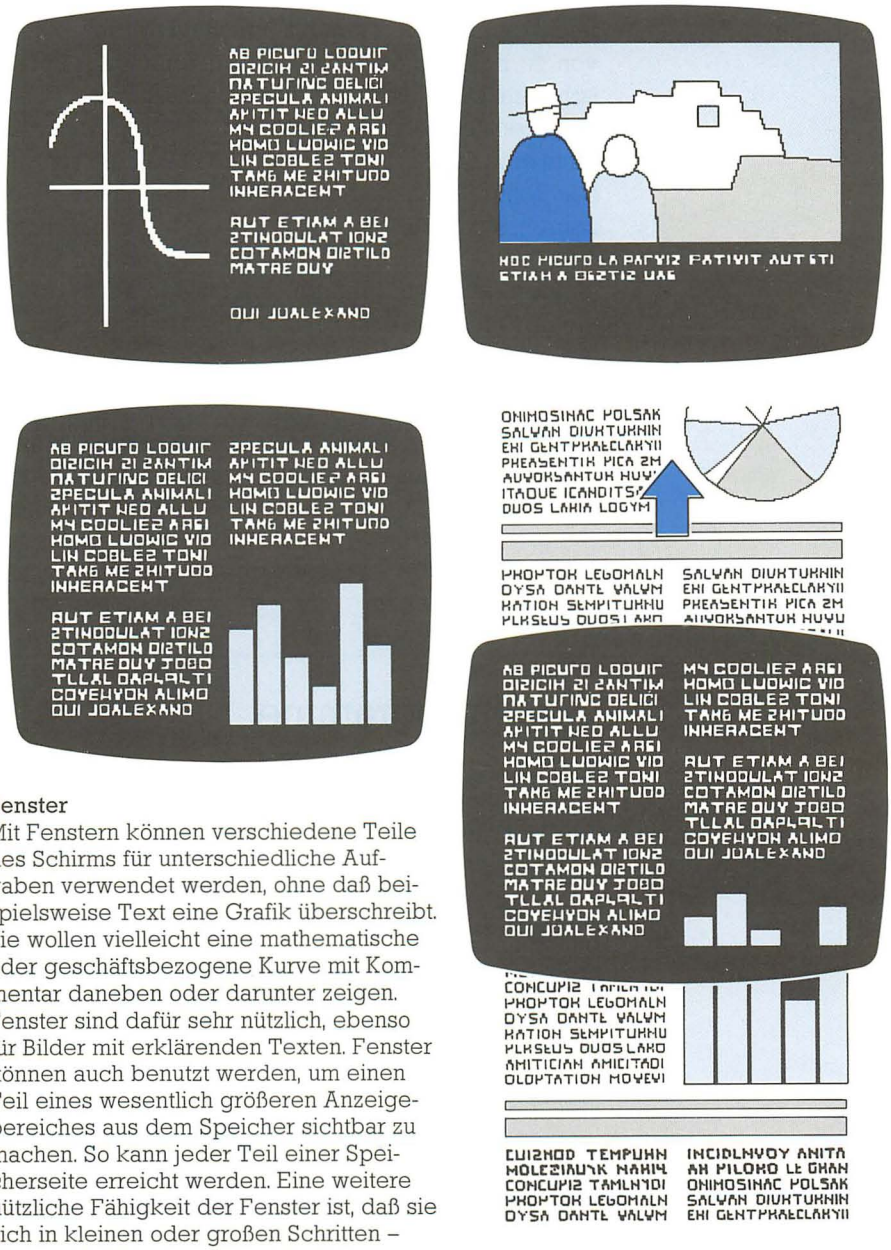
Der BBC-Computer hat noch eine Einrichtung, die ausgesprochen nützlich sein kann: das Konzept der »Fenster« oder »Windows«. Was ist ein Fenster? Nun, das Wort wird in 2 geringfügig verschiedenen Bedeutungen verwendet. Es ist am einfachsten, sich ein Fenster als einen Teil des Bildschirms vorzustellen, der für einen bestimmten Zweck reserviert wird, typischerweise für Grafik oder Text. Sie könnten etwa den Schirm in 2 Teile teilen und den Computer anweisen, die linke Seite für Text und die rechte für Bilder zu verwenden. Oder Sie könnten Illustrationen auf dem ganzen Schirm erzeugen – außer den 3 oder 4 untersten Zeilen, die Sie für erklärende Texte verwenden.

Der Vorteil definierter Fenster ist, daß der Computer automatisch Text und Daten trennt. Es werden keine umständlichen Anweisungen im Programm benötigt, um zu vermeiden, daß die Erläuterungen in die Grafik hineinlaufen und diese teilweise überdecken.

In einem etwas anderen Sinn könnten Sie sich den Bildschirm als Teil eines wesentlich größeren Schirmes vorstellen, der Teil des Computerspeichers ist. Denken Sie sich ein Quadrat mit 30 cm Seitenlänge, das Sie durch eine Schablone mit einer quadratischen Öffnung von 10 cm betrachten! Indem Sie die Schablone bewegen, können Sie das ganze Quadrat sehen, aber immer nur einen Teil auf einmal.

Sie sind eventuell schon an diese Arbeitsweise eines Computers in Verbindung mit Textverarbeitung gewöhnt (bei der nur ein Teil eines langen Textes gesehen werden kann) oder Tabellenkalkulation (bei der immer nur ein Teil einer großen Tabelle angezeigt wird). Der Schirm kann von einem Fenster zum anderen springen oder gleichmäßig in die gewünschte Richtung »scrollen« (abrollen).

Auf manchen Computern mit diesem Fensterkonzept können Sie eine Speicher-»Seite« mit anzuzeigenden Daten einrichten, die wesentlich größer als der Schirm ist. Sie öffnen dann ein oder mehrere Fenster auf diese Speicherseite unter Benutzung des ganzen oder eines Teils des Schirms, um sich Teile davon anzuschauen. (Sie könnten die Tabellenüberschrift mit einem Fenster im Blickfeld behalten, während Sie scrollen und ein anderes benutzen, mit dem Sie die letzten Zeilen anschauen.) Der Vorteil bei der Programmierung von Grafik ist, daß Sie eine »theoretische« hohe Auflösung verwenden können, um sich die ein-



Fenster

Mit Fenstern können verschiedene Teile des Schirms für unterschiedliche Aufgaben verwendet werden, ohne daß beispielsweise Text eine Grafik überschreibt. Sie wollen vielleicht eine mathematische oder geschäftsbezogene Kurve mit Kommentar daneben oder darunter zeigen. Fenster sind dafür sehr nützlich, ebenso für Bilder mit erklärenden Texten. Fenster können auch benutzt werden, um einen Teil eines wesentlich größeren Anzeigebereiches aus dem Speicher sichtbar zu machen. So kann jeder Teil einer Speicherseite erreicht werden. Eine weitere nützliche Fähigkeit der Fenster ist, daß sie sich in kleinen oder großen Schritten – ganz nach Wunsch – über die Seiten bewegen können. Bei Grafik kann so etwa eine Nahansicht der Zeichnungen des Schirms geliefert werden. Auf manchen Computern läßt sich ein Grafikfenster mit hoher Auflösung definieren, um das herum sich gewöhnlicher Text befindet.

zelnen Teile Ihres Designs nacheinander aus der Nähe anzuschauen.
Der Nachteil der Fenstertechnik ist, daß nicht jedermann leicht damit zurecht kommt. Um einen besseren Begriff davon zu bekommen, wollen wir uns noch ein Beispiel anschauen. Der NewBrain ist ein Computer, der ausschließlich Fenstertechnik verwendet. Sein »Anzeige-Editor« kann eine Seite mit insgesamt 255 Zeilen Text verarbeiten. Der Schirm kann davon maximal 30 Zeilen gleichzeitig anzeigen, sich aber in dieser Seite auf- und abwärts bewegen, um die restlichen Textteile zu zeigen. Auf dieser Seite können Sie einen Grafikteil reservieren, als würden Sie mit einem Stift einen Kasten auf ein Blatt Papier malen und nur hier Bilder malen. In diesem Grafikfenster (das ganz oder teilweise auf dem aktuellen Schirm zu sehen sein kann) verwenden Sie Grafikbefehle wie PLOT, TURN, DRAW und so weiter. Auf der restlichen Seite bearbeiten Sie Text mit normalen PRINT-Kommandos wie gehabt.

Grafikprogramme

Bis jetzt haben wir so getan, als ob der Computer die gesamte Information, die zum Abbilden einer Darstellung benötigt wird, im Speicher behält. Das macht er auch, wenn das Bild wirklich auf dem Schirm ist. Auf Grund der Wirkungsweise einer Kathodenstrahlröhre, die ja der Grundbaustein eines Monitors ist, muß der Computer das Bild aber auch viele Male pro Sekunde neu zeichnen, bevor es verblaßt.
Wenn ein Programm abläuft, ändert sich das angezeigte Bild aber auch noch viele Male. Der Computer schreibt verschiedene Meldungen für Sie, zeichnet Bilder über diejenigen, die sich bereits auf dem Schirm befinden, löscht auch manchmal den Schirm und fängt neu an. Behält der Computer dann diese ganzen verschiedenen Schirme voller Informationen im Speicher?
Nein, natürlich nicht. Aus diesem Grund können Sie einen Informationsaustausch mit dem Computer nicht »zurückblättern« und nachsehen, was er 100 Zeilen vorher einmal geantwortet hat. Statt dessen gibt das Programm selbst dem Computer Anweisungen, wie der Bildschirmspeicher geändert werden soll, und zwar genau dann, wenn es nötig ist. Der Computer hat nur eine »Seite« Bildschirmspeicher,

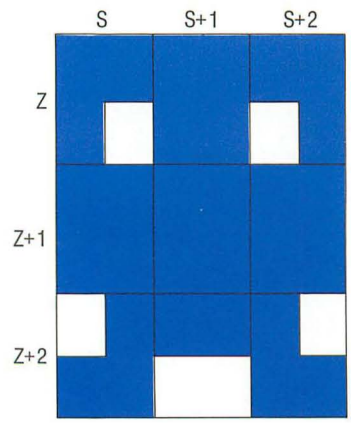
auf der er ständig »ausradiert« und »einträgt«, elektronisch natürlich, wie es das Programm ihm vorschreibt.
Auf diese Weise können Sie sich ein »dynamisches« Bild aufbauen. Sie können den Computer dabei beobachten, wie er eine Linie über den Schirm »zeichnet«. Sie können ihn eine Rakete zeichnen und diese auf dem Schirm aufsteigen lassen. Tatsächlich beobachten Sie den Computer dabei, wie er sein Anzeigeregister und das Bild, das sich daraus ergibt, kontinuierlich ändert.
Das ist sehr nützlich, denn es bedeutet, daß Sie mit wenig Speicherplatz bewegte Grafiken erzeugen können. Der Nachteil ist, daß sich das Bild nur langsam ändert. Die Geschwindigkeit mag ausreichen, wenn Sie nur eine einfache Figur mit ein paar BASIC-Befehlen zeichnen und sie dann ein Stück weiter auf dem Schirm neu erzeugen, aber nicht, wenn Sie mehrere Elemente eines komplizierten Bildes gleichzeitig ändern möchten. Es kann für den Computer einen Zeitaufwand von mehreren Sekunden, ja Minuten bedeuten, größere Teile seines Anzeigeregisters zu ändern.
Um dieses Problem zu umgehen und um sofort von einem Bild auf das nächste schalten zu können – wie in einem Film – werden mehrere Seiten für die Anzeige gleichzeitig im Speicher reserviert. Mit einfachen Kommandos können Sie dann den Computer zwischen den einzelnen Seiten umschalten lassen.
Viele Computer bieten diese Möglichkeit. Auf dem Tandy Color Computer mit 32 kB RAM können Sie bis zu 8 Seiten vorsehen. Die Voraussetzung ist dabei natürlich, daß Sie den Speicherbedarf jeder einzelnen Seite durch eine geringe Auflösung und eine Beschränkung der Menge der möglichen Farben begrenzen. Auf dem Commodore 64 können Sie zwischen 4 Seiten für allgemeine Anzeigeninformationen wählen, aber es gibt insgesamt nur einen Teil des Hauptfarbspeichers, der's erst interessant macht. Auf dem NewBrain kann man sich so viele Seiten definieren, wie es vom Speicher her möglich ist.
Natürlich benötigt jede Speicherseite eine Menge Platz. Selbst wenn der Speicher schon fast von Ihren wundervollen Grafikdarstellungen ausgefüllt ist, benötigen Sie immer noch Platz für Ihr Programm und Arbeitsspeicher für den Computer. Deshalb sind die mit kleinen Heimcomputern erreichbaren Animationseffekte begrenzt. Um bewegte Videos professioneller Qualität zu erzeugen, brauchen Sie eine entsprechend ausgereiftere Ausrüstung, die natürlich auch ihren professionell anmutenden Preis hat.

Sprites

Ein nützliches Werkzeug, das die Programmierung bewegter Figuren vereinfacht, sind die sogenannten »Sprites«. Wir wollen uns einmal die Bewegung von Figuren anschauen und die Art und Weise, wie Sprites benutzt werden. Wie werden normalerweise Figuren mit BASIC-Kommandos über den Bildschirm bewegt? Zuerst wird das Aussehen der Figur, die mehr als eine grafische Einheit umfassen kann (Zeichen, Pixel oder Punkt), auf dem Papier geplant. Wir wollen als Beispiel eine einfache »Space-Invader«-Figur mit 9 einfachen grafischen Elementen benutzen. (Die gleichen Prinzipien gelten, wenn Sie selbstdefinierte Zeichen verwenden, um eine größere Genauigkeit zu erreichen, oder wenn die Figur wesentlich größer wäre.) Weiterhin werden noch eine oder mehrere Variable benötigt, um die Lage der Details auf dem Schirm (Zeile und Spalte) festzuhalten. Wir wollen dafür Z und S – wie in der Figur unten – verwenden.
Danach wird ein Programmteil in BASIC oder einer anderen Programmiersprache (Unterprogramm, Prozedur oder was auch immer die von Ihnen verwendete Sprache unterstützt) erstellt, um den Computer die Figur zeichnen zu lassen. Unseres könnte so aussehen:

```
500  REM UNTERPROGRAMM ZUM ZEICHNEN DES INVADERS
510  PRINT@ (R, C), CHR$ (129) + CHR$ (128) + CHR$ (130)
520  PRINT@ (R+1,C), CHR$ (128) + CHR$ (128) + CHR$ (128)
530  PRINT@ (R + 2,C), CHR$ (136) + CHR$ (131) + CHR$ (132)
540  RETURN
```

Sprites



129	128	130
128	128	128
136	131	132

Wenn die Figur jetzt richtig bewegt werden sollte, könnte noch ein »freier Rand« programmiert werden, so daß sich die Figur vor jeder Bewegung selbst löscht. Wir werden diese Verbesserung jetzt außer acht lassen.
Zu guter Letzt wird ein Programmteil zum Ändern der Werte für Z und S, zum Löschen der Figur und Neuzeichnen an der gewünschten Stelle errichtet, etwa so:

```
10  REM INVADER BEWEGEN
20  FOR X = 1 TO 10
30  READ R, C
40  GOSUB 400: REM ALTES BILD LÖSCHEN
50  GOSUB 500: NEUES BILD ZEICHNEN
60  NEXT X
70  DATA 5, 10, 5, 11, 6, 11, 6, 12, 7, 12
80  DATA 7, 13, 7, 14, 8, 14, 8, 13, 8, 12
```

Nicht schwer zu programmieren, aber ganz schön umständlich! Jedesmal, wenn Sie die Figur bewegen möchten, müssen Sie den Computer den kompletten Befehlssatz zum Löschen und Zeichnen ausführen lassen. Das sind jedesmal rund 15 Programmzeilen, durch die sich der Computer hindurcharbeiten muß, wenn er die Figur neu zeichnet (einschließlich der hier nicht aufgeführten Löschroutine).
Durch die Benutzung eines Sprites kann dieser Aufwand beträchtlich reduziert werden. Ein Sprite ist eine recht große Figur, die aus mehreren einzeln definierbaren Punkten besteht, welche dann als Ganzes bewegt werden können. Der Computer hat ein paar spezialisierte Register im Speicher, in denen die Einzelheiten des Spritemusters aufbewahrt werden. Das funktioniert so ähnlich wie das Erstellen eines besonders großen benutzerdefinierten Zeichens. Sie geben das gewünschte Muster in Schwarzweiß (oder in Farbe, wenn der Computer es erlaubt) ein, und der Computer speichert es unter einem Codenamen, Sprite 2 beispielsweise. Sie sagen dann dem Computer, indem Sie wie oben Variable benutzen, an welcher Stelle er Sprite 2 drucken soll. Aber statt jedesmal ein Unterprogramm mit den ganzen Einzelheiten des Zeichens abarbeiten zu lassen, befehlen Sie jetzt dem Computer lediglich »Lösche Sprite 2« und dann »Zeichne Sprite 2« (natürlich mit den passenden Kommandos). Der Effekt ist, daß das nicht nur schneller zu programmieren ist, sondern vom Computer auch rascher ausgeführt werden kann.

Wir haben Sprites auf dem Commodore 64 benutzt, einem Computer, der nicht gerade gute Kommandos für die hohe Auflösung besitzt. Deshalb müssen die ganzen notwendigen Einzelheiten Stück für Stück mittels POKE-Kommandos in den Speicher geladen werden (wie das geht, ist in Kapitel 7 erklärt). Die Programme sind nicht einfach zu verstehen und auf andere Maschinen zu übertragen, so daß wir sie hier nicht aufführen. Um Ihnen einen Eindruck von der Leistungsfähigkeit des Verfahrens zu geben, hier einige Einzelheiten: Der Commodore kann die Daten von bis zu 8 Sprites verarbeiten, von denen jeder mit 21 × 24 Punkten definierbar ist. Er kann ihre Geschwindigkeit und Richtung beeinflussen und sie automatisch in ihrer Größe verdoppeln – horizontal, vertikal oder beides. Er ordnet ihnen Prioritäten zu, so daß ein Sprite vor einem anderen vorbeigleiten kann; ferner kann er Kollisionen zwischen Sprites feststellen, was für Spiele ausgesprochen nützlich ist.
Sprites sind auch auf dem Texas Instruments 99/4A (mit Extended BASIC) und den Atari Computern erhältlich, wo sie »Player missile graphics« heißen. Jede dieser Maschinen hat das leistungsfähige Konzept der Sprites auf eine andere Art und Weise verwirklicht.

Farbbehandlung

Auf Seite 33 gaben wir Ihnen eine Liste von 8 Farben und den zugehörigen Codes. Dies sind tatsächlich die 8 auf Heimcomputern am häufigsten benutzten Farben. Zumindest sind die Namen identisch, die zugehörigen Farbtöne können auf den einzelnen Computern und Bildschirmen völlig verschieden aussehen.
Wie Sie wahrscheinlich wissen, sind Rot, Grün und Blau 3 »Grundfarben« des Lichts. Alle anderen Farben können durch das Kombinieren aus Lichtstrahlen dieser 3 Typen hergestellt werden. Die angegebenen Farbcodes wurden so gewählt, daß die Codes dieser 3 Grundfarben durch einfaches Addieren andere Farben ergeben. Cyan (Hellblau) beispielsweise ist eine Kombination aus Blau und Grün. Der Code für Cyan (dezimal 6, binär 0110) ist die Summe der Werte für Grün (dezimal 2, binär 0010) und Blau (dezimal 4, binär 0100). Dies ist das übliche Codesystem, obwohl die Kommandos, um Vorder- oder Hintergrundfarben zu ändern, von Gerät zu Gerät stark variieren. Die meisten Heimcomputer haben mindestens 8 Farben,

obwohl sie nicht immer Schwarz und Weiß enthalten und nicht immer alle Farben in allen Betriebsarten verwendet werden können. Andere häufig verwendete Farben sind Beige und Orange. Andere Computer bieten 9 Farben, indem sie Schwarzweiß zusätzlich zu den 8 »Hauptfarben« bieten. Andere tun so, als ob sie 16 Farben hätten. Aber seien Sie gewarnt! Obwohl beispielsweise der Commodore 64 wirklich 16 Farben verwendet, gibt es auf den BBC-Microcomputern nur 8 verschiedene. Die anderen »Farben«, die dann die angegebene Gesamtzahl ausmachen, sind blinkende Versionen des Grundsets.
Auf vielen Computern ist es möglich, den Eindruck anderer Farben hervorzurufen, indem schachbrettartig abwechselnd verschiedene Farben gezeichnet werden. Der Wechsel von Gelb und Rot erzeugt aus einer gewissen Entfernung den Eindruck von Orange. Kompliziertere Techniken können diesen Bereich noch weiter ausdehnen. Die Atari-Computer bieten beispielsweise 16 verschiedene Helligkeiten für jede ihrer 16 Farben. Viele für den Apple II erhältlichen Spiele verblüffen durch vielfältige Farbschattierungen, die aus den Grundtönen erzeugt werden. Um so etwas jedoch selbst zu machen, müssen Sie ein erfahrener Programmierer sein und/oder ein spezielles »Zeichenprogramm« haben, das Ihnen bei der Erzielung solcher Effekte hilft.

BASIC-Grafikkommandos auf einem Mikrocomputer

Wir können hier nicht einmal sämtliche Grafikbefehle beschreiben, die es allein in BASIC gibt, geschweige denn von anderen grafikorientierten Sprachen wie LOGO. Um Ihnen einen Eindruck zu vermitteln, was Ihnen eine einigermaßen gute BASIC-Version an Unterstützung für Ihre Grafiken bietet, wollen wir uns die wichtigsten Kommandos des Microsoft-BASIC auf dem Dragon 32 anschauen. Wie wir bereits in der Einleitung erwähnt haben, werden die gleichen oder ähnliche Kommandos auf einer Vielzahl anderer Maschinen verwendet. Die Programme, die Sie weiter hinten im Buch finden, sind alle in Microsoft-BASIC geschrieben, so daß dieser Teil auch als Erklärung für unbekannte Kommandos dient, die Sie dort finden mögen.
Bei niedriger Auflösung benutzt dieses BASIC den Befehl

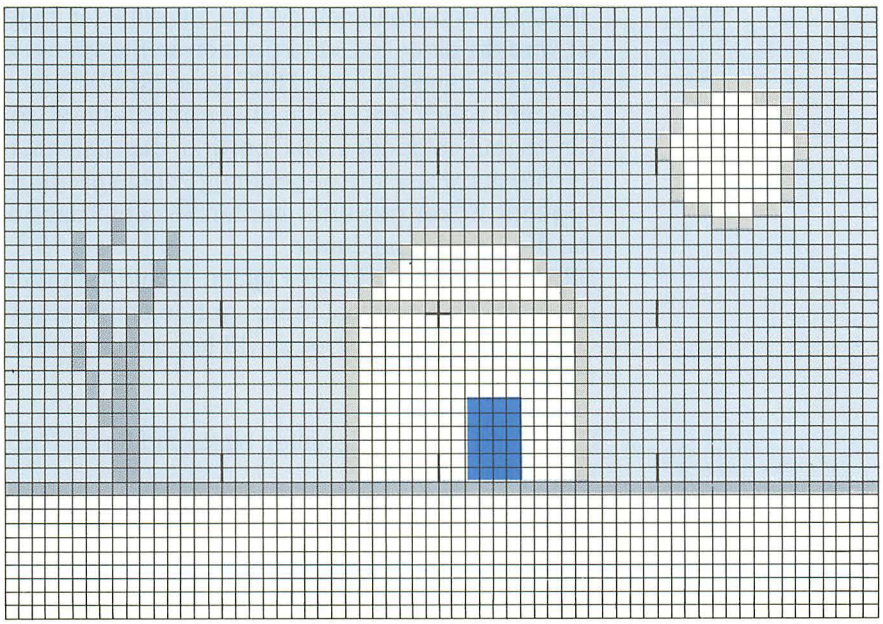
PRINT@ als Haupt-»Positionierungsmittel«. Der nutzbare Schirm hat 16 Zeilen mit 32 Zeichen, die von 0 aus hochgezählt werden. Mit einer einzigen Zahl kann deshalb eine Schirmposition beschrieben werden, wobei der erlaubte Bereich von 0 bis 511 reicht.
In niedriger Auflösung gibt es 9 Farben. In höherer Auflösung können nur begrenzte Kombinationen davon verwendet werden. Es ist ein ziemlich standardmäßiger Satz mit den folgenden Codenummern:

Schwarz	0	Grün	1
Gelb	2	Blau	3
Rot	4	Beige	5
Cyan	6	Magenta	7
Orange	8		

(Kein Weiß, wie man sieht.) Hintergrundfarben werden mit CLS ausgewählt. So färbt CLS 6 etwa den Schirm cyanfarbig.
Eine nervtötende Eigenschaft des Dragon-BASIC ist, daß es normalerweise Text nur in Schwarz auf grünem Hintergrund darstellen kann. Unsere Programme versuchen zu zeigen, wie das als Positivum ausgenutzt werden kann (siehe beispielsweise die Anzeige für »Würfeln« auf Seite 113). Es gibt 16 grafische Zeichen, die nur durch das CHR$-Kommando angesprochen werden können. Die Vordergrundfarbe ist immer Schwarz, die Hintergrundfarbe kann durch das Ändern der Codezahl beeinflußt werden. CHR$(140) produziert eine schwarzgrüne Figur, CHR$(204) dagegen die gleiche Figur, aber mit beigem Hintergrund. Es können keine zusätzlichen Figuren definiert werden.
Wie wir bereits früher erwähnten, können niedrige und mittlere Auflösung zusammen verwendet werden. Die mittlere Auflösung beträgt 64*32 Punkte. Zur Lokalisierung dieser Punkte oder Pixel sind diesmal 2 Zahlen notwendig (zuerst Spalte, dann Reihe). Das Kommando SET (gefolgt von Spalte, Reihe, Farbzahl) wird benutzt, um Pixel einzuschalten. RESET schaltet sie wieder aus. POINT wird benutzt, um festzustellen, ob bestimmte Pixel an oder aus sind. Sie werden das in einigen unserer Spiele finden.
Hochauflösende Grafik muß vollkommen getrennt von niedriger und mittlerer Auflösung benutzt werden. Sie finden sie beispielsweise in dem Programm »Künstler« auf Seite 119 ff. Es gibt 5 verschiedene Betriebsarten:

0	128 96	2 Farben aus 2 Sets
1	128 96	4 Farben aus 2 Sets (benötigt mehr Speicherplatz)
2	192 128	2 Farben
3	192 128	4 Farben
4	256 192	2 Farben

Die Zwei-Farb-Sets sind Schwarz/Grün (0) und Schwarz/Beige (1). Die Vier-Farb-Sets sind Beige/Cyan/Magenta/Orange (1) und Grün/Gelb/Blau/Rot (0). Es gibt keine Möglichkeit, die Farben in den Sets zu ändern.



Hochauflösende Grafik

Wir wollen typische Kommandos für hochauflösende Grafik anhand eines einfachen Bildes verdeutlichen.
Zuerst wählen wir einen passenden Vierfarbmodus und löschen den Schirm:

```
10 PMODE 1,1:SCREEN 1,0
20 PCLS
```

Wir haben in diesem Bild blauen Hintergrund und grünen Vordergrund:

```
30 COLOR 1,3
```

Jetzt wollen wir mit dem LINE-Kommando die Erde malen und mit PAINT einfärben:

```
40 LINE (0,9)-(63,9), PSET
50 PAINT (0,0), 1, 1
```

Um den Baum zu zeichnen:

```
 60 LINE (8,19)-(9,10), PSET, BF
 70 LINE (5,19)-(7,14), PSET
 80 LINE (5,27)-(7,19), PSET
 90 LINE (8,28)-(7,25) PSET
100 LINE (12,27)-(9,20), PSET
```

(Beachten Sie, wie der Computer gerade Linien »annähert«!)
Um die Tür zu zeichnen:

```
110 LINE (34,15)-(37,10), PSET, BF
```

Mit dem CIRCLE-Kommando zeichnen wir eine gelbe Sonne und füllen sie dann aus:

```
120 CIRCLE (53,33), 5,2:PAINT (53,33), 2,2
```

Zum Schluß erstellen wir mit relativen Zeichenbefehlen die Umrisse des Hauses:

```
130 DRAW C4; BM25, 10; U12; E5, R7; G5;
NL16; D12
```

(Beachten Sie im DRAW-Befehl die Codes für »BLANK MOVE« und »NO UPDATE«!)

Diverse »Dienst«-Kommandos schalten die einzelnen Betriebsarten ein:
PMODE: Gefolgt von einer Zahl für die Betriebsart und einer anderen für die zu benutzende Grafikspeicherseite.
SCREEN: Gefolgt von einer Zahl für den Typ (0 für Text und 1 für hochauflösende Grafik).
COLOR: Gefolgt von den Codes für Vorder- und Hintergrundfarben.
PCLS, PSET, PRESET, PPOINT: Die Äquivalente von CLS, SET, RESET, POINT für die hohe Auflösung.
LINE: Zeichnet eine Linie von einem Koordinatenpaar zu einem anderen. Es werden also 2 Koordinaten der hochauflösenden Grafikschirme benutzt, um die Endpunkte der Linie zu bezeichnen. (Wenn nur ein Paar angegeben ist, wird die aktuelle Cursorposition als Startpunkt verwendet.) LINE kann auch von PSET oder PRESET gefolgt werden, um in der Vorder- oder Hintergrundfarbe zu zeichnen. Ein darauf folgendes »B« bewirkt das Zeichnen eines Rechtecks, wobei die gegebenen Punkte als gegenüberliegende Ecken gewertet werden. »BF« füllt das Rechteck mit der Vordergrundfarbe.
DRAW: Zeichnet »relative« Linien. In anderen Worten: Es ist nicht nötig, Koordinaten anzugeben, sondern nur Anweisungen wie »10 hoch« oder »15 herunter«. Dies sind die möglichen Codes:

U, D, L und R: Hoch, Herunter, Links und Rechts (up, down, left, right).
E, F, G und N: Im Winkel von 45, 135, 225 und 315° zur Senkrechten.
C: Farbänderung.
A: Um gegebenen Winkel verschieben (1, 2, 3 oder 4 = 0,90, 180, 270°).
S: Ändern des Maßstabs. 1 = Viertelung, 4 = Normal, 62 = Maximum.
N: Startposition unverändert lassen.
M: Zu den angegebenen Koordinaten oder um angegebenen Betrag (+ oder −) bewegen.
BM: Verändern des Startpunktes, ohne zu zeichnen.
PAINT: Beginnend mit den gegebenen Koordinaten wird der Schirm in einer gegebenen Farbe bemalt, bis die Grenze einer anderen Farbe erreicht ist. Ein Beispiel: PAINT (10,10), 4,6 füllt den Schirm mit Rot vom Punkt 10,10 aus, bis die Grenze einer cyanfarbenen Fläche erreicht wird.
CIRCLE: Die Anweisung wird gefolgt von Mittelpunktskoordinaten, Radius, Farbe und eventuell anderen Codes, um Kreisbögen oder Ellipsen zu bezeichnen.

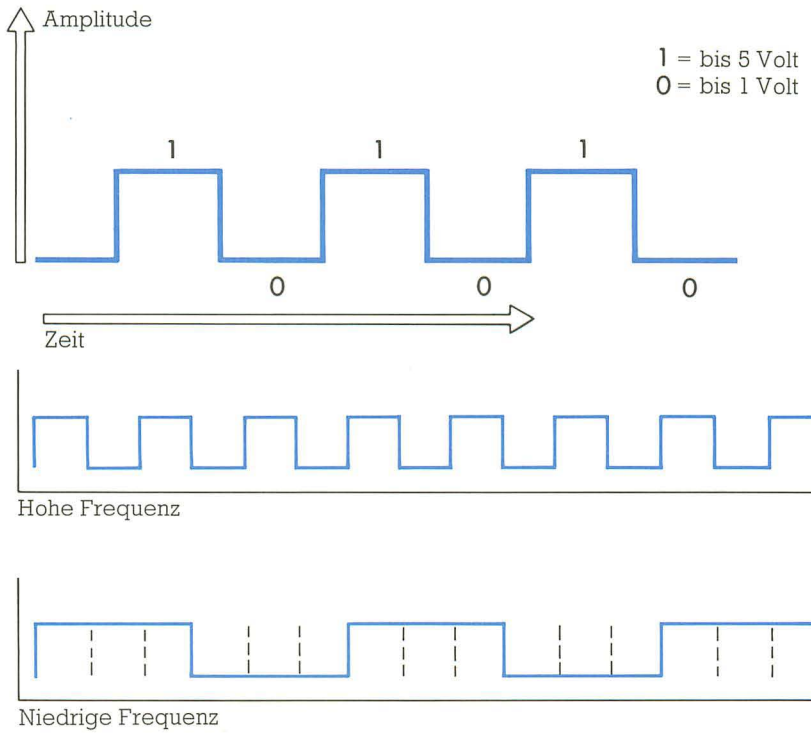
GET und PUT: Dieses Kommandopaar imitiert in gewisser Weise Fähigkeiten der Sprites. Mit GET können Sie einen rechteckigen Bereich des Schirmes (beliebige Größe mit von Ihnen angegebener Lage) in einer selbstdefinierten Matrix speichern. Mit PUT können Sie ihn dann an einer beliebigen Stelle des Schirms neu zeichnen. Dies stellt eine praktische Möglichkeit dar, um viele identische Muster in einer festen oder zufälligen Anordnung zu zeichnen.
Das Dragon-BASIC ist nicht das flexibelste BASIC, das erhältlich ist, aber es ist leicht zu benutzen und kann leicht eine Vielzahl verschiedener Bilder erzeugen, wie Sie noch aus unseren Programmen erkennen werden.

Töne aus Ihrem Computer

Für Spieler sind Töne besonders interessant, denn schon ein paar ganz einfache Toneffekte können ein Computerspiel so richtig zum Leben erwecken. Interessanterweise wurde in Spielhallen festgestellt, daß ein Gerät nicht mehr angefaßt wird, wenn sein Ton ausgefallen ist, selbst wenn der Rest der Schaltung absolut korrekt arbeitet. Es scheint, als würden wir es mögen, wenn uns das Spielgeschehen über Töne vermittelt wird.
Das trifft natürlich nicht nur für Spiele zu. Viele Computer benutzen ihre Tonmöglichkeiten, um einen einfachen »Piepser« zu erzeugen, wenn eine Taste gedrückt wird. Wir werden sehen, daß jeder Heimcomputer mit der Fähigkeit zur Tonerzeugung ganz leicht einfache Töne erzeugen kann. Einige Computer haben wesentlich weiterreichende Fähigkeiten und können wie ein spezieller Musiksynthesizer programmiert werden. Für solche Programme benötigen Sie jedoch musikalisches Fachwissen wie für jedes Musikinstrument. In diesem Kapitel konzentrieren wir uns auf einfache Effekte, die man mit minimalem musikalischen Wissen erreichen kann.

Wie Computer Töne erzeugen

Genauso wie er Zahlen, Buchstaben oder grafische Zeichen darstellen kann, kann der Binärcode, mit dem der Computer arbeitet, den Verlauf von Tonfrequenzen repräsentieren. Für den Computer ist dieser Code natürlich ein elektronisches Signal, ein digitales Signal. Elektronisch werden die binären Elemente durch 2 verschiedene Spannungen repräsentiert (normalerweise etwa 5 Volt für binär 1 und 0 Volt für binär 0). Um den Verlauf der Signale, die auch Pulse genannt werden, grafisch zu zeigen, benutzen wir sogenannte Rechteckwellen, die an jedem Punkt nur einen von 2 Zuständen annehmen können. Diese 2 Werte sind der untere und obere Bereich der Kurve. Der Computer kann Folgen von Bits (binären Elementen) an den Lautsprecherschaltkreis senden. Abhängig von dem Zeitintervall, in dem das Signal von 1 nach 0 und umgekehrt wechselt, ändert sich die Frequenz der Töne.



Rechteckwellen
Die Rechteckwelle repräsentiert die Spannungswechsel im digitalen Signal des Computers im Verlauf der Zeit. Der Wechsel zwischen den beiden Zuständen (von 1 nach 0 oder umgekehrt) geschieht theoretisch augenblicklich – dies deshalb, weil das elektronische Signal des Computers nur einen der beiden Zustände erlaubt »mit nichts dazwischen« (das gilt auch für den Binärcode). Je höher die Zahl der Wechsel (Pulse) in einem gegebenen Zeitraum, desto höher die Frequenz der Welle.

Betrachten Sie einmal die Rechteckwellen in unserer Illustration! Im oberen Diagramm wechselt das Signal sehr schnell zwischen 0 und 1 hin und her. Die Rechteckwelle hat deshalb eine sehr hohe Frequenz. Im unteren Diagramm hat die Rechteckkurve eine geringere Frequenz, weil das Signal viel seltener wechselt. Die vom Computer erzeugten elektrischen Pulse werden von einem Lautsprecher in Töne übersetzt, entweder von einer hohen Frequenz (eine hohe Note), wenn das originale Digitalsignal sehr schnell wechselt, oder von einer niedrigeren Frequenz, wenn die Pulse länger waren (was eine tiefe Note ergibt).
Sehr einfache Tongeneratoren in Computern arbeiten auf diese Weise. Sie senden elektrische Pulse an den ein-

gebauten Lautsprecher oder an einen angeschlossenen externen Lautsprecher (häufig derjenige des Fernsehgeräts). Wir hören sie dann als einfache Töne oder Piepser. Da sie nicht allzu melodisch sind, ist »Piepser« wohl der bessere Ausdruck.
Wir wollen uns einmal kurz den Spectrum/Timex 2000 anschauen, um zu sehen, wie man eine solche einfache Tonfähigkeit nutzt. Das BEEP-Kommando bewirkt einen Ton mit vom Programmierer wählbarer Dauer und Tonhöhe. Die Dauer wird in Sekunden angegeben, wobei man natürlich auch Bruchteile davon wählen kann. Die Tonhöhe wird als Zahl angegeben. Das mittlere C ist 0, und die Halbtöne werden in Einerschritten hoch- oder heruntergezählt. So repräsentieren 12 ganze Zahlen alle Noten in einer Oktave. Wenn Sie die Tonhöhe justieren möchten, müssen Sie sie entsprechend genauer angeben. Ein Befehl wie

```
BEEP 1, 2
```

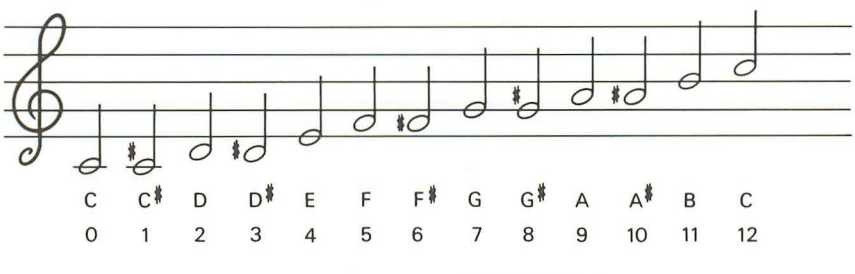
würde mit diesen 2 Parametern den Computer das D über dem mittleren C (C = 0, C♯ = 1, D = 2) 1 Sekunde lang spielen lassen; und

```
BEEP .75, 0.01
```

würde für eine ¾ Sekunde einen Ton ganz leicht über dem mittleren C erzeugen.
Durch eine Folge von BEEP-Kommandos mit entsprechend eingefügten PAUSEn kann man den Computer eine einfache Melodie spielen lassen. Man müßte schon ein ganz schön verrückter Musiker sein, um den Spectrum zur ernsthaften Musikerzeugung einzusetzen, aber die erzeugten Töne sind ganz gut geeignet, um Spiele zu beleben. Hier nun mögliche Anwendungsbereiche der Tonerzeugung:

- Ein hoher Ton, wenn Sie ein Ziel treffen, und vielleicht ein tiefer, wenn Sie es verpassen.

Musikalische Kennzeichnung



- Ein Piepser sagt Ihnen, wenn der Computer seinen Schachzug berechnet hat (da das ein paar Minuten dauern kann, brauchen Sie vielleicht eine Erinnerung, um an den Schirm zurückzukommen).
- Vielleicht auch eine kurze Melodie, um eine neue Spielphase einzuleiten, oder als Belohnung für den Gewinner.

Etwas sollte bei einfachen Kommandos wie diesem nicht vergessen werden: Manche Computer (aber nicht alle) bleiben an dem BEEP-Kommando hängen, bis der Ton beendet ist. So ist ein Piepston von einer Sekunde Dauer gleichbedeutend mit einer Sekunde Verzögerung im Programmablauf. Das kann manchmal ganz nützlich sein; manchmal wird es aber auch zum Problem, und Sie müssen unter Umständen die Toneffekte in Ihren Spielen begrenzen, um den Ablauf nicht allzusehr zu verlangsamen.
Musikprogramme mit dem BEEP-Kommando zu schreiben, ist eine enorm langwierige Angelegenheit, da für jeden Ton ein separates BEEP und für jede Pause dazwischen ein PAUSE-Kommando programmiert werden muß. Ein Beispiel eines etwas komfortableren Kommandos ist PLAY auf dem Dragon und den Tandy-Maschinen. PLAY kann von einem ganzen Satz Daten gefolgt werden und nicht nur von 2 Parametern. Es können Variable angegeben werden und sogar ganze Melodien in einem Datensatz. Es können Pauseanweisungen eingefügt werden, ohne daß dafür separate Kommandos verwendet werden müssen. Die Lautstärke kann mit einer Zahl von 1 bis 31 geregelt werden.
Einige einfache Tonkommandos erzeugen Töne, die nicht völlig mit den richtigen Noten übereinstimmen, da der Computer nie so ganz genau »gestimmt« ist. Geeignetere Kommandos wie PLAY sind genau abgestimmt; um die Sache weiter zu verbessern, erlauben sie es, anstelle von Zahlen für die Tonhöhe direkt die Buchstaben A bis G einzugeben, dazu » ♯ « und » – «-Symbole, um die erhöhten oder erniedrigten Töne zu verdeutlichen. Ein anderer Parameter zeigt dem Computer die gewünschte Oktave an (es stehen 5 zur Auswahl). Hier ist ein typischer PLAY-Befehl:

```
PLAY O1L4GGP4L3FF
```

Das bedeutet für den Computer, die Noten in Oktave 1 (O1) zu spielen, zuerst mit der Länge 4 (die genaue Dauer hängt vom gewählten Tempo ab, das man auch ändern kann) und später mit der Länge 3. 2 G's werden gespielt, gefolgt von einer Pause und dann 2 F's.

Auf Seite 98 ff. können Sie das PLAY-Kommando des Dragon in unserem »Orgelspieler«-Programm arbeiten sehen. Dieses einfache Programm erlaubt es, Melodien in »Echtzeit« zu spielen, wenn Sie die Tastatur wie eine Klaviertastatur benutzen. Wenn Sie einen anderen Computer haben, können Sie das Programm vielleicht anpassen oder ein ähnliches in einem Buch oder einer Zeitschrift finden.

Erzeugung komplexerer Töne

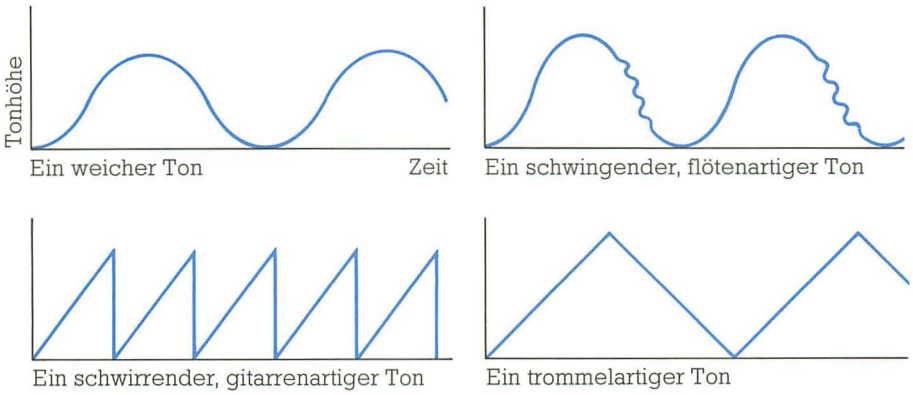
Der von einer normalen Rechteckwelle erzeugte Ton ist zwar ganz angenehm anzuhören, aber mutet trotzdem recht begrenzt an. Traditionelle Musikinstrumente erzeugen Klänge aus wesentlich komplizierteren Wellenmustern und Geräusche, die während ihres Andauerns in der Lautstärke und manchmal auch der Tonhöhe schwanken können. Ein Computer, der das nachmachen möchte, muß den »Verlauf« der Töne steuern können und idealerweise auch verschiedene Wellenformen erzeugen können.

Die ENVELOPE-Funktion erledigt die erste dieser Aufgaben. Sie benötigt eine Reihe von Parametern (14 bei der BBC-Maschine), die den Ton genau beschreiben, indem er in 4 Phasen eingeteilt wird: ATTACK, DECAY, SUSTAIN und RELEASE (Anstieg, Abfall, Halten und Freigeben). Durch Ändern der Werte für Lautstärke und Tonhöhe in diesen Phasen ist es möglich, eine annehmbare Imitation beispielsweise einer Flöte oder einer Trommel zu erreichen, genauso aber auch Klangeffekte wie Nebelhörner, Donnergrollen oder Rumpelgeräusche.

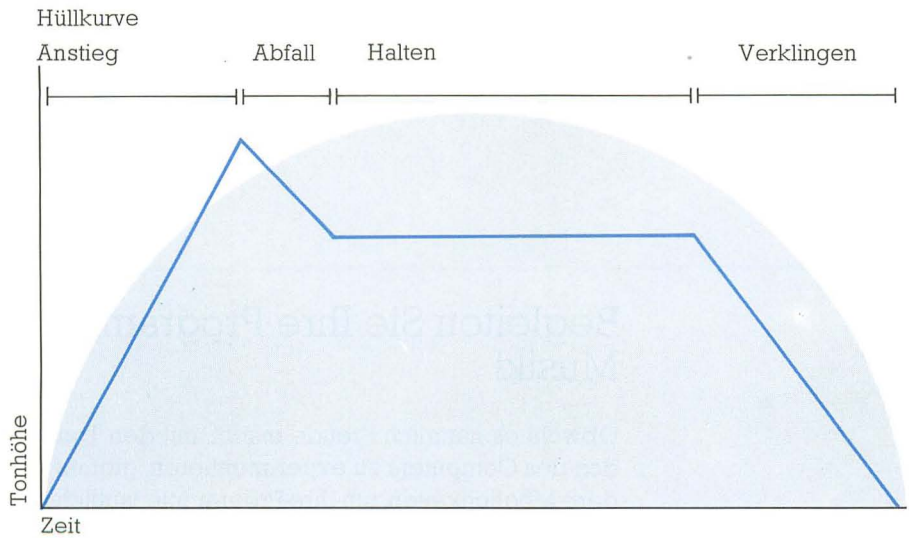
Die Benutzung verschiedener Wellenformen mit Beeinflußbarkeit des Tonkurvenverlaufs ist auf dem Commodore 64 möglich. Diese Maschine kann 3 verschiedene Wellenformen erzeugen: die bekannte Rechteckkurve, eine Sägezahnkurve und eine Dreieckskurve. Damit erzeugte Töne können dann über den ADSR-Verlauf geformt werden.

Computer mit der ENVELOPE-Funktion haben für gewöhnlich mehrere verschiedene Tonkanäle. Sie können so programmiert werden, daß verschiedene Töne gleichzeitig gespielt werden. Damit kann der Computer eine Melodie mehrstimmig spielen – ein stärkerer Effekt, als wenn Sie immer nur eine Note gleichzeitig spielen würden. Der BBC-Computer und der Commmodore 64 bieten beide 3

Wellenformen und Hüllkurven



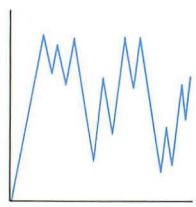
Die Töne verschiedener Musikinstrumente stellen unterschiedliche Wellenformen dar, wie oben gezeigt. Dies ist einer der Gründe, weshalb etwa eine Flöte und eine Gitarre nicht gleich klingen, selbst wenn sie denselben Ton spielen. Wenn die Saite eines Instruments gezupft wird oder eine Taste angeschlagen wird, ändert sich die Lautstärke des erzeugten Tons mehrfach. Dieses Muster, normalerweise als Hüllkurve bekannt, wird in 4 Teile geteilt: Anstieg, Abfall, Halten und Verklingen. Computer mit musikalischen Fähigkeiten können die Hüllkurven verschiedener Instrumente annähern. Ein typisches Beispiel – die Computerhüllkurve eines Pianos – wird unten gezeigt. Der farbige Bereich zeigt den Anstieg und Abfall der Tonhöhe. Mit der ENVELOPE-Funktion des Computers und anderen Kommandos zur Beeinflussung von Lautstärke und Tonhöhe in den 4 Phasen ist es oft möglich, verschiedene Musikinstrumente nachzuahmen und eine Vielzahl anderer Klangeffekte zu erzeugen.



verschiedene Musikkanäle. Natürlich muß der Computer nach dem ersten SOUND-Kommando (das in Verbindung mit ENVELOPE zur Tonerzeugung benutzt wird) weiterarbeiten, um etwa zu sehen, ob noch mehr zu spielen ist. Das heißt, er hört nicht auf zu arbeiten, wenn er einen Ton spielt. Besondere Schaltkreise führen den Ton fort, während der Computer in seinem Programm weiterarbeitet.

»Weißes« Rauschen

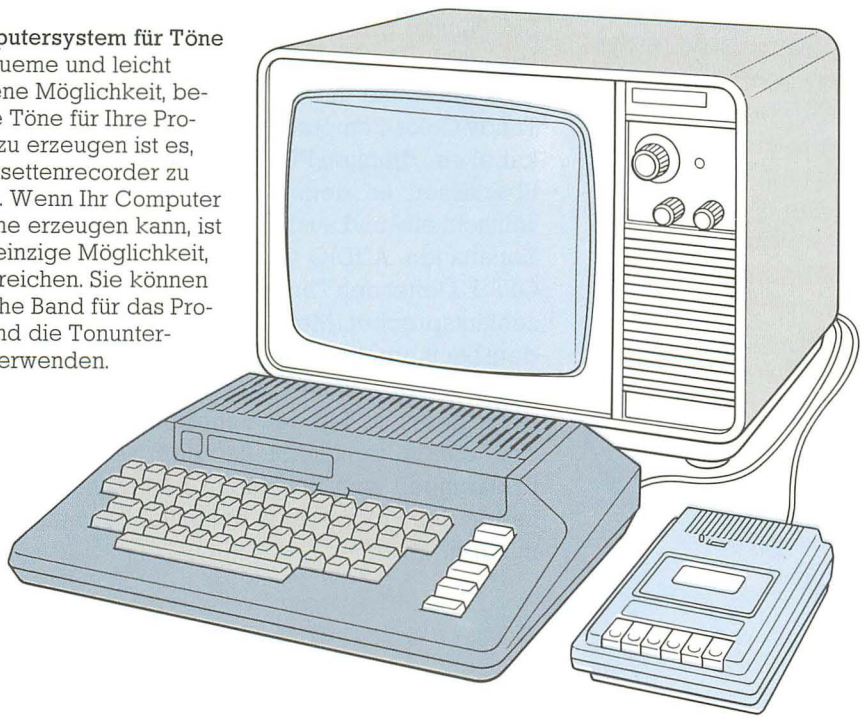
Wir müssen noch ein spezielles »Geräusch« als Gegensatz zu den Tönen erwähnen: »weißes« Rauschen. Das ist dieses nichtssagende Geräusch, das beispielsweise in einem Radio zu hören ist, das auf keinen speziellen Sender eingestellt ist. Es hat eine unregelmäßige Wellenform.
Der Computer kann weißes Rauschen genauso erzeugen wie Zufallszahlen, indem er mit seinen komplexen Schaltkreisen eine zufällige Bitfolge erzeugt. Viele Computer können das genauso gut wie (oder anstatt) mehr oder weniger musikalische Töne zu erzeugen. Der BBC-Mikrocomputer hat beispielsweise 3 Tonkanäle und einen Rauschkanal. Dieser kann 8 verschiedene Arten von Rauschen erzeugen (mit verschiedenen Wellenformen, Frequenzen, abhängig oder unabhängig von der Frequenz anderer Tonkanäle).
Weißes Rauschen ist in Spielen besonders nützlich. Es wird benutzt, um Zusammenstöße, Knallen oder Explosionen zu erzeugen. Wenn Ihr Computer diese Fähigkeit besitzt, werden Sie viele Beispiele dafür in Weltraumspielen finden.



»Weißes« Rauschen
»Weißes« Rauschen erzeugt das Zischen im Lautsprecher und den Schnee im Fernsehen. Die charakteristische Wellenform wird hier gezeigt.

Begleiten Sie Ihre Programme mit Musik!

Obwohl es natürlich Freude macht, mit den Tonkommandos des Computers zu experimentieren, gibt es noch andere Möglichkeiten, um Ihre Programme lautlich zu untermalen. Vergessen Sie nicht Ihren treuen Kassettenrecorder! Sie können seine Fähigkeiten benutzen, um Ihre Pro-



Ein Computersystem für Töne
Eine bequeme und leicht übersehene Möglichkeit, begleitende Töne für Ihre Programme zu erzeugen ist es, Ihren Kassettenrecorder zu benutzen. Wenn Ihr Computer keine Töne erzeugen kann, ist dies die einzige Möglichkeit, das zu erreichen. Sie können das gleiche Band für das Programm und die Tonuntermalung verwenden.

gramme mit Ton oder Geräusch zu beleben, selbst wenn Ihr Computer gar keine eingebauten Tonerzeugungsmöglichkeiten hat. Sie können ganz normale Aufnahmen zusammen mit den Computerprogrammen verwenden – selbst auf dem gleichen Band wie das Programm, wenn Sie wollen (natürlich an einer anderen Stelle des Bandes). Sie können dann das Band abspielen, um Hintergrundmusik oder Kommentare zu den Bildschirmanzeigen zu erzeugen.

Das ist besonders gut möglich, wenn Ihr Computer eine Fernbedienung hat, mit der er selbst den Kassettenrecorder ein- und ausschalten kann. Sie können Kommandos für MOTOR ON oder MOTOR OFF (die genauen Befehle sind von System zu System natürlich verschieden) in sein Programm einbauen und Ton und Bild synchronisieren. Sie können sogar das Tempo steuern, indem Sie INPUT- oder INKEY$-Funktionen einbauen und so den Motor nur einschalten, wenn der Benutzer eine Taste drückt. Hier ist ein Beispiel:

```
50  A$ = INKEY$
60  IF A$ = "Y" THEN MOTOR ON ELSE GOTO 50
70  (nächster Programmteil mit Tonuntermalung)
```

Um den Ton sauber synchronisiert zu bekommen, sollten Sie Pausen zwischen den Kommentaren auf dem Band lassen!
Eine Maschine mit der beschriebenen Fähigkeit ist der Tandy Color Computer. Sie stöpseln das Fernbedienungskabel ein, drücken PLAY auf dem Kassettenrecorder und überlassen es dem Computer, den Recorder wie erwünscht ein- und auszuschalten. Es gibt noch ein weiteres Kommando: AUDIO ON/OFF zusätzlich zu MOTOR ON/OFF. Es leitet den Ton vom Kassettenrecorder in den Fernsehlautsprecher. Meistens hören Sie nämlich nichts aus dem Lautsprecher des Kassettenrecorders, wenn das Verbindungskabel zum Computer in die Kopfhörerbuchse eingesteckt ist. Mit der Benutzung des AUDIO-Kommandos können Sie sicherstellen, daß der Benutzer Musik oder Erklärungen zum Programm hört, ohne sich vorher das Laden des Programmes bei voller Lautstärke anhören zu müssen.

Sprachsynthese

Mit Kontrollkommandos für den Kassettenrecorder (falls Ihr Computer solche hat) können Sie mit Ihrer eigenen Stimme oder irgend etwas anderem auf Band Ihre Programme begleiten lassen. Aber wie wäre es, wenn Ihr Computer zu Ihnen sprechen würde?
Sprachsynthese ist bereits verbreitet, und es gibt mehrere Systeme, die Heimcomputer zum Sprechen bringen. Die effektiveren benutzen spezielle Hardware, einen sogenannten Sprachsynthesizerchip. Die Schaltung darin enthält die Daten, die der Computer benötigt, um die Wellenformen zu produzieren, die für ein kleines Vokabular notwendig sind. Wenn Sie den Computer anweisen, Worte »auszusprechen«, werden die betreffenden Wellenformen generiert, und etwas Sprachähnliches ist zu hören.
Sprachsynthesizerchips finden immer größere Verbreitung. Der Texas Instruments TI99/4A hat ein solches Peripheriegerät für Sprachgenerierung, und auch für die Sinclair/Timex Computer sind verschiedene erhältlich. Wir haben einen Torch Computer mit eingebautem Sprachchip und erhalten so beispielsweise beim Schlangenspiel gesprochene Kommentare. Jedesmal wenn die Schlange eine Zahl frißt, nennt uns eine Stimme unsere Punktzahl.

Dasselbe geschieht, wenn wir einen Fehler machen und das Spiel verlieren.
Es ist möglich, einen ähnlichen Effekt mit reiner Software zu erzielen. Hier wird die normale Arbeitskapazität des Computers dazu verwendet, die Sprachkurven zu erzeugen. Es gibt beispielsweise ein Programm auf Kassette für den Sinclair Spectrum/Timex 2000, mit dem man wahlfreie Sätze laut aussprechen kann, die der Computer aufnimmt, codiert und reproduzieren kann. Auf diese Weise können Sie Ihr eigenes Vokabular wählen und sind nicht auf das begrenzte Vokabular der Sprachchips angewiesen.
Eines sollten Sie sich bei derartigen Investitionen immer vor Augen halten (das gilt genauso für Joysticks und Lichtgriffel, über die wir im nächsten Kapitel sprechen werden): daß nicht allzuviele kommerzielle Programme davon Gebrauch machen. Es werden sich wohl kaum alle anderen Benutzer Ihres Computermodells für diese Extras entschieden haben: dagegen versuchen Softwarehersteller einen möglichst großen Markt zu erfassen. Diese Zusatzgeräte werden nur dann ausgelastet, wenn Sie darauf vorbereitet sind, Ihre eigenen Programme dafür zu schreiben.

Computer-Hardware

In diesem Kapitel werden wir uns die Hardware der Computersysteme vom Standpunkt des Spielers aus anschauen. Welche Systeme können für Spiele und Grafik benutzt werden? Nach welchen Eigenschaften sollten Sie suchen? Mit welchen Zusatzgeräten können Sie Ihr System erweitern, um es mit noch mehr Spaß benutzen zu können?

Der Computer

Eigentlich können Spiele auf jedem Computer gespielt werden, obwohl manche offensichtlich eher für diesen Zweck entwickelt wurden als andere. Manche ermöglichen beispielsweise das Arbeiten mit Farbe und Tönen, während bei anderen eher auf leistungsfähige arithmetische Fähigkeiten Wert gelegt wurde. Welches sind die Kriterien, die den einen Computer geeigneter für Spiele machen als den anderen?

Speicher des Computers

Zweifellos ist das wichtigste Kriterium die Speichergröße Ihres Computers. Davon abhängig ist die Komplexität der Programme, die darauf laufen können. Die Speicherung des Programmes benötigt Speicherplatz: weitere Speicherkapazität wird als Arbeitsspeicher verbraucht – für die Variablen, die Bildschirmanzeigen und so weiter. Wenn Ihr Computer einen kleinen Speicher hat, kann er nur beschränkte Programme verarbeiten.
Wenn Sie einen Computer mit weniger als etwa 5 kB (etwas mehr als 5000 Bytes) Speicher haben, so werden Sie feststellen, daß seine Einsatzmöglichkeiten sehr begrenzt sind. Er kann nur einfache Spielprogramme handhaben: Rennspiele, Spiele wie »Breakout«, einfache Ratespiele und ähnliches. Berufsprogrammierer haben eine Menge Einfallsreichtum darauf verwendet, gute Spiele für kleine

Maschinen zu entwickeln; aber die Resultate können sich von der Vielfalt her einfach nicht mit Programmen messen, die auf größeren Computern, das heißt auf Computern mit mehr Speicher, laufen. Sie werden keine Version von speicherintensiven Spielen für kleine Maschinen finden, und Sie werden gleichfalls auf keine Spiele mit hochauflösender Grafik stoßen (obwohl solche Spiele für diese Maschinen in ROM-Kassetten erhältlich sind).

Viele mittlere Heimcomputer haben 16 kB RAM. Größere Geräte haben 32, 48 oder 64 kB. Manche Geräte werden in verschiedenen Ausbaustufen angeboten: mit geringem oder mit vollem Speicherausbau. Lohnt es sich, für den zusätzlichen Speicher Geld – manchmal eine ganze Menge – auszugeben? Ja, wenn Sie irgendwelche auch nur entfernt ehrgeizigen Pläne mit Ihrem Computer verfolgen!

Schauen Sie sich beispielsweise bei Ihrem Händler oder in einer Computerzeitschrift nach der Anzahl von Spielen um, die für den Sinclair Spectrum/Timex 2000 erhältlich ist, einem bekannten Computer, der entweder mit 16 oder 48 kB RAM ausgerüstet ist. Es gibt einige sehr gute Spiele für die kleinere Maschine, die natürlich auch auf der größeren laufen. Dazu gehören etwa »Hungry Horace«, »Meteor Storm« und »Spectres«, kürzere Abenteuerspiele wie »Sorcerer's Castle« und »Faust Folly« sowie einfachere Simulationen wie »Nightflight« (nicht so dramatisch wie ein Flug am Tag, da man nicht viel von der Erde sieht). Aber nur für die 48 kB-Version gibt es grafische Abenteuer wie »Der kleine Hobbit« und »Pimania«, gute Schachprogramme, hochwertige Videospiele wie »The Penetrator« und so weiter. Wie wir in Kapitel 3 sahen, benötigt die hochauflösende Grafik enorme Mengen an Speicherplatz und Sie finden einfach mehr hervorragende grafische Anzeigen in Programmen für die leistungsfähigeren Heimcomputer.

Es ist überlegenswert, nicht nur die Qualität käuflicher Programme, sondern auch die selbstgeschriebener zu bedenken. Mit 16 kB Speicher kann man ganz gute Programme schreiben – nach unserer Definition eines, mit dem man länger spielen kann als es gedauert hat, es in den Computer einzutippen. Er reicht vollkommen für die Art von Programmen weiter hinten im Buch aus. Aber für den ungeübten BASIC-Programmierer sind 16 kB einfach nicht genug, um etwas wirklich Besonderes zu programmieren (zumal BASIC mehr Speicherplatz benötigt als Maschinensprache, in der viele kommerzielle Programme geschrieben sind). Mit weniger als 16 kB wird man recht häufig die Meldung »OUT OF MEMORY« sehen und wahrscheinlich mehr Zeit mit der Anpassung an den Speicher verbringen als bei der Planung des Spielablaufs.

Grafik- und Tonausstattung

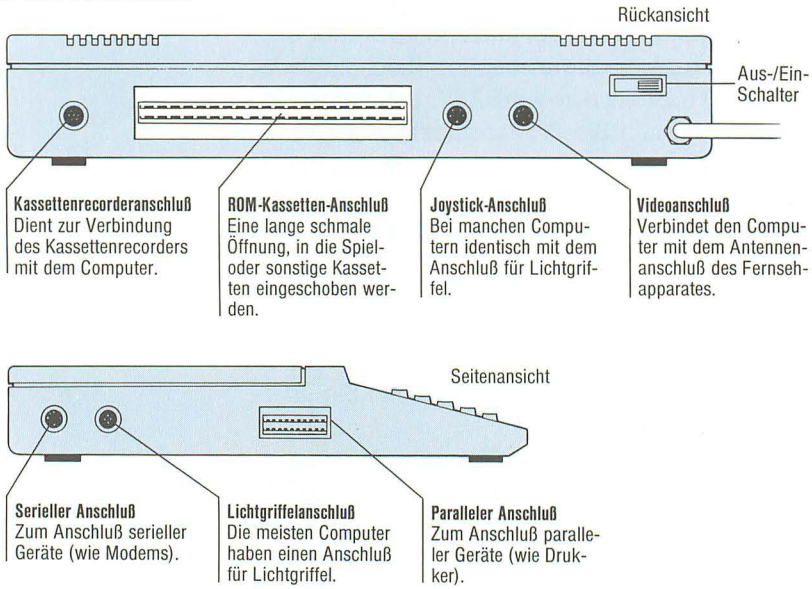
Welche Gesichtspunkte gibt es weiterhin bezüglich Ihres Computers? Grafik und Ton sind offensichtlich sehr wichtig für Sie. Hier sind 2 Gesichtspunkte zu unterscheiden: Was die Maschine selbst kann, und was Sie mit der Maschine anfangen können.
Zuerst wollen wir sehen, was die Maschine tun kann. Selbst einfache Maschinen können dazu überredet werden, mit niedriger Auflösung ansprechende Darstellungen zu zeigen. Wenn Ihr Computer benutzerdefinierbare Zeichen kennt, erweitert das seine Möglichkeiten beträchtlich. Die Darstellung eines Mannes, der mit seinem Hund eine Herde Schafe umrundet, kann sehr eindrucksvoll sein. Ohne die Benutzerdefinition müssen Sie sich auf Text und grafische Zeichen beschränken. Einen Stern über den Bildschirm zu bewegen ist längst nicht so aufregend wie das Steuern eines erkennbaren Raumschiffes, auch wenn das Spiel ansonsten genauso gespielt wird.
Hohe Auflösung wird vielleicht überbewertet. Sie kommt zum Zuge, wenn der Computer ausgetüftelte Bilder – wie das auf Seite 51 – zeichnen soll. Sie ist hervorragend dazu geeignet, Kurven zu zeichnen und für andere geschäftliche und wissenschaftliche Anwendungen. Aber bei den meisten Maschinen ist in der höchsten Auflösung nur noch eine begrenzte Farbauswahl möglich. Meistens sind es gar nur noch 2 in der höchsten Stufe. Viele Spiele sehen in Farbe mit geringerer Auflösung besser aus. Das natürlich nur, falls Ihr Computer Farbe verarbeiten kann. Wenn nicht – wie etwa der Sinclair ZX81/Timex 1000 – versäumen Sie wirklich etwas (obwohl Sie zweifelsohne Geld sparen).
Wie steht's mit Tönen? Ein Rauschgenerator ist für Spiele nützlich, obwohl nicht alle Geräte ihn anbieten. Ein einzelner Tonkanal genügt für begrenzte Toneffekte, ist aber nutzlos, um Musik zu spielen. 3 oder 4 Kanäle mit der »ENVELOPE«-Ausstattung sind wunderbar. Jedoch ist es weder für Sie noch für viele kommerzielle Programmierer leicht, vielfältige Tonerzeugungsmöglichkeiten auch voll auszunutzen. Sie werden merken, daß die diesbezügliche Qualität von Maschinen wie dem BBC-Mikrocomputer oder dem Commodore 64 nicht gleichbedeutend ist mit qualitativ höherstehenden Spielen. Vielleicht ist sogar die Möglichkeit des Computers wichtiger, Töne zu verstärken, indem die Wiedergabe über einen externen Verstärker anstelle des winzigen eingebauten Lautsprechers geschieht. Die meisten davon erzeugen nicht gerade ohren-

betäubende Explosionen, wenn das Raumschiff explodiert, eher ein dumpfes oder sehr gedämpftes »Pffutt«.
Nun zurück zur Frage, was Sie mit den Fähigkeiten Ihres Computers anfangen können! Bei manchen Geräten werden besondere Hardware-Eigenschaften (hochauflösende Grafik und Ton) nur teilweise von BASIC unterstützt. Um sie zu benutzen, müssen Sie in Maschinensprache programmieren oder diese von BASIC aus mit PEEK- und POKE-Kommandos simulieren. (Wir werden das in Kapitel 7 behandeln.) Bei anderen Maschinen erfaßt BASIC (oder eine andere Hochsprache wie etwa FORTH) alle diese Eigenschaften, obwohl ein gewiefter Maschinencode-Programmierer wesentlich mehr herausholt als ein BASIC-Programmierer. Wenn Sie natürlich die meisten Programme fertig kaufen wollen, ist dieser Unterschied für Sie unerheblich.
Die besseren Spielprogramme, von Experten für Maschinen wie Commodore VC20 oder 64 und Atari 400 und 800 geschrieben, nutzen diese Fähigkeiten voll aus. Sie stechen Spiele für Geräte mit (verglichen mit dem Preis) weniger aufregenden Eigenschaften, wie den Tandy Color Computer oder den Spectrum 2000, meistens aus. Wenn Sie jedoch Spiele selbst programmieren wollen, werden Sie bestimmt das Fehlen guter Grafik- und Tonkommandos auf den Commodore- oder Atari-Maschinen »verfluchen«. Auf dem Spectrum/Timex 2000 können Sie beispielsweise ein unterhaltsames Spiel mit nicht allzuviel Anstrengung schreiben, während Sie in BASIC auf dem Commodore 64 leicht die fünffache Zeit brauchen. Sie könnten ein besseres Spiel für den Commodore schreiben, wobei Sie seine Grafik und die drei Tonkanäle voll ausnutzen, aber das könnte sogar 20 mal so lange dauern. Ist es sinnvoll, soviel Zeit zu investieren? Das können nur Sie entscheiden.

Der Ausbau Ihrer Maschine

Wenn Sie einen Computer aussuchen, sollten Sie nicht nur die eingebaute Ausrüstung in Betracht ziehen, die Sie jetzt benötigen, sondern auch Zusatzgeräte, die Sie vielleicht in Zukunft einmal verwenden wollen. Dieser wichtige Gesichtspunkt wird in der Werbung nicht gerade hervorgehoben.

Ports und Anschlüsse



Auf welche Weise können Sie zusätzliche Hardware an Ihren Computer anschließen? Prinzipiell gibt es 2 verschiedene Lösungen: Zum einen können Sie das Gehäuse des Computers öffnen und zusätzliche Komponenten einstekken – normalerweise sind das Chips, die in die dafür freigelassenen Steckplätze auf der Hauptplatine des Computers passen. (Dies sollten Sie nur machen, wenn Sie eine gewisse Erfahrung besitzen; bitten Sie ansonsten Ihren Computerhändler, es für Sie zu tun.) Zum anderen können Sie Zubehör mit dafür vorgesehenen äußeren Anschlüssen verbinden. Dies können Buchsen sein, in die passende Stecker gesteckt werden – beispielsweise haben nahezu alle Heimcomputer Buchsen für Kassettenrecorder (oder auch Steckerleisten). Eine Gehäuseöffnung macht quasi einen Teil der Hauptplatine zugänglich, an die dann direkt Zusatzgeräte angeschlossen werden. Eine besondere Version davon ist der Anschluß für ROM-Kassetten, über den die Kassettenschaltkreise direkt mit der Hauptplatine verbunden werden.

Es gibt verschiedene Arten von »Ports« (»Port« ist der Fachausdruck für eine Anschlußstelle). Die Unterschiede sind nicht nur schaltungstechnischer Art, sondern liegen auch in der Art der zugehörigen Programmierung – die diversen Arten, Ports zu programmieren, damit sie mit

verschiedenem Zubehör zusammenarbeiten können. Wir können die einzelnen Typen hier nicht im Detail diskutieren. Wichtig ist aber festzustellen, welche Ports Ihr Computer bietet und welcher Port von dem Zubehör gefordert wird, das Sie einsetzen wollen.
Manchmal kann die Zahl vorhandener Anschlüsse durch Zusätze erweitert werden. Sie haben vielleicht schon für schwer erweiterbare Computer – wie den Sinclair ZX81 – Einheiten gesehen, die die software- und hardwaremäßige Aufgabe übernehmen, eine einfache Steckerleiste zu einer Reihe von Ports auszubauen.

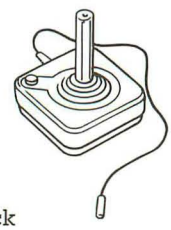
Speicherausbau

Eine sehr wichtige Möglichkeit, Ihren Computer auszubauen, muß es sein, seine oben genannten grundlegenden Einschränkungen auszugleichen. Wenn Ihr Gerät wenig Speicher hat, sollten Sie eventuell Extraspeicher hinzufügen können. Wenn das BASIC begrenzt ist, sollten Sie es vielleicht mit zusätzlichen BASIC-Fähigkeiten versehen können. Wenn es keinen Ton oder hochauflösende Grafik hat, sollte es vielleicht um diese Fähigkeiten erweiterbar sein.
Wir wollen diese Punkte nacheinander diskutieren. Die höchste Speichermenge, die ein Heimcomputer normalerweise verkraften kann, sind 64 kB. Das liegt darin begründet, daß der Mikroprozessor des Computers nur 65 536 verschiedene Adressen gleichzeitig verarbeiten kann. Um Speicher hinzuzufügen, falls Ihr Computer weniger hat, kaufen Sie entweder Speicherbausteine, die im Inneren des Computers angeschlossen werden, oder sogenannte »RAM-Packs«, die außen angeschlossen werden. Die erste Lösung ist die elegantere, aber nicht alle Computer können mit zusätzlichen Speicherbausteinen versehen werden. Andererseits können RAM-Packs Probleme verursachen, wenn sie nicht richtig befestigt sind. Was Sie bekommen können, hängt vom Angebot für Ihren Computer ab. Die Speichererweiterungen unterscheiden sich in der Benutzung nicht, so daß es sinnvoll ist, für die benötigte Menge und den Typ das billigste Angebot ausfindig zu machen.
Um die vorhandenen Fähigkeiten oder die des zusätzlichen Speichers besser ausnutzen zu können, brauchen

Sie geeignetere Befehle. Hier kommen Extended BASIC, Grafikpakete und dergleichen zum Zuge. Sie versehen den Computer mit leistungsfähigeren Kommandos und erleichtern das Programmieren.
Diese Erweiterungen sind am einfachsten zu benutzen, wenn sie in ROM-Form (Read Only Memory) vorliegen. Sie haben vielleicht einen ROM-Chip oder eine ROM-Kassette oder eine vollständige separate Einheit, die an den Computer angeschlossen wird. Manchmal kommen diese Kommandos auch als Software in Form von Programmen auf Kompaktkassetten oder Floppy Disks vor. Gerade Kassetten sind umständlicher zu handhaben als ROMs. Anstatt das ROM einzustecken und es anschließend zu »vergessen«, muß hier die zusätzliche Software jedesmal mit LOAD geladen werden, wenn der Computer eingeschaltet wird.

Joysticks und Drehregler (Paddles)

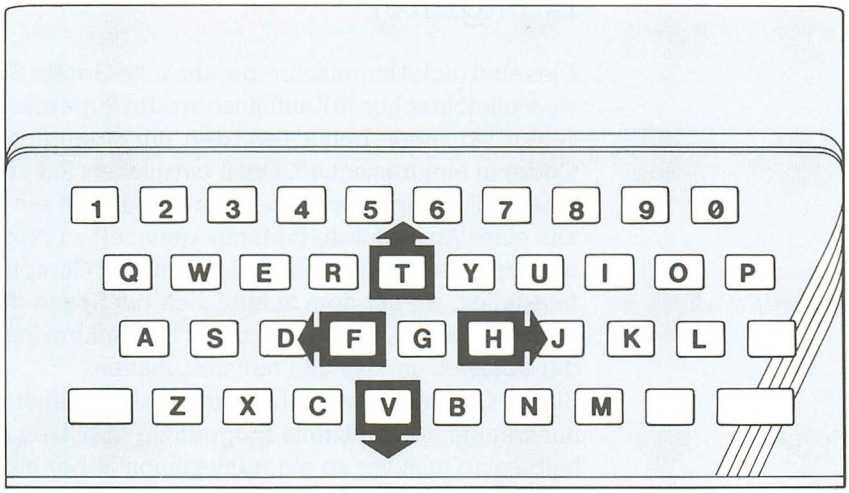
Sie kennen bestimmt schon Joysticks. Wenn Sie jemals mit einem Fernsehspiel gespielt haben, dürften Sie sie sehr wahrscheinlich benutzt haben. Es sind die Kontrollgeräte der Spieler, mit einem »Stab« in der Mitte, der bewegt werden kann, um ein oder mehrere Elemente auf dem Bildschirm zu kontrollieren – den Cursor, ein Raumschiff oder ein Wesen, das ein Labyrinth durchdringt. Normalerweise besitzen Sie auch einen »Feuerknopf«, mit dem Raketen abgeschossen werden, und manchmal zusätzliche Kontrollelemente, um die Geschwindigkeit zu verlangsamen oder zu erhöhen und um andere Befehle an den Computer zu geben.
Die meisten auf Spiele ausgerichteten Heimcomputer (wie die Atari-Geräte oder die Commodore VC20 und 64) haben Ports für Joysticks. Aber Joysticks können natürlich nur bei Programmen verwendet werden, die so geschrieben sind, daß sie deren Daten auch einlesen. Es gibt in der Tat nur wenige kommerzielle Programme, die einen Joystick benutzen. Normalerweise ist das auf der Packung angegeben. Sie können aber auch Ihre eigenen Programme schreiben, um die Joysticks zu benutzen. Wie das gemacht wird, sehen Sie in dem Programm »Breakout« für den Dragon auf Seite 129 ff.



Joystick

Drehregler (Paddle)

Tastatur des ZX Spectrum/Timex 2000

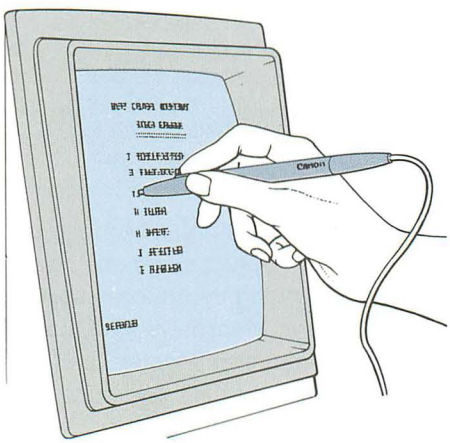


Spiele können über die Tastatur kontrolliert werden. Manchmal werden die Cursor-Tasten verwendet (mit oder ohne Großbuchstabentaste, abhängig vom Computermodell). Auf anderen Maschinen werden die mittleren Tasten benutzt, wie hier auf dem Spectrum: die Richtungstasten für ein typisches Spiel sind eingezeichnet. Bei manchen Spielen für den Spectrum wird die Auf/Ab-Bewegung mit der Z- und der Q-Taste von der linken Hand gesteuert und die Links/Rechts-Bewegung mit O und P durch die rechte Hand. Viele Spiele für Heimcomputer fragen am Anfang, welche Tasten benutzt werden sollen.

Es gibt eine Menge verschiedener Modelle, aber nicht alle arbeiten mit allen Computern zusammen, so daß Sie sich vor der Auswahl sorgfältig informieren sollten. Die teureren sind meistens den Mehrpreis wert, da sie normalerweise genauer arbeiten und leichter handhabbar sind.
Manche Computer unterstützen Joysticks nicht direkt, so wie beispielsweise die Sinclair/Timex-Geräte. Wenn Ihr Computer nicht ohne weiteres mit Joysticks arbeitet oder Sie nicht noch mehr Geld dafür ausgeben möchten, können Sie Ihre Spiele auch über die Tastatur steuern, indem Sie verschiedene Tasten für die Richtungen Links, Rechts, Hoch und Herunter (manchmal noch mehr) verwenden. Manche Leute empfinden das als absolut ausreichend; andere finden es schwierig und sehen Joysticks als absolutes Muß an. Wenn Sie jedoch die Schwierigkeit auf sich nehmen, Joysticks an ein Gerät anzuschließen, welches das sonst nicht erlaubt (dafür wird ein sogenanntes »Interface« benötigt), werden Sie vermutlich Schwierigkeiten haben, passende Software zu finden (vielleicht hat irgendein Händler das, was Sie brauchen).
Paddles (Drehregler) sind genauso wie Joysticks dazu geeignet, Spiele zu kontrollieren. Manche Geräte bieten sie als Alternative an, viele nicht.

Lichtgriffel

Dies sind dicke kugelschreiberähnliche Geräte. Sie haben sie vielleicht schon in Kaufhäusern oder Supermärkten gesehen, wo sie oft benutzt werden, um Streifencodes (Bar Code) in elektronische Kassen einzulesen. Sie können an einem Bildschirm verwendet werden, um auf ein Element aus einer Auswahlliste (»Menü« genannt) zu zeigen oder um ein Bild auf den Schirm zu malen. Der Computer kann feststellen, wo auf dem Schirm sich die Spitze des Lichtgriffels befindet, da die Elektronik den Schirm immer wieder abtastet, um das Bild neu aufzubauen.
Einige Computer bieten Lichtgriffel als Zubehör an, aber nur wenige kommerzielle Programme benutzen sie. Deshalb sollte man vor so einer Investition lieber auf ein geeignetes Programm warten!



Die Benutzung eines Lichtgriffels
Ein Lichtgriffel erkennt den hellen Punkt, der ununterbrochen den Bildschirm von links nach rechts und von oben nach unten überstreicht (damit wird das Bild erzeugt). Sobald der Punkt unter der Spitze des Lichtgriffels vorbeikommt, merkt das der Computer und kann die genaue Position des Stiftes berechnen. Mit vielen Lichtgriffeln können Zeichen unter der Spitze erzeugt werden. Wenn dieses Zeichen ein Punkt ist, kann mit mehreren Punkten eine Linie erzeugt und so direkt auf den Schirm gezeichnet werden.

Roboter und Schildkröten

Roboter werden nicht *in* Spielen benutzt: sie *sind* Spiele! Einige Roboter und ähnliche kleine Geräte können von Heimcomputern kontrolliert werden. Dieses Gebiet wird sich in den nächsten Jahren sehr wahrscheinlich noch ausweiten. Zu allen solchen Geräten gehören Software-Pakete, um sicherzustellen, daß der Computer zur Kontrolle entsprechend programmiert werden kann.

Ein besonderer Typ von Roboter ist die Schildkröte (»Turtle«). Dies ist ein kleines halbkugelförmiges Gerät auf Rädern, auf dessen Unterseite ein Schreibstift eingebaut ist. Durch Steuersignale vom Computer kann die Schildkröte in verschiedenen Richtungen fahren und den Stift heben oder senken, um während der Bewegung Linien zu ziehen. Dieser Prozeß wird »Turtle-Grafik« genannt. Oft wird dafür die Sprache LOGO eingesetzt. Wenn Sie keine mechanische Schildkröte besitzen, können Sie mit ähnlichen Grafikkommandos eine sichtbare »Schildkröte« bewegen – eine kleine pfeilförmige Markierung auf Ihrem Bildschirm. Das Besondere an der Turtle-Grafik ist, daß die Schildkröte immer in eine ganz bestimmte Richtung schaut. Normale Grafikkommandos beschreiben einen Punkt auf dem Schirm und sagen dann dem Computer, in einer bestimmten Richtung zu zeichnen – Hoch, Herunter oder auf einen anderen Punkt zu. Turtle-Grafik-Kommandos sehen eher so aus: »Geh 10 Einheiten geradeaus, dreh dich dann um 80° nach links, geh dann 20 Einheiten geradeaus, dreh dich dann um und geh 5 Einheiten zurück.« Das kann anfangs verwirrend sein, aber viele Leute können damit leichter arbeiten als mit den konventionellen Kommandos.

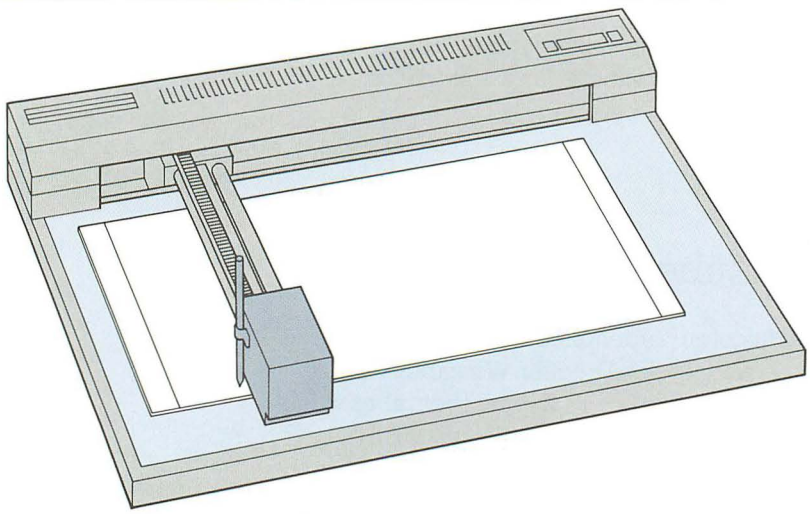
Sonstiges Zubehör

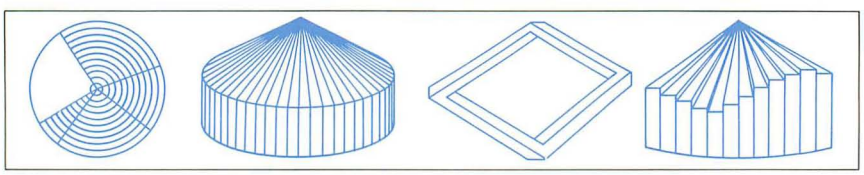
Eines der wichtigsten Zubehörteile haben wir bis jetzt noch gar nicht erwähnt – den Drucker. Wir haben nicht den Platz, Drucker hier ausführlich zu besprechen, aber wenn Sie an Grafik interessiert sind, müssen Sie das auch bei der Auswahl eines Druckers beachten. Matrixdrucker können meistens auch programmiert werden, Diagramme, Kurven und Bilder auszudrucken. Mit Typenraddruckern ist das nicht so leicht.

Plotter sind speziell dafür ausgelegt, Grafiken zu produzieren. Sie zeichnen glattere Linien als jeder Drucker, und viele können in verschiedenen Farben arbeiten, indem sie gleichzeitig einen Satz Farbstifte verwenden. Sie sind jedoch sehr teuer.

Wenn Sie ernsthaft mit hochauflösender Grafik arbeiten wollen, werden Sie wahrscheinlich die schlechte Bildqualität Ihres Fernsehschirmes als Handicap empfinden. Ein Farbmonitor liefert ein wesentlich besseres Bild. Es gibt 2 verschiedene Typen: RGB (Rot – Grün – Blau) und PAL, die jeweils unterschiedliche Systeme zur Signalerzeugung

Die LOGO-Schildkröte
Die Schildkröte ist eine Art von Roboter, die angewiesen werden kann, Linien oder Muster auf Papier zu zeichnen. Grafikkommandos für die Schildkröte werden normalerweise in der Sprache LOGO gegeben. Um Bewegungen zu befehlen, werden keine festen Koordinaten – wie normalerweise für Grafik – angegeben, sondern statt dessen immer Daten in bezug auf die aktuelle Schildkrötenposition.





Plotter
Ein Computer kann programmiert werden, mit einem Flachbettplotter hochwertige farbige Grafiken auf Papier zu erzeugen (oben). Bei den meisten preiswerten Plottern ist der Stift an einem Balken befestigt, der sich über das Papier bewegt; der Stift seinerseits bewegt sich auf und ab und zeichnet dabei auf dem Papier. Bei einigen Geräten bewegt sich auch das Papier. Einige einfache Beispiele für die Art von Grafik, die ein Plotter produzieren kann, sehen Sie hier.

verwenden. Sie können aber ohnehin nur einen Monitor benutzen, wenn Ihr Computer zusätzlich zu dem normalen UHF-Fernsehausgang noch einen Videoausgang besitzt. Manchmal ist die Bildqualität jedoch durch die vergleichsweise simplen Schaltkreise des Computers selbst begrenzt, und der Einsatz eines Monitors bewirkt keine Verbesserung. Das ist ein Grund dafür, daß die billigsten Computer oftmals keinen Videoausgang haben.

In der Zukunft

Die Qualität der Heimcomputer und der Umfang des erhältlichen Zubehörs steigern sich von Tag zu Tag. Wie sieht es in der Zukunft aus? Was kann der Spieler in vielleicht 10 Jahren erwarten?
Eine Eigenschaft, die häufiger anzutreffen sein wird, ist die Mehrfachbenutzbarkeit – Spiele, die nicht nur von 2 Spielern, sondern von 4 oder mehr gespielt werden (jeder mit eigenem Joystick). Mit mehr als 2 Spielern wird es schwer, jedem einen vernünftigen Blick auf den Bildschirm zu verschaffen. Ein zweiter Bildschirm wird dieses Problem lösen!
Wir können erwarten, daß hochwertige Grafik immer mehr Standard wird. Computerspeicher wird ständig billiger, und das wird es erleichtern, vielfarbige hochauflösende Grafik anzubieten. Neue Computerbildschirme (vielleicht flache) könnten die Qualität und die Auflösung weiter verbessern.
Eine Schwierigkeit mit Grafik liegt jedoch nicht in den Fähigkeiten der Maschine, sondern in der benötigten Zeit, Bildschirmanzeigen zu entwerfen. Verbesserte Software wird dieses Problem lösen. Mit einem Lichtgriffel etwa läßt sich ein kompliziertes Bild viel schneller erstellen als mit BASIC-Kommandos wie DRAW oder LINE. Spezielle Hardware und Software-Grafikpakete werden es bald jedem von uns ermöglichen, künstlerische Meisterwerke auf dem kleinen Bildschirm zu erzeugen!
Bereits heute ist ein Trend zu attraktiveren Bildern in Abenteuerspielen erkennbar, und in der Zukunft werden mehr und mehr Spiele wirklich gute Bilder aufzuweisen haben. Die Umwelt und die Charaktere werden realistischer wirken, und der Spieler wird mehr und mehr Elemente kontrollieren können. Spiele, die auf dem wirklichen Leben

basieren, werden so gut wie alle Einflußfaktoren berücksichtigen, und nicht nur ein paar davon.
Kassetten sind ziemlich umständlich in der Benutzung – wie Sie wohl wissen, wenn Sie es probiert haben! Sie können leicht mehr Zeit mit dem Laden eines Programmes verbringen, als hinterher damit zu spielen. Wir erwarten, daß Geräte mit leichterer Handhabung (wie Endlosbänder oder Floppy Disks) immer verbreiteter werden – und billiger. Wenn Benutzer immer mehr »ernsthafte« Anwendungen für ihre Heimcomputer finden, können sie auch die höheren Kosten solcher Zusatzgeräte vertreten und sie dann natürlich auch für ihre Spiele einsetzen.
Gleichgültig welche Ausrüstung Sie jetzt haben oder sich kaufen wollen: Sie können sicher sein, daß sie in den nächsten Jahren billiger wird. Aber warten Sie nicht, um den Sprung zu wagen. Fast jeder kann sich heute ein einfaches Computersystem leisten. Wenn es Sie nicht mehr reizen sollte, können Sie sich mit geringen Zusatzkosten ein besseres System zulegen. Und von da aus können Sie sich zu immer noch besseren Geräten weiterbewegen.

Das Programmieren von Spielen

In diesem und dem nächsten Kapitel wollen wir uns Spielprogramme anschauen, die Sie mit BASIC als Programmiersprache für sich selbst schreiben können. Zuerst wollen wir erwägen, was Sie beim Schreiben der Programme beachten müssen: Welche Spiele Sie selbst schreiben können, und wie Sie die Programme planen sollten. In Kapitel 7 wollen wir dann detailliert die Vorgehensweise beim Programmieren einiger Spiele besprechen und Beispiellistings bringen, um Ihnen zu zeigen, wie sie arbeiten.

Programmieren lernen

Besitzen Sie bereits Grundkenntnisse des Programmierens? Wenn nicht, müssen Sie zuerst die Grundlagen lernen, entweder aus Ihrem Computermanual oder aus einem einfachen Programmierlehrbuch. Wenn Sie sich auf Spiele und Grafik konzentrieren wollen, versuchen Sie ein Buch zu bekommen, das dafür auch Beispiele enthält. Sie werden feststellen, daß gewisse Programmierlehrbücher den Schwerpunkt auf Geschäftsanwendungen und mathematische Programme legen.

Einige Spielprogramme sind sehr einfach. Beispiele wie »Rate mal, welche Zahl sich der Computer denkt« sind in vielen Programmierbüchern zu finden. Wenn Sie sie durcharbeiten, werden Sie anfangen zu lernen, wie Zufallszahlen und andere Eigenschaften von BASIC in Spielen benutzt werden.

Wenn Sie erst einmal ein wenig BASIC verstanden haben, ist es eine gute Möglichkeit weiterzumachen, indem Sie sich Programmlistings für Ihren Computer beschaffen, also Listen mit Programmbefehlen zum Eintippen. In Computerzeitschriften werden Sie viele Listings finden. Blättern Sie einmal verschiedene Ausgaben durch, um zu sehen, ob sie irgendwelche Listings für Ihr Gerät enthalten! (Wir werden das auf Seite 144 ausführlicher besprechen.) Vielleicht finden Sie sogar ganze Bücher mit Programmen für Ihren Computer. Tippen Sie alle Programme ein, die Sie finden: Das mag Ihnen vielleicht wie Stunden langweiliger

Arbeit vorkommen, aber Sie werden langsam lernen, wie Spiele geschrieben werden – und gleichzeitig eine kleine »Bibliothek« einfacher nützlicher Programme aufbauen.
Sie werden sich natürlich auch einige Programme auf Kompaktkassette, ROM-Kassette oder Disketten kaufen wollen. Aber erwarten Sie nicht allzuviel Lehrreiches von ihnen. Oft können Programme auf Kassetten nicht gelistet werden: Sie wurden von ihren Verfassern gegen Mißbrauch geschützt. Mit Sicherheit können Sie keine Programme aus ROM-Kassetten listen. Außerdem laufen Sie bei diesen Programmen Gefahr (obwohl sie vielleicht zeigen, was wirklich aus Ihrem Computer herausgeholt werden kann), daß Sie irgendwie auf einen falschen Pfad beim Schreiben Ihrer eigenen Programme geführt werden. Das, was professionelle Programmierer am besten können – schnelle Actionspiele und dergleichen –, ist nicht unbedingt für Sie zur Nachahmung geeignet. Bis Sie zu einem Experten geworden sind, sollten Sie nicht versuchen, gekaufte Programme zu ersetzen, sondern zu ergänzen!

Welche Art von Programmen sollten Sie schreiben?

Wenn Sie noch am Anfang stehen, sollten Sie nicht zu ehrgeizig sein. Träumen Sie nicht gleich davon, Ihr erstes »Meisterwerk« an eine Zeitschrift zu verkaufen! Konzentrieren Sie sich stattdessen auf ein paar einfache Programme, mit denen Sie auch gerne spielen möchten! Sie werden wahrscheinlich erkennen, daß ein halbes Dutzend einfacher Programme mehr Freude bringen kann, als ein einziges kompliziertes. Sie werden auf diese Art auch weniger Zeit vergeuden. Es kann schon ganz schön frustrieren, Stunde um Stunde in ein ehrgeiziges Programm zu investieren, nur um dann zu merken, daß es nicht wie geplant funktioniert oder – was vielleicht noch schlimmer ist – daß es einfach keinen Spaß macht, damit zu spielen.
Viele der alten Spiele sind sehr gut. Sie werden im nächsten Kapitel Programmideen für bekannte Spiele finden. Etwas wie »Mastermind« hört sich vielleicht nicht so aufregend an wie »Space Invaders«, aber es kann überraschend viel Spaß machen. Außerdem ist es nicht schwer, das Hauptprogramm auszuarbeiten (besonders, weil wir schon die meiste Arbeit für Sie erledigt haben). Sie können

sich darauf konzentrieren, es nett zu gestalten, mit ansprechenden Bildern zu versehen und die ganze »Hausarbeit« zu erledigen, die ein Spiel erst erfreulich macht.

Die Planung Ihrer Programme

Es ist wichtig, sich beim Programmschreiben eine bestimmte Methode anzueignen. Versuchen Sie, das Programm in kleine Stückchen aufzuteilen – jedes mit lediglich 5 oder 10 Programmzeilen, nie mehr als 20 –, die Sie dann gründlich austesten können. Wenn Sie sicher sind, daß jedes kurze Segment richtig arbeitet, können Sie sie zu einem funktionsfähigen Programm zusammenbauen!
Handbücher liefern nicht viel Unterstützung beim Erlernen des Programmierens. Sie konzentrieren sich vielmehr darauf, Schlüsselworte und ihre Benutzung zu erklären. Beim Programmieren dagegen ist es vielmehr nötig zu lernen, wie ein Algorithmus, ein »Spielplan«, entwickelt wird und wie die Entwicklung und das Testen in vernünftigen Etappen geplant wird. Für den Fall, daß es Ihnen nicht klar ist, wie ein typisches Spiel aufgegliedert wird, folgen hier einige Vorschläge.
Zuerst erarbeiten Sie das Hauptprogramm. Erfassen Sie die grundlegende Strategie, und vergewissern Sie sich, daß das Spiel so arbeitet, wie Sie es wollen! Planen Sie es so einfach wie möglich! Kümmern Sie sich jetzt noch nicht um benutzerdefinierte Zeichen: Benutzen Sie statt dessen alphabetische Zeichen! Beachten Sie ferner noch nicht die verschiedenen Spielgrade, sondern halten Sie sich an den einfachsten! Testen Sie diese einfache Spielversion gründlich, bevor Sie versuchen, sie zu verbessern! Nehmen Sie sich die Zeit, jetzt alles ins richtige Lot zu bringen! Vielleicht machen Ihnen spezielle Feinheiten mehr Spaß, etwa die Bewegungen herauszuarbeiten; aber gute Bildschirmanzeigen in einem schlechten Spiel sind kein Ersatz für ein gutes Spiel. Bemühen Sie sich von Anfang an um ein gutes Spiel – eines, das Sie auch später noch gerne spielen –, ehe Sie das komplette Programm »drumherum« schreiben!
Als nächstes bestimmen Sie die Anzeige oder die Anzeigen auf dem Bildschirm. Wie soll das »Spielfeld« aussehen (das Schachbrett, das Labyrinth, die Reihe Invaders oder was auch immer)? Planen Sie den Bildschirm auf Zeichenpapier – auf Millimeterpapier oder einem speziellen Zei-

chenblock für Ihren Computer! (Wenn Sie keinen finden, fotokopieren Sie den Plan für die Bildschirmorganisation Ihres Computers oder zeichnen ihn ab). Erarbeiten Sie zuerst die festen Elemente und dann die bewegten! Können sich die Figuren bewegen, ohne den Hintergrund zu stören? Können sie sich sanft gleitend bewegen? Sie werden bald lernen, was Sie mit der Grafik Ihres Computers anfangen können, und daß Sie irgendwelche grandiose Ideen auf Ihre eigenen Programmierfähigkeiten zurechtstutzen müssen.

Sie finden einige Vorschläge zur Planung des Bildschirms im nächsten Kapitel. Unsere Programme zeigen, wie wir verschiedene Schwierigkeiten behandelt haben, die beim Arbeiten mit dem Dragon und dem Tandy Color Computer auftauchen – beispielsweise, daß er unbedingt schwarz auf einen grünen Hintergrund schreiben will. Wenn Sie einen anderen Computer haben, gibt es andere Stärken und Schwächen, um die Sie sich kümmern müssen. Wir deuten an, wo diese vielleicht liegen könnten, und machen ein paar Vorschläge, wie verschiedene Computer behandelt werden sollten, um das Beste aus ihnen herauszuholen.

Nach der Planung der Anzeige erarbeiten Sie die Titelsequenz. Es macht einen Riesenunterschied, wenn ein Spiel eine gute Einleitung hat, um die Spieler in die richtige »Stimmung« zu versetzen. Vergessen Sie eine Zusammenfassung der Spielregeln nicht, und erläutern Sie, welche Tasten beispielsweise bedient werden müssen! Jetzt wissen Sie natürlich noch, daß Sie »F« drücken müssen, um zu feuern und nicht die Leertaste, aber irgendwann haben Sie ein Dutzend Spiele gespeichert und vergessen, was genau in jedem einzelnen gemacht werden muß. Deshalb benötigen Sie die Erinnerung auf dem Bildschirm.

Wenn Ihr Computer knapp an Speicherplatz ist, können Sie daraus ein eigenes »Programm« machen. Auf vielen Geräten können Sie im Einleitungsprogramm ein Kommando geben, durch das der Computer automatisch das richtige Spielprogramm lädt und startet.

Richten Sie zu guter Letzt Ihr Augenmerk auf die Feinheiten! Wie wird der Punktestand angezeigt? Kann eine Tabelle der höchsten Punktzahlen irgendwo auf dem Schirm angezeigt werden? Wollen Sie Toneffekte? Wenn ja, dann programmieren Sie sie jetzt! Probieren Sie sie zuerst einzeln aus und fügen Sie sie dann ins Hauptprogramm ein! Stellen Sie sicher, daß es einwandfrei gemeldet wird, wenn ein Spieler gewonnen hat! Sie könnten vielleicht ein besonderes »Sie haben gewonnen!«-Bild mit blinkenden Lichtern und Freudentönen programmieren.

Reservieren Sie eine besondere »Freispiel-Sequenz« und so weiter!
Sie sollten nicht glauben, mit einem selbstgeschriebenen Programm oder mit einem aus irgendeiner Quelle stammenden Listing jemals ganz fertig zu werden. Wenn Sie neue Programmiertechniken lernen, können Sie Ihre alten Versuche immer noch verbessern, entweder im Erscheinungsbild oder in der Geschwindigkeit, so daß sie eine neue Herausforderung darstellen. Darum geht es eigentlich beim Programmieren!

Einige allgemeine Probleme

Sie werden immer Probleme beim Programmieren haben. *Jeder* hat sie. Sie könnten schwören, daß alles richtig ist, und dann erscheint der Punktestand in der Mitte anstatt in der Ecke (wie beabsichtigt); oder der Punktestand wird nicht korrekt hochgezählt; oder das Spiel läuft in der ersten Runde absolut richtig, aber dann spielt es verrückt... die Liste ist endlos.
Wir können Ihnen bei den ganz speziellen Problemen, die Sie haben werden, nicht helfen. Sie müssen sich schon hinsetzen und selber daran herumtüfteln. Jedes gute Buch über Programmieren wird Ihnen einige Ratschläge zum Testen und zum »Entfehlern« von Programmen geben. Aber wir können einige allgemeine Probleme beim Schreiben von Spielprogrammen aufführen.
Ein Hauptproblem ist die Speicherkapazität – was wir ja auch schon früher erwähnt haben. Wie können Sie den Speicher Ihres Gerätes optimal ausnutzen? Verschiedene Computer verwalten ihre Speicher auf unterschiedliche Weise; sie speichern beispielsweise Matrizen unterschiedlich. Zeitschriftenartikel über Ihren Computer können hilfreiche Tips über dessen Eigenheiten geben und wie sie am besten ausgenutzt werden können. Wir geben hier einige allgemeine Vorschläge.
Denken Sie zuerst daran, wieviel Speicher die hohe Auflösung benötigt! Auf vielen Computern erhalten Sie wesentlich mehr Speicherplatz für Ihre Spiele, wenn Sie sich an eine niedrigere Bildschirmauflösung halten. Für viele Spiele sind einfache, grobe Anzeigen gut genug. Versuchen Sie zuerst die niedrige Auflösung! Nur wenn sie sich als

unangemessen herausstellt, sollten Sie zu den höheren Versionen übergehen.
Bedenken Sie dann, wieviel Platz Ihr Programm benötigt! Vergewissern Sie sich, daß Ihr Programm logisch aufgebaut ist! Wenn sich irgendwelche Kommandofolgen mehr als zweimal wiederholen (bei langen reicht schon zweimal), sollten Sie eigene Unterprogramme oder Prozeduren daraus machen, so daß die Kommandos nur einmal eingegeben werden. Wenn Sie Platzprobleme haben, beschneiden Sie unbarmherzig Ihr Listing! Entfernen Sie alle Blanks und REM-Zeilen, denn sie benötigen unnötig viel Speicher. (Aber bewahren Sie sich irgendwo ein dokumentiertes Programmlisting auf, so daß Sie nie die Übersicht über die Funktion verlieren). Nicht alle Computer geben bei Programmverkürzungen den entsprechenden Speicherplatz automatisch frei. Speichern und laden Sie deshalb die gekürzte Version, um sehen zu können, wie Sie vorankommen!
Denken Sie an Ihre Daten! Matrizen können Speicherplatz auffressen. Es hilft oft, einzelne Variable zu verwenden (als Beispiel A1, A2, A3 anstatt einer Matrix mit 3 Elementen wie A(3). Lange Variablennamen, falls Ihr Computer sie überhaupt erlaubt, sehen schön aus, aber auch sie verbrauchen Speicherplatz.
Schließlich könnten Sie Ihr Programm auch aufteilen. Ein Einleitungsprogramm könnte nicht nur die Titelseite liefern, sondern beispielsweise auch die Definition der selbstdefinierten Zeichen enthalten. Diese werden beim Laden eines Programmes oft nicht gelöscht, wie etwa der Hauptspeicher. Wenn Ihr Programm 2 oder mehr Spielebenen enthält, machen Sie daraus eigene Programme. Durch sorgfältige Planung können Sie sicherstellen, daß es den Benutzern klar ist, wie sie von einer Ebene auf die nächste gelangen können.
Weiterhin bereitet die Geschwindigkeit den Programmierern oft Kopfzerbrechen. Sie glauben, daß Sie eine tolle Methode gefunden haben, ein schnelles Spiel wie »Breakout« zu programmieren, aber nach dem Eintippen stolpert der Ball mit der Geschwindigkeit einer fußkranken Schnecke herum. Wie schaffen es nur diese professionellen Programmierer, daß alles so schnell abläuft? Oft benutzen sie aus genau diesem Grund Maschinensprache anstatt BASIC. Wenn Sie es ihnen nicht unbedingt gleichtun wollen, aber trotzdem einigermaßen schnelle Programme schreiben möchten – hier sind einige einfache Tips:
Schauen Sie sich die Programmlistings genau an, um zu sehen, ob es besondere Gründe für die Verzögerungen gibt! Haben Sie zu viele Anweisungen in einer Schleife? Schnei-

den Sie alles Unwichtige heraus, oder legen Sie es an eine andere Stelle des Programms! Verzögern Toneffekte den Programmablauf? Holen Sie sie aus der Schleife für die Bewegung heraus, und kompensieren Sie das durch zusätzliche Töne an anderen Stellen!
Weiterhin kann die hohe Auflösung das Problem sein. Figuren zu zeichnen und besonders sie auszumalen dauert recht lange. Zeichnen und Bewegen von normalen Zeichen niedriger Auflösung geht sehr viel schneller.
Viele Computer bieten verschiedene Möglichkeiten, denselben grafischen Effekt hervorzurufen. Auf dem Dragon können Sie mit SET oder RESET Ihre Pixel in der mittleren Auflösung beeinflussen oder die entsprechenden Werte in die steuernden Speicherzellen POKEn. Die einzelnen Geschwindigkeiten dieser Methoden variieren von Computer zu Computer. Probieren Sie verschiedene aus, und schauen Sie, welche am effektivsten auf Ihrer Maschine arbeitet!
Manchmal wird die Arbeitsgeschwindigkeit eines Computers künstlich verlangsamt, um das Laden und Speichern von Programmen sicherer zu machen. Sie können die Geschwindigkeit durch das Ändern eines passenden Speicherwertes beeinflussen. Wir haben dies in einigen unserer Programme getan. POKE & HFFD7,0 beschleunigt den Dragon, POKE & HFFD6,0 verlangsamt ihn wieder. (Nach Programmende dürfen Sie das nicht vergessen!) Wenn Sie einen anderen Computer benutzen, müssen Sie die entsprechenden Zahlen für diesen Trick herausfinden (wenn er möglich ist) – aus Ihrem Handbuch oder einem Buch mit Programmiertips für Ihr Gerät.
Das mag aber jetzt für eine Einleitung reichen! Wir wollen uns nun ein paar richtige Programme anschauen.

Einige Programme zum Ausprobieren

In diesem Teil des Buches werden wir ausführlich Programme für einfache Spiele, Grafik und Töne besprechen. Wir werden uns ansehen, wie verschiedene Effekte programmiert werden und vorschlagen, wie die Programme angepaßt werden könnten, um die Kapazität verschiedener Computer voll auszunutzen. Außerdem zeigen wir für jedes Spiel ein Beispiellisting.

Die Listings sind in Microsoft-BASIC geschrieben und so allgemein wie möglich gehalten, so daß jeder sie ohne größere Schwierigkeiten an jeden beliebigen Computer anpassen kann – natürlich innerhalb vernünftiger Grenzen. (Es ist sinnlos, auf einem Computer Orgel spielen zu wollen, der nicht einmal irgendwelche Töne erzeugen kann!) Sie wurden alle getestet und laufen so wie abgedruckt auf dem Tandy Color Computer mit Extended BASIC und auf dem Dragon 32. Sie werden auch auf einer Menge ähnlicher Geräte funktionieren. Eigentlich sollen sie auch eher Ideen vermitteln und Programmierstrategien erklären, und nicht so sehr ausgefeilte Programme darstellen; deshalb sollte es keine Probleme geben, sie an eine andere Maschine anzupassen. Die Hauptsache ist: zu verstehen, was wir in den Programmen machen und warum wir es machen, und sie dann anzupassen und zu verbessern.

Wenn Sie unser Buch »Einführung in die Programmiersprache BASIC« kennen, werden Sie die dort verwendete BASIC-Version wiedererkennen. Wir haben allerdings die PRINT@-Befehle so abgeändert, daß sie die einzelne Zahl verwenden, die der Tandy/Radio Shack und der Dragon erwarten. Eine Bildschirmkarte dieser Computer auf Seite 88 zeigt den Zusammenhang zwischen diesen Zahlen und dem Bildschirm. Wenn Sie einen Bildschirm anderer Größe verwenden, werden Sie bestimmt unsere Gestaltung verändern wollen. Sie werden dabei unsere Zeichnungen der wichtigsten Bildschirmanzeigen für jedes Programm als hilfreich empfinden.

Mastermind

Sie kennen bestimmt das Spiel »Mastermind«, wenn auch eventuell unter einem anderen Namen. Es ist das Spiel, in dem Sie eine Folge von Buchstaben, Zahlen oder Farben raten müssen, die von dem anderen Spieler (in diesem Fall dem Computer) zufällig ausgewählt wurden. In unserem Spiel sind die zu ratenden Elemente die Buchstaben A bis G. Farben wären schwerer zu programmieren, und Zahlen könnten bei Nichtmathematikern Unbehagen auslösen.

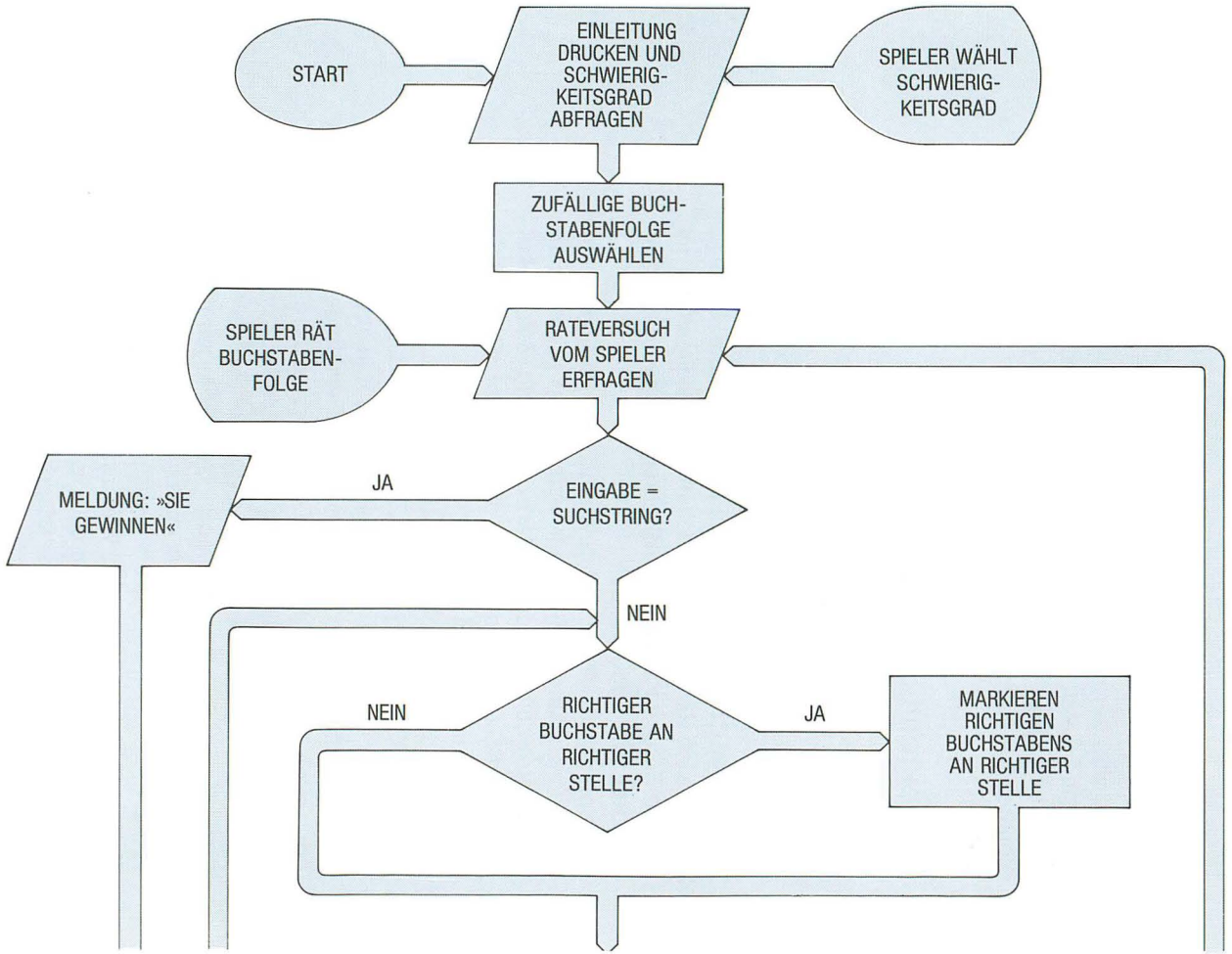
Das eigentliche Spiel

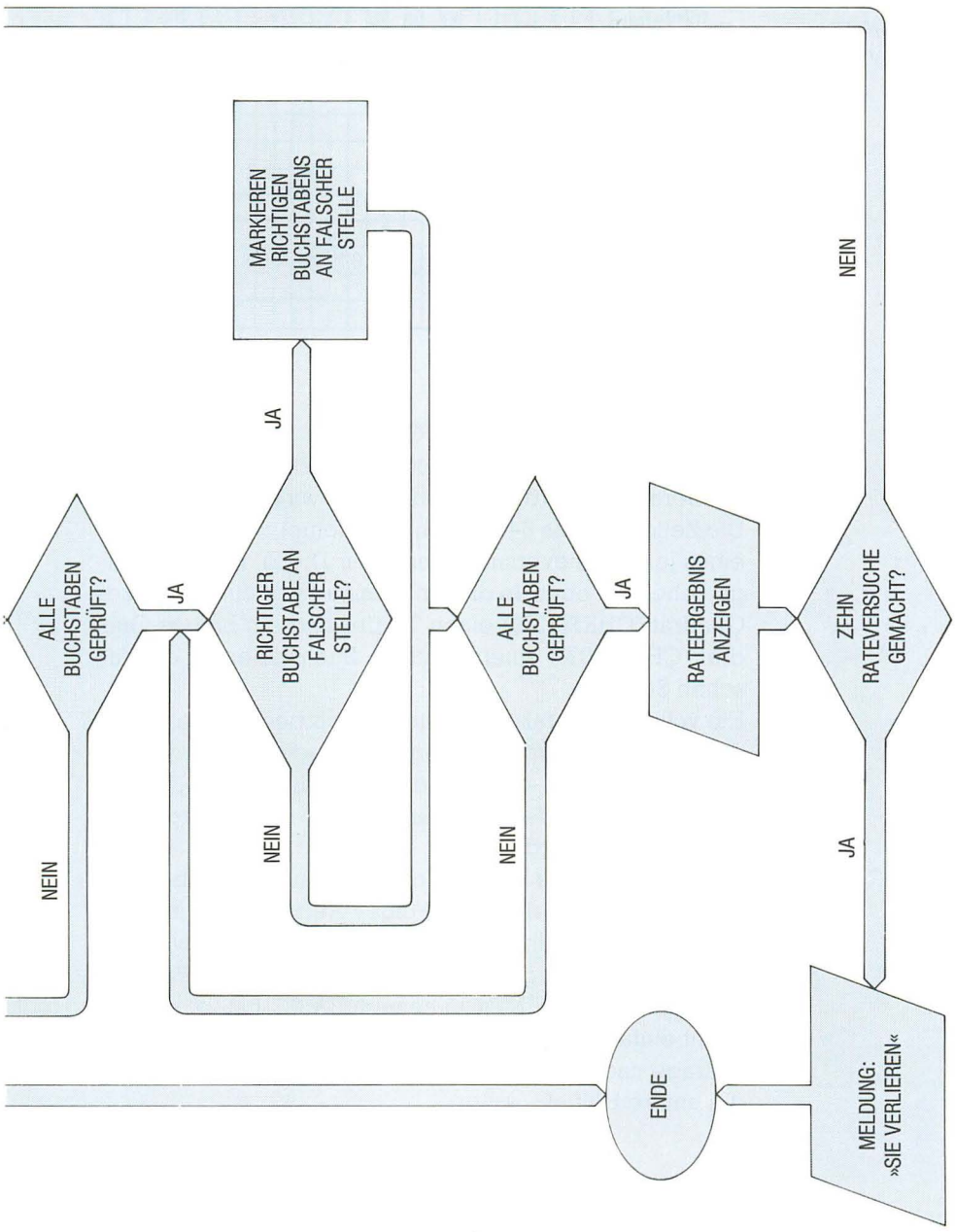
Der Computer wählt in den Zeilen 210 bis 260 eine zufällige Folge von Buchstaben aus, die der Spieler erraten muß. Sie können sehen, daß er jedesmal – so oft wie nötig – eine Zufallszahl auswählt und diese in einen Buchstaben von A bis G verwandelt, indem er den entsprechenden ASCII-Code verwendet. CHR$ (65) ist A, CHR$ (66) ist B und so weiter. Er baut sich dann aus den Zeichen eine Zeichenkette auf, die wir S$ nennen. Der nächste Programmteil wird diese Zeichenkette verändern.

Der Spieler hat genau 10 Rateversuche. Zeile 310 sorgt für 10 Wiederholungen der »Rate eine Folge«-Schleife, und bei jedem Schleifendurchlauf wird ein String (Zeichenkette) erfragt. Der Computer prüft dann (Zeilen 350 und 360), ob die richtige Anzahl Zeichen enthalten ist.

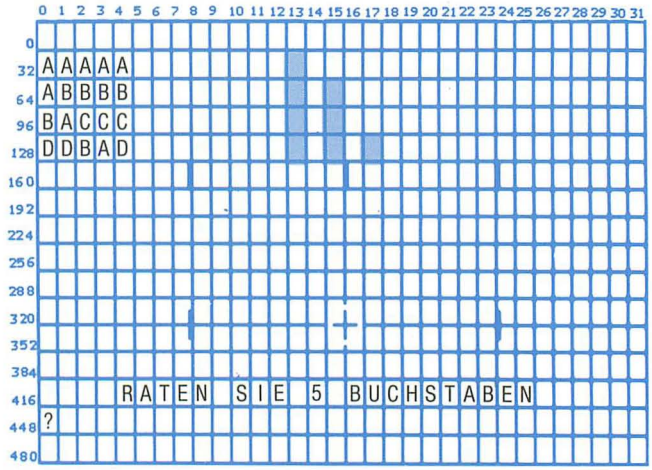
Wenn der Spieler richtig geraten hat, springt das Programm zur Gewinnsequenz auf Zeile 370. Wenn nicht, vergleicht der Computer jedes Zeichen des Ratestrings G$ mit dem entsprechenden Zeichen des Teststrings S$ (in den Zeilen 380 bis 420). Das MID$-Kommando ist eine bequeme Funktion für diese Aufgabe. Wenn Ihr Computer diesen Befehl nicht kennt, gibt es bestimmt eine andere Methode zur Stringbehandlung, oder Sie legen die Zeichen in eine einfache Matrix anstelle eines Strings.

Wenn die beiden Strings übereinstimmen, teilt der Computer das dem Spieler mit. Wenn nicht, überprüft er im nächsten Teil (in den Zeilen 430 bis 470), ob der richtige Buchstabe eventuell an einer falschen Stelle genannt wurde. Die Matrix C dient dazu, ein Zeichen als passend zu markieren und dafür zu sorgen, daß die beiden Prüfungen nicht durcheinandergehen. Die Matrix N wird zusammen mit der Variablen M benutzt, um die Zahl der richtig geratenen Zeichen festzuhalten.


START
EINLEITUNG DRUCKEN UND SCHWIERIG-KEITSGRAD ABFRAGEN
SPIELER WÄHLT SCHWIERIG-KEITSGRAD
ZUFÄLLIGE BUCH-STABENFOLGE AUSWÄHLEN
SPIELER RÄT BUCHSTABEN-FOLGE
RATEVERSUCH VOM SPIELER ERFRAGEN
EINGABE = SUCHSTRING?
JA
MELDUNG: »SIE GEWINNEN«
NEIN
RICHTIGER BUCHSTABE AN RICHTIGER STELLE?
NEIN
JA
MARKIEREN RICHTIGEN BUCHSTABENS AN RICHTIGER STELLE


ALLE BUCHSTABEN GEPRÜFT?
JA
NEIN
RICHTIGER BUCHSTABE AN FALSCHER STELLE?
JA
NEIN
MARKIEREN RICHTIGEN BUCHSTABENS AN FALSCHER STELLE
ALLE BUCHSTABEN GEPRÜFT?
JA
NEIN
RATEERGEBNIS ANZEIGEN
ZEHN RATEVERSUCHE GEMACHT?
NEIN
JA
MELDUNG: »SIE VERLIEREN«
ENDE

Mastermind –
Der Bildschirmaufbau
Schwarze Zeichen erscheinen auf grünem Grund. Die Blöcke rechts von den geratenen Buchstaben sind schwarz (für richtige Tips) und beige (für falsche Tips).



Die Bildschirmanzeige

Der Spieler erfährt nicht, welche Buchstaben richtig sind, nur deren Anzahl (deshalb benötigen wir außer C auch N). Die Zeilen 500 bis 540 lassen den Computer das Resultat eines jeden Rateversuchs anzeigen. Die Bildschirmanzeige sehen Sie oben. In dieser Version zeigt ein schwarzes Quadrat (CHR$(128)) einen Treffer und ein beiges Quadrat (CHR$(207)) einen richtigen Buchstaben an der falschen Stelle an.

Ein vollkommen falscher Rateversuch bedeutet ein Quadrat in der Hintergrundfarbe. Die Ratesequenz wird ebenfalls ausgegeben – versuchen Sie es mal ohne! –, so daß eine Spalte mit Rateversuchen und eine mit den Ergebnissen aufgebaut wird.

Während sich das Bild vom oberen Rand aus aufbaut, erscheint die »Rate eine neue Folge«-Aufforderung im unteren Teil. Wieder einmal haben wir die PRINT@-Anweisung benutzt, um sie an die richtige Stelle zu bringen. Wenn Ihr Computer kein PRINT@ in seinem BASIC hat, ist es vielleicht einfacher, die Anzeige umzukehren: Legen Sie die Abfrage nach oben, und POKEn Sie die Ergebnisfolge in die untere Hälfte!

Ändern des Schwierigkeitsgrades

Es ist viel schwerer, 5 Buchstaben zu raten als 4, 6 schwerer als 5. In dieser Version bezieht sich der Schwierigkeitsgrad (1, 2 oder 3) auf die Zahl der zu ratenden Buchstaben (4, 5 oder 6). Die Variable D zeigt die Wahl des Spielers an. Eine Alternative wäre es, die Zahl der Rateversuche für den Spieler zu ändern, oder den Spieler so oft raten zu

lassen, wie er benötigt. Am Schluß könnten die Spieler nach der Zahl der Rateversuche bewertet werden.

Benutzte Variablen

Schwierigkeitsgrad	D
Ratestring	S$
Matrix für Buchstaben	S$(6)
Matrix für richtige Tips	C(6)
Matrix für Druckausgabe	N(6)
Schleifenzähler:	
allgemein	A, B, X, Y
in Rateversuchen	G
Zwischenvariablen für Ratestring	X, A$
Rateversuch des Spielers	G$
Zähler für N-Matrix	M
Neuspielvariable	Q$

```
10 REM *** MASTERMIND ***
20 REM VON MARGARET NORMAN
30 CLS: PRINT "MASTERMIND"
40 PRINT "ICH DENKE MIR EINE FOLGE"
50 PRINT "DER BUCHSTABEN VON"
60 PRINT "A BIS G . SIE HABEN 10"
70 PRINT "VERSUCHE , SIE ZU RATEN ."
80 PRINT "JEDER RICHTIGE BUCHSTABE AN"
90 PRINT "RICHTIGER STELLE WIRD DURCH"
100 PRINT CHR$(128) ; "ANGEZEIGT"
110 PRINT "JEDER RICHTIGE BUCHSTABE AN"
120 PRINT "ANDERER STELLE WIRD DURCH"
130 PRINT CHR$(207) ; "ANGEZEIGT"
140 INPUT "WELCHER SCHWIERIGKEITSGRAD ( 1 , 2 ODER 3 )";D
150 IF D < 1 OR D > 3 THEN GOTO 140
160 CLS
200 DIM S$(6) : DIM C(6) : DIM N(6) : LET S$=""
210 REM RATESTRING AUFBAUEN
220 FOR A=1 TO 3+D
230 LET X=RND(7)
240 LET A$=CHR$(64+X)
250 LET S$=S$+A$
260 NEXT A
300 REM RATETEIL
310 FOR G=1 TO 10
320 LET M=0
330 FOR A=1 TO 6 : LET N(A)=143 : NEXT A
340 PRINT@ 420,"RATEN SIE ";3 + D;" BUCHSTABEN"
350 INPUT G$ : IF LEN(G$)=3+D THEN GOTO 370
360 PRINT@ 448,"    ":PRINT@ 420,"NEIN, ";3+D;" BUCHSTABEN !":GOTO 350
370 IF G$=S$ THEN GOTO 640
380 REM BUCHSTABEN PRUEFEN
390 FOR Y=1 TO 3+D
400 LET S$(Y)=MID$(S$,Y,1)
```

```
410 IF S$(Y)=MID$(G$,Y,1) THEN LET M=M+1 : LET N(M)=128 :
    LET S$(Y)=" ":LET C(Y)=1
420 NEXT Y
430 FOR Y=1 TO 3+D
440 FOR X=1 TO 3+D
450 IF S$(X)=MID$(G$,Y,1) AND C(Y)=0 THEN LET M=M+1 : LET N(M)=207 :
    LET S$(X)=" " : GOTO 470
460 NEXT X
470 NEXT Y
500 REM ERGEBNIS DRUCKEN
510 PRINT@ (32*G),G$
520 FOR A=1 TO 3+D
530 PRINT@ (32*G+10+2*A),CHR$(N(A))
540 NEXT A
550 REM NAECHSTES RATEN VORBEREITEN
560 FOR B=1 TO 3+D : LET C(B)=0 : NEXT B
570 PRINT@ 448,"   "
580 NEXT G
600 REM SPIELER VERSAGT
610 PRINT "  SIE HABEN VERSAGT"
620 PRINT "  ES WAR : ";S$
630 GOTO 650
640 PRINT "  GUT GEMACHT, ES WAS ";S$
650 INPUT "NOCH EIN SPIEL (J/N)";Q$
660 IF Q$="J" THEN RUN
670 END
```

Zeilenweise Anmerkungen zu »Mastermind«

10–130 Programmeinleitung.

140–150 Eingabe und Prüfung Schwierigkeitsgrad.

210–260 Wahl zufälliger Zahlen von 1 bis 7, Umwandlung in Zeichen A bis G.

310 Schleife für 10 Durchläufe.

320 Setzen N auf 0.

330 N-Matrix mit Code für Grün füllen (Hintergrundfarbe).

340–360 Abfragen Rateversuch, Prüfung richtiger Länge.

370 Gewinnsequenz, wenn richtig geraten.

390 Prüfungsschleife für jeden Buchstaben.

400 Matrix aufbauen für die einzelnen Buchstaben.

410 Jeden Buchstaben mit entsprechendem im Ratewort vergleichen. Wenn korrekt, N entsprechend auf Code für Schwarz ändern. N ändern; Buchstaben löschen, damit nicht bei nächster Prüfung wieder erkannt.

430–440 Doppelschleife; jeden Buchstaben mit jedem vergleichen.

450 Alle Buchstaben abwechselnd prüfen; wenn richtig, Stelle in N auf Code für Beige ändern, sonst wie oben.

510 Ergebnis oben auf den Schirm drucken (unter die vorherigen).
520–540 Entsprechende Farbquadrate auf dem Schirm ausgeben.

Starten und Beenden des Spiels

Bei diesem Spiel brauchen Sie mit Sicherheit eine Anleitung, so daß Sie eine »Titel-Folge« einbauen müssen. Sie könnten unsere einfache Version verbessern, indem Sie PRINT@ oder PRINT TAB verwenden oder aber – falls Ihr Computer das kann – farbige oder doppelt hohe Buchstaben. Der Spieler beendet die Einleitung durch die Wahl des Schwierigkeitsgrades.
Hier haben wir – wie versprochen – einen »Neuspiel-Teil« eingefügt (Zeilen 650 und 660). Dies ist die allereinfachste Möglichkeit: Nur die Antwort »J« ergibt ein neues Spiel. Es gibt viel ausgeklügeltere Alternativen, von denen wir einige in späteren Programmen verwenden werden. Beachten Sie, daß wir RUN für ein neues Spiel verwenden und nicht GOTO! Wenn Sie mit GOTO neu starten, müssen Sie alle Ihre Variablen auf Null setzen. RUN erledigt das automatisch.

Andere Verbesserungsvorschläge

Es gibt nicht allzuviele Möglichkeiten für größere Verbesserungen. Dramatische Bilder und Geräusche liegen kaum auf der Linie von MASTERMIND. Die Anziehungskraft des Spiels – und die kann sehr hoch sein – liegt in seiner Einfachheit.
Sie könnten die Farben entsprechend Ihren Vorlieben ändern: ein dramatischer schwarzer Hintergrund hebt die Trefferquadrate besonders hervor. Wenn Ihr Computer über keine Farbe verfügt, können Sie natürlich auch eine Schwarzweißversion aufbauen. 2 verschiedene Zeichen (vielleicht ein schwarzes und ein umrandetes Quadrat) könnten die beiden Typen von Treffern verdeutlichen.

Das Nimm-Spiel

Das »Nimm-Spiel« ist ein anderes einfaches Logikspiel. Weil es vom Typ her Mastermind ähnlich ist, ist es interessant, die Unterschiede zu betrachten, wenn wir zum Programmieren kommen.

Auch hier haben Sie das Spiel vielleicht schon einmal ohne Computer gespielt. Bei der Computerversion ist einer der Spieler der Computer. Er zeigt eine zufällige Anzahl von Steinen – in unserer Version in einer oder 2 Farben, abhängig vom gewählten Schwierigkeitsgrad. Die Spieler nehmen dann abwechselnd einen, zwei oder drei Steine. Das Ziel ist es, den Gegner den letzten Stein nehmen zu lassen.

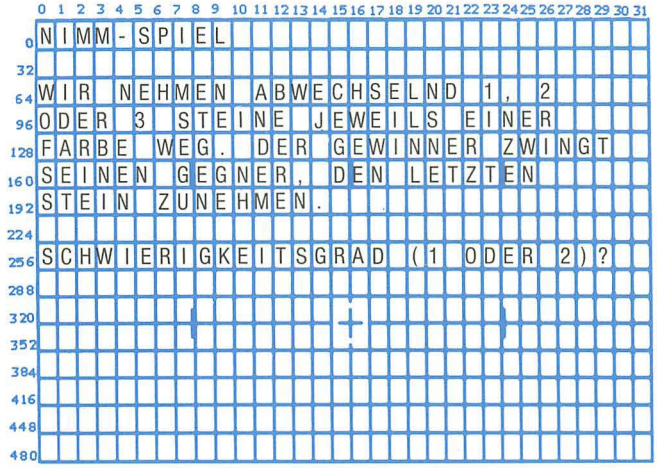
Das Spiel an sich

Der Computer stellt diesmal nicht nur die Aufgabe. Er spielt tatsächlich mit, und ein Großteil des Programms wird zur Berechnung der Züge benötigt. Weil dies ein einfaches Spiel ist, kann der Computer problemlos gute Züge machen. Er wird einen Spieler mit schlechter Strategie immer schlagen! Um seinem menschlichen Gegner eine Chance einzuräumen, erlaubt das Programm ihm – oder natürlich ihr – anzufangen.

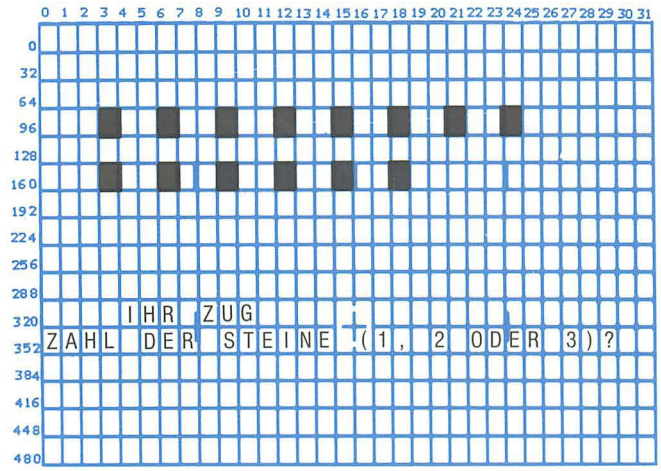
2 Variablen – NR (Anzahl der roten Steine) und NB (Anzahl der blauen Steine) – kontrollieren die zufällige Zahl (zwischen 5 und 10) von Steinen jeder Farbe. In der einfachen Stufe werden nur rote Steine benutzt. Die Zeilen 60 bis 130 ermitteln die Zahl der Steine und drucken sie auf den Schirm.

Die Zeilen 140 bis 240 sind für den Zug des menschlichen Spielers vorgesehen. Der Computer fragt nach der Zahl der wegzunehmenden Steine und nach der Farbe, sofern es eine Auswahl gibt. Die Zahl geht in die Variable N, die Farbe in die Variable C$. Der Computer überprüft, ob die Eingaben sinnvoll sind, und springt dann zu einem Unterprogramm, um die Steine auf dem Schirm zu löschen. Dies ist eine ziemlich umfangreiche Aufgabe und muß für den Zug des Computers und des Spielers ausgeführt werden; so ergibt sich der sinnvolle Einsatz eines Unterprogramms. Es gibt eine weitere Prüfung in dem Unterprogramm. Angenommen, der Spieler möchte 3 rote Steine wegnehmen, und es sind nur noch 2 übrig: Dann wird die Fehlermarke »E« gesetzt, und das Ganze geht von vorne los.

Die Zeilen 250 bis 390 erledigen die Züge des Computers. Es wird 1 zur Zahl der Steine addiert und das Ergebnis in M gespeichert. Auch hier ist die Verarbeitung von 1 aufwärts ein leichter Fall für den ON... GOTO-Befehl. Falls keine Steine mehr übrig sind, hat M den Wert 1. Von Zeile 280 geht das Programm zu Zeile 290 und meldet einen Gewinn. Wenn der Computer noch nicht gewonnen hat, berechnet er einen guten Zug, und danach springt das Unterprogramm zum Ausführen an. Achten Sie darauf, daß diese Zeilen den aktuellen Zug nicht auf dem Schirm ausgeben!



Nimm-Spiel – Einleitung
Beachten Sie, daß durch den Aufbau der PRINT-Befehle am Zeilenende kein Wort getrennt wird. Diese Seite wird gelöscht, sobald der Spieler den Schwierigkeitsgrad gewählt hat.



Nimm-Spiel –
Der Bildschirmaufbau
Die Steine sind rot und blau, der Hintergrund ist grün.

Dies macht das Unterprogramm für beide Spieler (in Zeile 1510). Vielleicht möchten Sie an dieser Stelle die Gewinnstrategie des Computers erkunden. Wenn Sie es schaffen und sie nachahmen, werden Sie selbst regelmäßig gewinnen!
Abschließend wird überprüft, ob noch ein Stein übrig ist. Wenn nicht, wird der Spieler zu seinem Sieg beglückwünscht.
Eine Schwäche des Programms in seiner jetzigen Form ist, daß es bis zum bitteren Ende durchlaufen muß. Wenn nur noch 1 Stein auf dem Spielfeld übrig ist, gibt der Computer nicht sofort auf; wenn der Gegenspieler am Zug ist und nur noch 1 Stein übrig ist, muß der Spieler diesen Stein entfernen, um gesagt zu bekommen, daß er verloren hat. Eine offensichtliche Verbesserung wäre es, diesen Schlußzug entfallen zu lassen.

Die Bildschirmanzeige

Wie auch »Mastermind« hat dieses Spiel die Steine oben im Bild und die Abfragen unten. Falls Sie das auf einem anderen Computer als ungünstig zu programmieren empfinden, kann es leicht umgekehrt werden.

Wir haben wieder einmal grafische Zeichen für die beiden Reihen farbiger Steine benutzt. SET und RESET für Pixels wären eine andere Möglichkeit, aber die Zeichen ergeben eine deutlichere Anzeige. Es gibt in dem von uns verwendeten Zeichensatz keine runden Zeichen. Falls aber Ihr Computer solche Zeichen hat, würden sie das Spielfeld realistischer gestalten.

Weil bei unserem Computer der Text immer auf einem grünen Hintergrund erscheint, haben wir wieder einmal unseren Bildschirm grün geschaltet. Wir wissen, daß das nicht jedermanns Sache ist. Schwarz, Weiß oder Beige sehen alle genauso gut (oder gar besser) aus. Grüne Quadrate löschen die roten oder blauen, sobald die Steine entfernt werden. Die seltsame Zeile 1540 erledigt das elegant, wobei immer der Stein entfernt wird, der sich am weitesten rechts befindet. Eine Alternative wäre es, den Schirm zu löschen und dann die verbleibenden Steine neu zu zeichnen.

Wenn Sie ein Textfenster definieren können, erleichtert das die Sache. In etwa die unteren 6 Zeilen könnten ein Fenster bilden, in dem die Abfragen langsam hochrollen, während der Rest des Schirms für die Grafik benutzt wird. Sie müßten hier Grafikkommandos benutzen und keine Druckbefehle, um Ihre Steine zu zeichnen!

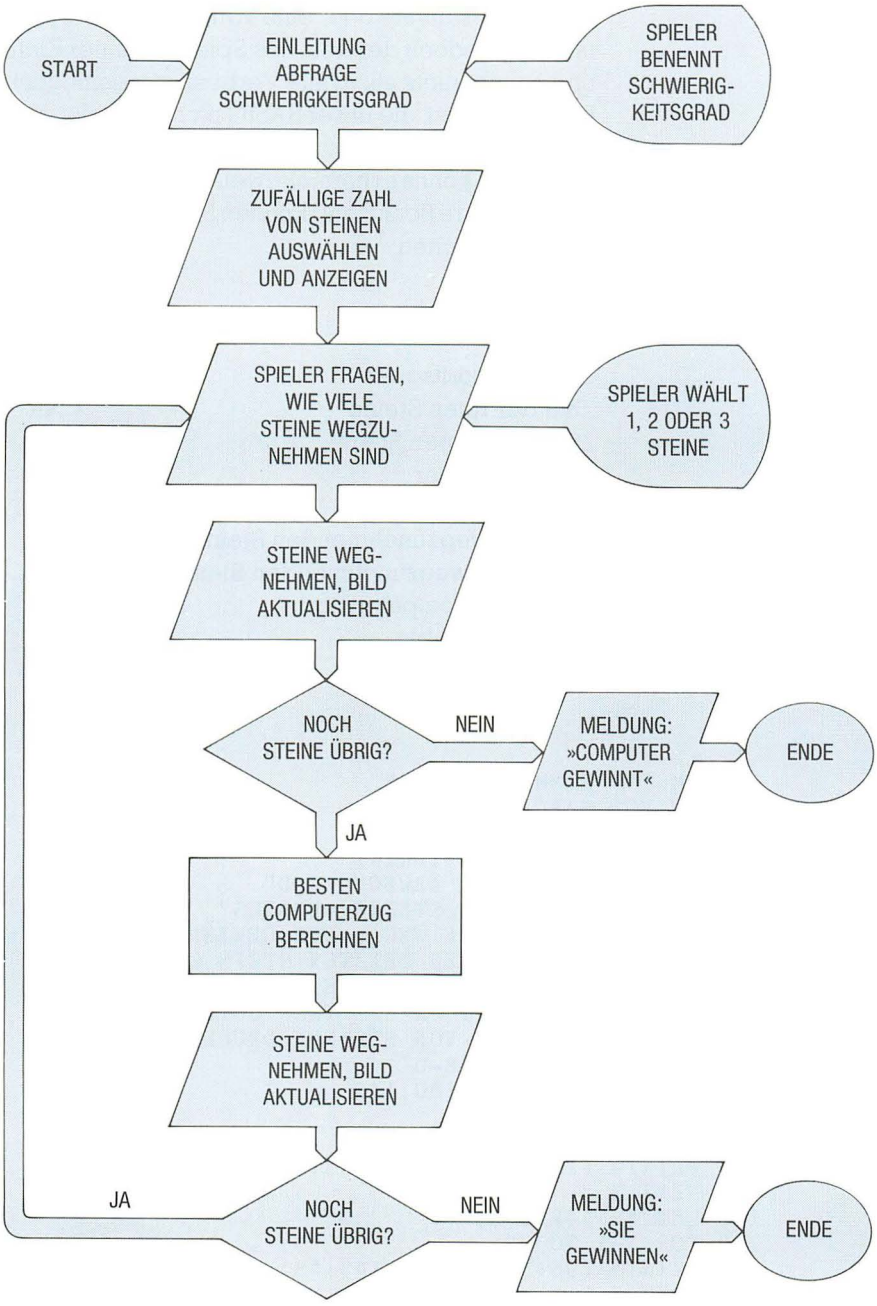
Ändern der Schwierigkeit

Mehr Steine jeder Farbe würden die Schwierigkeit nicht wirklich erhöhen, da es erst interessant wird, wenn nur noch wenige Steine übrig sind. Eine gute Möglichkeit wäre dagegen die Einführung einer dritten Reihe von Steinen. Das läßt sich einfach bewerkstelligen.

Anfang und Ende des Programms

Auch hier könnte der sehr einfache Startbildschirm mit den Möglichkeiten Ihres Computers »aufgemöbelt« werden. Der Teil für ein neues Spiel ist diesmal ein wenig umfangreicher: Jede Antwort, die mit einem »J« beginnt, bewirkt ein Neuspiel. Unser Computer verfügt nicht über Kleinbuchstaben; wenn Ihrer diese Möglichkeit bietet, könnten Sie statt dessen folgende Version verwenden:

```
IF LEFT$(G$,1) = y OR LEFT$(G$,1) = j
```


START
EINLEITUNG
ABFRAGE
SCHWIERIGKEITSGRAD
SPIELER
BENENNT
SCHWIERIG-
KEITSGRAD
ZUFÄLLIGE ZAHL
VON STEINEN
AUSWÄHLEN
UND ANZEIGEN
SPIELER FRAGEN,
WIE VIELE
STEINE WEGZU-
NEHMEN SIND
SPIELER WÄHLT
1, 2 ODER 3
STEINE
STEINE WEG-
NEHMEN, BILD
AKTUALISIEREN
NOCH
STEINE ÜBRIG?
NEIN
MELDUNG:
»COMPUTER
GEWINNT«
ENDE
JA
BESTEN
COMPUTERZUG
BERECHNEN
STEINE WEG-
NEHMEN, BILD
AKTUALISIEREN
JA
NOCH
STEINE ÜBRIG?
NEIN
MELDUNG:
»SIE
GEWINNEN«
ENDE

Andere Verbesserungsvorschläge

In der jetzigen Form gibt es keine Toneffekte im Programm, aber man könnte etwa ein »Ping« hinzufügen (jedesmal, wenn der Computer einen Stein löscht) oder auch eine Siegesfanfare oder eine »Untergangsmelodie«. Auch hier liegt jedoch der Reiz des Spiels in seiner Einfachheit, und es gibt nicht allzuviele Verbesserungsmöglichkeiten. Eine Fähigkeit, die unser BASIC nicht aufweist, sind Ganzzahlvariablen; das sind Variablen, die nur ganze Zahlen speichern können (normalerweise endet deren Name mit %). Durch ihre Benutzung können Sie das Programm fehlersicherer machen.

Benutzte Variablen

Schwierigkeitsgrad	L
Zahl der roten Steine	NR
Zahl der blauen Steine	NB
Schleifenzähler	X
Fehlermarke	E
Zahl der wegzunehmenden Steine	N
Farbe der wegzunehmenden Steine	C$
Zähler für Computerzug	M
Neuspielvariable	G$
Verzögerungsvariable	D

```
10 REM *** NIMM-SPIEL ***
21 REM VON MARGARET NORMAN
30 CLS
40 PRINT "NIMM-SPIEL":PRINT
50 PRINT "WIR NEHMEN ABWECHSELND"
60 PRINT "1,2 ODER 3 STEINE JEWEILS"
70 PRINT "EINER FARBE WEG . DER GEWINNER ZWINGT SEINEN"
80 PRINT "GEGNER , DEN LETZTEN STEIN ZU NEHMEN ."
90 PRINT : PRINT "SCHWIERIGKEITSGRAD ( 1 ODER 2 ) ?"
100 INPUT L : IF L<1 OR L>2 THEN GOTO 90
110 REM ZUFALLSZAHL VON STEINEN DRUCKEN
120 LET NR=0 : LET NB=0
130 CLS : ON L GOTO 180,140
140 LET NB=4 + RND(6)
150 FOR X=1 TO NB
160 PRINT@ (160+3*X),CHR$(143+32)
170 NEXT X
180 LET NR=4+RND(6)
190 FOR X=1 TO NR
200 PRINT@ (96+3*X),CHR$(143+48)
210 NEXT X
220 REM ZUG DES SPIELERS
230 LET E=0
240 PRINT@ 325,"IHR ZUG "
250 PRINT@ 352,"ZAHL DER STEINE (1,2,ODER 3) ?"
260 INPUT N : IF N<1 OR N>3 THEN GOTO 250
```

```
270 ON L GOTO 280,290
280 LET C$="R" : GOTO 310
290 PRINT@ 352,"FARBE DER STEINE ( R-OT ODER B-LAU ) ?"
300 INPUT C$ : IF C$<>"R" AND C$<>"B" THEN GOTO 290
310 PRINT@ 320,"    ":PRINT
320 PRINT@ 325, "IHR ZUG "
330 GOSUB 1500:REM STEINE LOESCHEN
340 IF E=1 THEN GOTO 230
350 REM ZUG DES COMPUTERS
360 LET M=NR+OB+1
370 IF M<=5 THEN GOTO 390
380 LET M=M-4 : GOTO 370
390 ON M GOTO 400,410,430,450,460
400 CLS:PRINT@ 448,"ICH GEWINNE":GOTO 540
410 LET N=1 : IF NB=0 THEN GOTO 420 ELSE LET C$="B" : GOTO 380
420 LET C$="R" : GOTO 500
430 LET N=1 : IF NB=0 OR NB=4 OR NB=8 THEN LET C$="R" : GOTO 500
440 IF NB=1 OR NB=5 OR NB=9 THEN LET C$="R" : GOTO 500 ELSE
    LET C$="B" : GOTO 500
450 LET N=2 : IF NC=0 OR NB=1 OR NC=4 OR NB=5 OR NC=8 OR NB=9
    THEN LET C$="R"!: GOTO 500 ELSE!LET C$="B" : GOUO 500
460 LET N=3 : IF NB=0 OR NB=1 OR NB=2 OR NB=5 OR NB=9 GOTO 470 ELSE
    LET C$="B" : GOTO 500
470 LET C$="R" : IF NR=2 THEN LET N=1
500 PRINT@ 325,"MEIN ZUG  "
510 GOSUB 1500:REM STEINE LOESCHEN
520 IF NR+NB <> 0 THEN GOTO 230
530 CLS:PRINT@ 448,"SIE GEWINNEN"
540 PRINT@ 460,"NOCH EIN SPIEL ?"
550 INPUT G$:IF LEFT$(G$,1)="J" THEN GOTO 90
560 END
1500 REM UNTERPROGRAMM ZUR STEINE-ENTFERNUNG
1510 PRINT@ 340,N;C$
1520 FOR D=1 TO 1000 : NEXT D
1530 IF C$<>"R" THEN GOTO 1590
1540 IF NR<N THEN GOTO 1640
1550 FOR X=1 TO N
1560 PRINT@ 99+3*(NR-X),CHR$(143)
1570 NEXT X
1580 LET NR=NR-N : GOTO 1670
1590 IF NB<N THEN GOTO 1640
1600 FOR X=1 TO N
1610 PRINT@ 163+3*(NB-X),CHR$(143)
1620 NEXT X
1630 LET NB=NB-N : GOTO 1670
1640 PRINT@ 325,"NICHT GENUG STEINE"
1650 FOR D=1 TO 1000 : NEXT D
1660 LET E=1
1670 RETURN
```

Zeilenweise Anmerkungen zum »Nimm-Spiel«

10– 80	Programm- und Bildschirmeinleitung.
90–100	Abfrage und Prüfung Schwierigkeitsgrad.
110–210	Zufällige Auswahl und Prüfung der Steinanzahl.
130	Keine zweite Reihe bei Schwierigkeitsgrad 1.
220–300	Abfrage des Spielerzugs.
230	Setzen der Fehlermarke auf 0.
270–300	Setzen der Steinfarbe auf Rot, wenn Stufe 1. Abfrage Farbe, wenn Stufe 2.
310–320	Schirmausgabe.
340	Wiederholung bei ungültiger Eingabe.
350–500	Berechnen des Computerzuges.
360	Arbeitsvariable.
370–380	Berechnen der Endposition.
390	Alternative Züge für verschiedene Endpositionen.
400	Ergebnis, wenn Spieler letzten Stein nimmt.
520	Nächster Zug des Spielers, wenn nicht letzter Stein.
530–560	Spielende, Frage nach Neuspiel.
550	Jede Antwort mit »J« oder »j« gibt neues Spiel.
1510	Ausgabe gewählter Zug auf Bildschirm.
1540, 1590	Fehlerteil, wenn Spieler mehr Steine als möglich gewählt.
1560, 1610	Zeichen in Hintergrundfarbe über Steine drukken.
1580, 1630	Ändern der Variablen für restliche Steine.
1660	Fehlermarke setzen bei ungültigem Zug.

Orgelspieler

Wenn Ihr Computer Töne und Klänge erzeugen kann, können Sie ihn per Programm als einfaches Musikinstrument agieren lassen. Sie schreiben ein Programm, so daß durch Drücken bestimmter Tasten bestimmte Noten erklingen. Hört sich das leicht an? In der Praxis kann das recht schwierig sein. Wir wollen einmal diskutieren, wie es gehen könnte.

Das grundlegende Kommando – wie bei so vielen Spielen – ist INKEY$ oder das entsprechende Äquivalent Ihres Computers. Über die Benutzung von INKEY$ erhält der

Computer einen Wert für beispielsweise A$. Dann prüft er, was dieser Wert bewirken soll – hier etwa eine Note zu spielen, die Oktave oder eine andere Eigenschaft der zu spielenden Note zu ändern (Länge, Lautstärke). Damit erhalten wir ungefähr folgende Programmsequenz:

```
100  LET A$ = INKEY$
110  IF A$ = "C" THEN PLAY .5, 50
```

wobei .5 die Länge der Note und 50 deren Wert darstellt. Unterschiedliche Computer stimmen mit der Reihenfolge dieser Werte oft nicht überein, wie schon gesehen.
Das geht ganz gut, wenn nur ein Wert von A$ geprüft werden muß, aber es wird lästig, wenn man vielleicht 15 oder mehr Tasten hat. Anstatt eine überlange Befehlsfolge von »IF A$ = ...«-Befehlen in Ihr Programm zu legen, wäre es schöner, eine Tabelle mit Notenwerten in einer Matrix zu haben, die zu Beginn mit DATAs gefüllt wird. Die Nummer des Tabellenelements könnte mit dem ASCII-Code der gedrückten Taste verknüpft werden, so daß der Befehl etwa so aussehen würde:

```
110  IF ASC(A$) > 65 AND (A$) < 77 THEN PLAY
     0.5,N(ASC(A$)-65)
```

ASC ist eine BASIC-Funktion, die den ASCII-Wert eines Zeichens liefert. Aber nicht alle BASIC-Versionen verfügen darüber. Unsere Zeile nimmt an, daß Sie die ASCII-Werte zwischen 65 und 77 verwenden – also die Großbuchstaben von A bis M. Die Matrix N enthält dann die Notenwerte, so daß, wenn C (ASCII-Code 67) die Taste für die Note 99 wäre, das Element 67–65, also 2, den Wert 99 enthält.
Wir machen das in unserem Beispielprogramm nicht, weil die ASCII-Werte der von uns benutzten Tasten viel zu weit auseinanderliegen, so daß es sinnlos ist. Trotzdem ist es eine Technik, die nicht vergessen werden sollte, wenn

Orgeltastatur
Hier sind unsere 13 Tasten. Der erste Teil unseres Programms wählt die Farben der Tasten und des Hintergrundes und zeichnet dann die Umrisse auf den Bildschirm. Der Rest ist der Spielteil. Wie echt die Noten klingen, hängt von Ihrer Maschine ab.

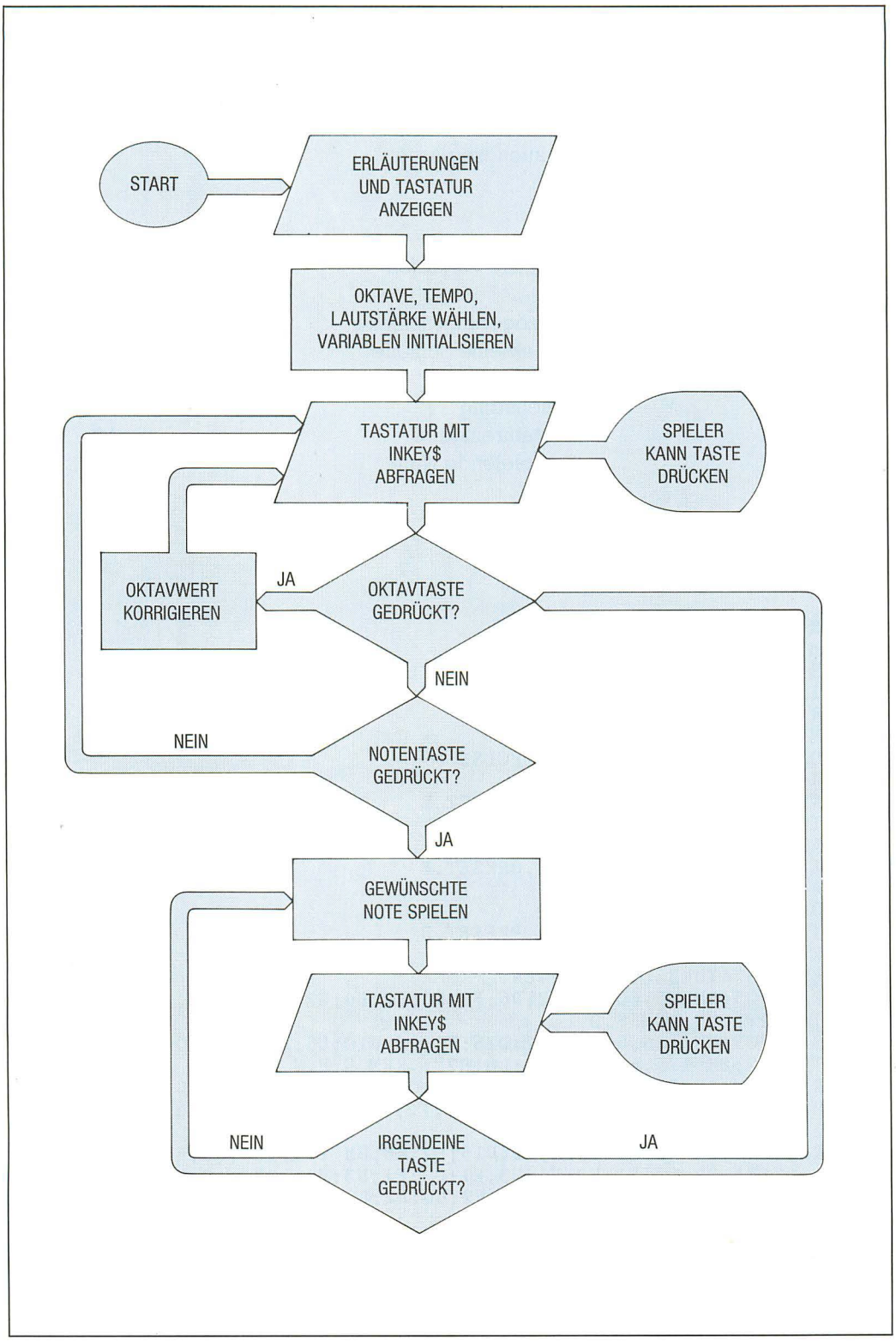
wenig Speicher zur Verfügung steht, oder so gut wie alle Tasten der Tastatur für verschiedene Töne benutzt werden sollen.
Wir haben uns entschieden, 13 Tasten zu verwenden, um eine komplette Oktave zu spielen, wobei C am unteren und am oberen Ende vorhanden ist. Die Tasten in der unteren Reihe der Tastatur entsprechen C-Dur; darüber befinden sich die erhöhten oder erniedrigten Töne. Der Aufbau ähnelt dem einer Klaviertastatur. Nichts passiert, wenn Sie eine falsche Taste drücken (außer einer Pause in der Melodie), aber zur Erinnerung haben wir einen Programmteil eingebaut, um das »Keyboard« auf den Bildschirm zu zeichnen. Dazu muß noch etwas gesagt werden.
In unserem BASIC werden die Noten durch Buchstaben bezeichnet und die Oktaven durch die Zahlen von 1 bis 5. Vielleicht müssen Sie C # und die anderen Notennamen so umwandeln, daß Ihr Computer sie erkennt. Außerdem haben Sie eventuell PLAY durch BEEP oder SOUND zu ersetzen und die Syntax entsprechend anzupassen.
Wir haben die Zahlen 1 bis 5 auf der Tastatur ausgewählt, um die Oktave zu bestimmen. In unserem BASIC ist der Standardwert dafür 3, falls noch nichts ausgewählt wurde. Wenn dies bei Ihrem BASIC anders ist, sollten Sie einen Befehl wie LET O=2 (O natürlich für Oktave) am Programmanfang einbauen. (Testen Sie verschiedene Werte, um zu sehen, welcher am besten paßt!) Wenn es keinen besonderen Oktavparameter in Ihrem BASIC gibt, wollen Sie bestimmt die Noten- und Oktavwerte so anpassen, daß die Notenwerte angemessen erhöht oder erniedrigt werden. Wenn es etwa 48 Noten in einer Oktave gibt, so daß eine Oktave bei Note 53 beginnt und die nächste bei 53 + 48, dann könnten Sie folgende Kommandofolge einbauen:

```
IF A$ = "2" THEN LET O = 5 + (2*48)
IF A$ = "3" THEN LET O = 5 + (3*48)
```

und das »Spiel einen Ton«-Kommando so aufbauen

```
IF A$ = "G" THEN SOUND (Länge, nicht Wert + O) . . .
```

Beachten Sie, daß das Drücken der »,«-Taste in Oktave 5 außerhalb des Tonbereiches liegt und einen »Programmabsturz« hervorruft!
Mit dem VAL-Kommando, falls vorhanden, könnten Sie die ganzen Oktavauswahlkommandos in ein einziges komplexes stopfen. Der etwas ungewöhnliche INSTR-Ausdruck in Zeile 260 (der den String A$ innerhalb von O$ sucht) ist


START
ERLÄUTERUNGEN UND TASTATUR ANZEIGEN
OKTAVE, TEMPO, LAUTSTÄRKE WÄHLEN, VARIABLEN INITIALISIEREN
TASTATUR MIT INKEY$ ABFRAGEN
SPIELER KANN TASTE DRÜCKEN
OKTAVWERT KORRIGIEREN
JA
OKTAVTASTE GEDRÜCKT?
NEIN
NEIN
NOTENTASTE GEDRÜCKT?
JA
GEWÜNSCHTE NOTE SPIELEN
TASTATUR MIT INKEY$ ABFRAGEN
SPIELER KANN TASTE DRÜCKEN
NEIN
IRGENDEINE TASTE GEDRÜCKT?
JA

eine andere Alternative, die Oktavbefehle zusammenzupacken.
Wir haben INSTR in Zeile 1000 wiederverwendet, um den erneuten Test, ob keine Taste gedrückt wurde, zu beschleunigen. Wenn Ihr BASIC dies nicht kann, können Sie es auch weglassen.

Benutzte Variablen

Verzögerungsschleife	D
Allgemeine Schleife	X
Oktavstring	O$
Notenstring	B$
Tastatureingabe	A$
Zu spielende Note	N$

```
10 REM ORGELSPIELER
20 REM VON MARGARET NORMAN
30 CLS : PRINT "SPIELEN SIE DIE ORGEL MIT"
35 PRINT "DEN BEZEICHNETEN TASTEN"
40 FOR D=1 TO 2000 : NEXT D : CLS
50 REM TASTATUR ZEICHNEN
60 PMODE 4,1 : SCREEN 1,1 : PCLS5
70 FOR X=60 TO 160 STEP 20
80 LINE (X,40) - (X+18,100),PRESET,B
90 PAINT (X+10,50),0,0
100 NEXT X
110 LINE (100,40)-(118,100),QRESET,B
120 PAINT (110,50),5,5
130 FOR X=50 TO 190 STEP 20
140 LINE (X,100)-(X+20,160),PRESET,B
145 NEXT X
150 REM TASTATURBEZEICHNUNGEN
160 DRAW "C5;BM 75,60;L8;G2;D6;F2;R6;F2;D6;G2;L8;BM 85,60;D20;R8;E2;
    U16;H2;L8": REM S,D
170 DRAW "C5;BM 136,60;L8;G2;D16;F2;R6;U10;L5;BM 145,60;D20;U10;R10;
    U10;D20;BM 175,60;D18;G2;L6;H2": REM G,H,J
180 DRAW "C0;BM 55,120;R10;D2;G2;D2;G2;D2;G2;D2;G2;D2;G2;R10;
    BM 75,120;D2;F2;D2;F2;D2;F2;D2;F2;D2;F2;BL10;U2;E2;U2;E2;U2;E2;
    U2;E2;U2;E2;":REM Z,X
190 DRAW "C0;BM 105,120;L8;G2;D16;F2;R8;BM 115,120;D3;F1;D3;F1;D3;
    F1;D3;F1;D3;F1;E1;U3;E1;U3;E1;U3;E1;U3;E1;U3" : REM C,V
200 DRAW "C0;BM 135,120;D20;R8;E2;U6;H2;L8;R8;E2;U6;H2;L8;BM 175,140;
    U20;F5;E5;D20": REM B,N
210 DRAW "C0;BM 155,140;U20;F2;D2;F2;D2;F2;D2;F2;D2;F2;D2;U20;BM 197,
    130;R3;D1;L3;D1;R3;D5;L3": REM M ,
220 REM SPIELEN
230 LET O$="12345" : LET B$="ZXCVBNM,SDGHJ"
240 PLAY "T20V30"
```

```
250 LET A$=INKEY$ : IF A$="" THEN GOTO 250
260 IF INSTR(1,O$,A$)<>0 THEN PLAY "O"+A$ : GOTO 250
270 IF INSTR(1,B$,A$)=0 THEN GOTO 250
280 IF A$="Z" THEN LET N$="C" : PLAY N$ : GOTO 1000
290 IF A$="S" THEN LET N$="C#" : PLAY N$ : GOTO 1000
300 IF A$="X" THEN LET N$="D" : PLAY N$ : GOTO 1000
310 IF A$="D" THEN LET N$="D#" : PLAY N$ : GOTO 1000
320 IF A$="C" THEN LET N$="E" : PLAY N$ : GOTO 1000
330 IF A$="V" THEN LET N$="F" : PLAY N$ : GOTO 1000
340 IF A$="G" THEN LET N$="F#" : PLAY N$ : GOTO 1000
350 IF A$="B" THEN LET N$="G" : PLAY N$ : GOTO 1000
360 IF A$="H" THEN LET N$="G#" : PLAY N$ : GOTO 1000
370 IF A$="N" THEN LET N$="A" : PLAY N$ : GOTO 1000
380 IF A$="J" THEN LET N$="A#" : PLAY N$ : GOTO 1000
390 IF A$="M" THEN LET N$="B" : PLAY N$ : GOTO 1000
400 IF A$="," THEN LET N$="O+CO-" : PLAY N$ : GOTO 1000
1000 LET A$=INKEY$ : IF A$="" THEN PLAY N$ : GOTO 1000
1010 GOTO 260
```

Zeilenweise Anmerkungen zum »Orgelspieler«

10– 40	Programm- und Bildschirmeinleitung.
60	Grafikmodus, Farbauswahl, beiger Hintergrund.
70–145	Tasten auf den Schirm zeichnen.
150–210	Mit Hi-Res-Kommandos die Noten auf die Tasten schreiben.
230	Stringvariablen für Oktavzahlen, zulässige Noten.
240	Tempo 20, Lautstärke 30.
250	Irgendeine Taste gedrückt?
260	Suche der Oktavauswahl und entsprechendes Setzen.
270	Suche der Notenauswahl. Wenn nicht, zurück zu 250.
280–400	Spielen der Note (aktuelle Länge, Lautstärke, Oktave).
1000	Neue Taste gedrückt? Wenn nicht, alten Ton weiter.
1010	Wenn neue Taste, zurück und auswerten.

Richtiges Timing

Sie müssen mit verschiedenen Notenlängen experimentieren, um einen Wert zu finden, der ausreichend ist, die Zeit bis zur nächsten Prüfung von A$ zu überbrücken. In un-

serem BASIC bleiben Tempo und Lautstärke der Noten nach ihrer Festsetzung erhalten, bis zur nächsten Änderung. Wir setzen sie am Programmanfang in Zeile 240. Sie müssen vielleicht für jede individuelle Note Länge und Lautstärke angeben, etwa so:

```
130 IF A$ = "C" THEN SOUND
    (Note, Länge, Lautstärke): GOTO 1000
```

Ein interessantes Problem ist es, sich zu entscheiden, was am Ende einer Note passieren soll. Soll sie weiterklingen, bis eine – auch eine nicht zugewiesene Taste – gedrückt wird? Oder sollte das Programm pausieren, wenn die Note einmal gespielt wurde? Wir fanden es besser, wenn die Note wiederholt wird; dieser Effekt wird in Zeile 1000 erreicht. Eine gedrückte Taste erzeugt eine Art Trillereffekt. Tastaturen unterscheiden sich jedoch, und wenn Ihre Tastatur »Auto-Repeat« hat, empfinden Sie das vielleicht als unnötig. (Eventuell müssen Sie auch den Tastaturpuffer löschen, ehe Sie A$ erneut prüfen.)

Komplexere Musik

Sind Sie mittlerweile total verwirrt? Wenn es Sie beruhigt – wir haben hier nur das absolute Minimum besprochen. Vergegenwärtigen Sie sich nur einmal 3 oder 4 verschiedene Tonkanäle, Hüllkurven für verschiedene Klangarten! Vom Koordinieren des Timings der Noten, um damit Akkorde spielen zu können, gar nicht zu reden! Schauen Sie sich in Ruhe das Flußdiagramm und das Programm auf den Seiten 101–103 an – und alles sollte klar werden.

Der Bildschirmaufbau

Es ist nicht notwendig, irgend etwas auf dem Schirm anzuzeigen, außer vielleicht einer Meldung wie »Drücken Sie irgendeine Taste, um eine Note zu spielen«. Trotzdem wird das Programm reizvoller, wenn eine Anzeige auf dem Bildschirm vorhanden ist.

Wir entschlossen uns, eine einfache »Klavier-Tastatur« auf den Bildschirm zu zeichnen, wie Sie auf Seite 99 sehen können. Schwarze und beige Tasten auf beigem Hintergrund (es gibt kein Weiß in unserem BASIC) tragen die zugehörigen Bezeichnungen.

Um ein schönes Bild zu erreichen, müssen Sie die Buchstaben größer als normal zeichnen. Unsere sind mit Grafik-

kommandos für hohe Auflösung gezeichnet. Eine Alternative wäre es, mit Blockgrafik große Buchstaben aufzubauen, oder jeden Buchstaben aus verschiedenen selbstdefinierten Zeichen aufzubauen, oder, wenn Ihr Computer Ihnen das bietet, ein Kommando für doppelte Buchstabengröße einzusetzen.
Die Tastatur zu zeichnen ist eine komplizierte Angelegenheit, wie man aus dem Programm ersehen kann: Es erfordert die Zeilen von 50 bis einschließlich 210. Wenn Sie diesen Teil nicht an einen anderen Computer anpassen möchten – das könnte auch schwierig sein, wenn Ihr Computer nicht über gute Grafikkommandos verfügt –, was könnten Sie sonst auf dem Schirm darstellen? Eine einfache Alternative wäre es, jedesmal wenn eine Note gespielt wird, eine verschiedenfarbige Fläche auf dem Schirm zu zeigen oder auf dem ganzen Schirm die Farbe zu wechseln. Sie könnten bestimmte Farben für bestimmte Töne wählen oder die Auswahl jedesmal zufällig treffen.

Hangman

Jetzt aber weiter mit einem Wortspiel – dem klassischen »Hangman«! Es gibt davon verschiedene Versionen für Heimcomputer; unsere ist eine recht einfache.

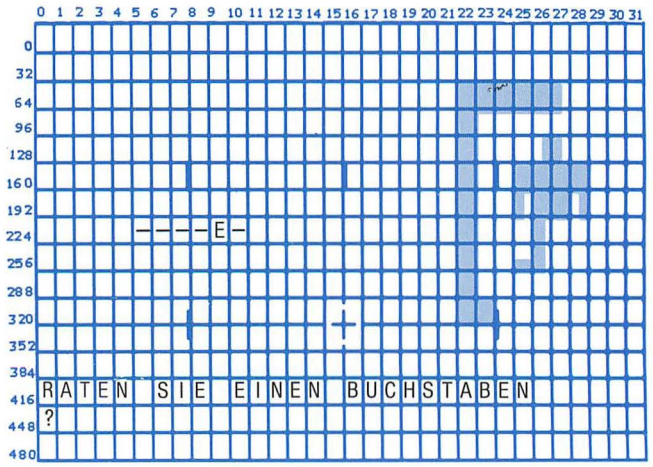
Das Spiel an sich

Das Ziel von Hangman ist es, ein Wort zu raten – entweder ein Wort eines anderen Spielers oder ein Wort des Computers. In unserer Version hat der Computer eine Liste von 10 Worten (in einer DATA-Zeile), aus denen er bei jedem Durchlauf eines auswählt. Sie können das Programm leicht so ändern, daß es statt dessen Worte per INPUT erfragt. Oder schauen Sie sich die relevanten Zeilen (70 und 80) an und tauschen die Worte aus! Wir haben eine einfache Prüfung eingebaut, damit während eines Ablaufs jedes Wort nur einmal ausgewählt wird. Wenn das Wort bereits benutzt wurde, ist die dem Wort A$ entsprechende Stelle in der Matrix A mit einer 1 markiert. Wenn WD, die Variable, die das zu ratende Wort bestimmt, mit einer Zufallszahl geladen wird, prüft der Computer, ob A (WD) auf 1 gesetzt ist; wenn das der Fall ist, wird die Operation wiederholt.

Der Spieler hat mehrfach die Möglichkeit, einen Buchstaben zu raten. Dieser wird mit den Buchstaben im Wort mittels des MID$-Kommandos verglichen. (Sie können diese Verwendung hier mit der in »Mastermind« vergleichen, wo wir etwas Ähnliches taten.) Wenn der Spieler richtig geraten hat, wird der Buchstabe auf dem Schirm ausgegeben, und der Spieler hat die Möglichkeit, das Wort zu raten. Wenn falsch geraten wurde, wird ein »hängender Mann« auf dem Bildschirm aufgebaut. Wenn dieser fertig ist, verliert der Spieler und erfährt die Antwort.
In einer Matrix A$ wird das zu ratende Wort gespeichert. Eine zweite »Minimatrix« wird zur Verarbeitung der einzelnen Rateversuche benötigt – für Blanks und bereits eingegebene Buchstaben. Jede Stelle dieser Matrix speichert nur einen einzigen Buchstaben. Die hinteren Stellen werden anfangs mit Blanks aufgefüllt und ignoriert, wenn nur ein kurzes Wort zu raten ist. Sie wundern sich vielleicht, warum wir keinen einfachen String anstelle einer Matrix benutzt haben. Die Antwort ist, daß es einfacher ist, eine Matrix zu ändern, wenn ein neuer Buchstabe eingegeben wird.
Wir benutzen eine Prüfvariable – SC –, um die Anzahl der vergeblichen Rateversuche zu speichern. Das Unterprogramm, das den hängenden Mann zeichnet, verzweigt dann mit einem ON ... GOTO-Befehl, abhängig vom Wert der Variablen SC. SC wird am Anfang des Unterprogramms angepaßt, so daß es, obwohl anfangs mit 0 besetzt, den Wert 1 hat, wenn das erste ON ... GOTO erreicht wird.

Die Bildschirmanzeige

Der Schirm ist in 3 Teile gespalten, die mit unterschiedlichen PRINT@ angesprochen werden. (Wenn Ihr Compu-



Hangman –
Der Bildschirmaufbau
Der Hintergrund ist grün, der Hangman und die Buchstaben sind schwarz.

ter diesen Befehl nicht kennt, können Sie das auch anders anordnen.) Ein Teil zeigt die Matrix B$ an (die Leerstellen und die schon geratenen Buchstaben), ein anderer die Fragen nach Buchstaben oder dem Wort, der dritte Teil schließlich zeigt das Bild des hängenden Mannes.
Unser ziemlich grob gezeichneter »gehenkter Mann« wird aus einem Block grafischer Zeichen aufgebaut. Wenn Sie auf Ihrem Computer selbst Zeichen definieren können, vermögen Sie ziemlich exotische Versionen zu erzeugen. Wenn der Spieler verliert, wird der Schirm gelöscht und eine passende Meldung gedruckt.

Anfang und Ende des Spieles

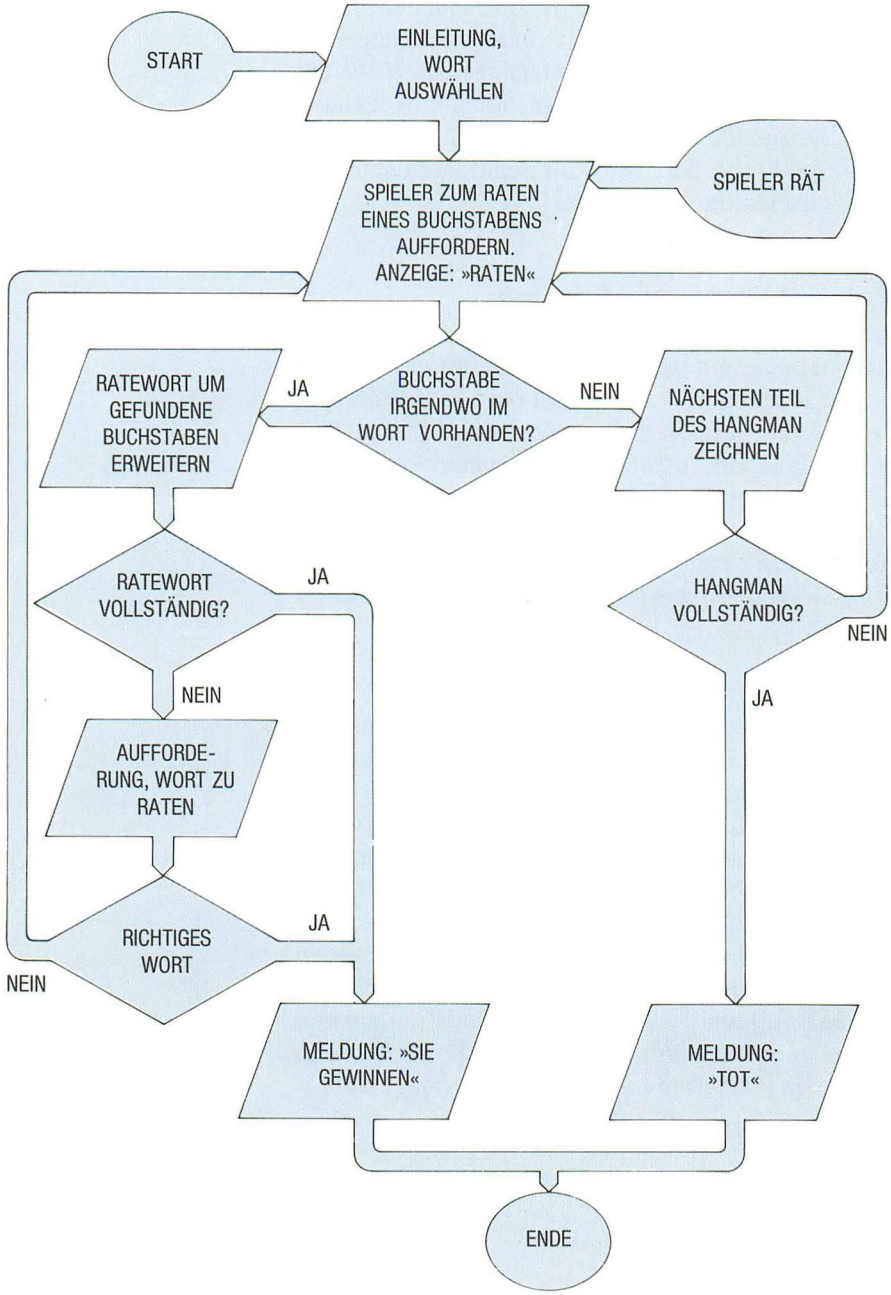
Die übliche Anzeige der Spielregeln leitet das Spiel ein, die Frage nach einem neuen Spiel beendet es. Diesmal starten wir das Spiel nicht neu, da wir ein neues Wort zum Raten aussuchen wollen, indem wir die Variable WD ändern, die das steuert. Zeile 840 übernimmt die nötige Verwaltung und setzt die Prüfvariablen vor einem neuen Spiel zurück.
Wir haben keine Punktewertung für den Spieler eingebaut, aber Sie können das noch tun, wenn Sie wollen.

Verbesserung des Spiels

Einigen kommerziellen Versionen liegen rastlose Anstrengungen zugrunde, das Spiel zu beleben. Es gibt sogar eine mit einer kompletten Schützenschwadron anstelle eines Galgens. Ausgefeiltheit ist jedoch nicht unbedingt der Kernpunkt des Spiels, und eine einfache Version funktioniert genauso gut. Sie haben vielleicht Spaß daran, ein paar kurze Töne für richtige und falsche Tips hinzuzufügen und möglicherweise eine längere Melodie (vielleicht einen passenden Grabgesang), wenn der Spieler verliert. Eine weitere Verbesserung wäre es, die bereits versuchten Buchstaben in einer Zeile in einem freien Schirmbereich auszugeben, mit unserem Layout eventuell ganz oben.

Benutzte Variablen

Matrix für Ratewort	A$(10)
Matrix für benutzte Worte	A(10)
Matrix geratener Buchstaben	B$(12)
Leben verloren/nicht verloren	LI
Anzahl verlorener Leben	SC
Spiel verloren	D
Schleifenzähler	X


START
EINLEITUNG, WORT AUSWÄHLEN
SPIELER RÄT
SPIELER ZUM RATEN EINES BUCHSTABENS AUFFORDERN. ANZEIGE: »RATEN«
BUCHSTABE IRGENDWO IM WORT VORHANDEN?
JA
NEIN
RATEWORT UM GEFUNDENE BUCHSTABEN ERWEITERN
NÄCHSTEN TEIL DES HANGMAN ZEICHNEN
RATEWORT VOLLSTÄNDIG?
JA
NEIN
HANGMAN VOLLSTÄNDIG?
NEIN
JA
AUFFORDE-RUNG, WORT ZU RATEN
RICHTIGES WORT
JA
NEIN
MELDUNG: »SIE GEWINNEN«
MELDUNG: »TOT«
ENDE

Dummy-INKEY$ nach Einleitung	X$
Länge des Wortes	W
Verzögerungsschleife	DL
Geratener Buchstabe	L$
Geratenes Wort	G$
Wort vollständig	CH
Neuspielvariable	R$

```
10 REM *** HANGMAN ***
20 REM VON SUSAN CURRAN
30 DIM A$(10):DIM A(10):DIM B$(12)
40 FOR X=1 TO 10
50 READ A$(X) : LET A(X)=0
60 NEXT X
70 DATA BRAUN,SCHIRM,ZEUGNIS,SCHALTUNG,ZUSTAND
80 DATA BREITE,KOENIGIN,ANGRIFF,ZENITH,LEERE
90 LET WD=RND(10) : LET A(WD)=1
100 LET LI=0 : LET SC=0 : LET D=0
110 CLS
120 PRINT : PRINT "    HANGMAN":PRINT
130 PRINT "VERSUCHEN SIE DAS WORT DURCH"
140 PRINT "ANGABE VON BUCHSTABEN ZU"
150 PRINT "RATEN . BEI FEHLERN BEGINNEN"
160 PRINT "WIR , SIE ZU HAENGEN !"
170 PRINT "NACH RICHTIGEN TIPS DUERFEN SIE"
180 PRINT "DAS WORT RATEN . HIER VERLIEREN"
190 PRINT "SIE KEINE LEBEN ."
210 PRINT "DRUECKEN SIE EINE TASTE"
220 LET X$=INKEY$ : IF X$="" THEN GOTO 220
300 CLS : REM SPIEL BEGINNT
320 LET W=LEN(A$(WD))
330 FOR X=1 TO W
340 B$(X)="-"
350 NEXT X
390 FOR X=1 TO W
400 PRINT@ (228+X),B$(X)
410 NEXT X
420 FOR DL=0 TO 200 : NEXT DL
430 PRINT@ 416, "RATEN SIE EINEN BUCHSTABEN";
440 INPUT L$:PRINT@ 416,STRING$(18," ");:PRINT "  ";
500 REM BUCHSTABEN PRUEFEN
510 FOR X=1 TO W
520 IF L$=MID$(A$(WD),X,1) THEN GOSUB 1500
530 NEXT X
540 IF LI=0 THEN GOSUB 1000
550 IF D=1 THEN GOTO 810
560 IF LI=0 THEN GOTO 420
570 LET LI=0
600 REM WORT VOLLSTAENDIG ?
610 LET CH=0
620 FOR X=1 TO W
630 IF MID$(A$(WD),X,1)<>B$(X) THEN LET CH=1
640 NEXT X
650 IF CH=0 THEN GOTO 800
660 REM WORT DARF GERATEN WERDEN
670 PRINT@ 416,"RATEN SIE JETZT DAS WORT";
```

```
680 INPUT G$:
690 IF G$=A$(WD) THEN GOTO 800
700 PRINT@ 416,"SCHADE , VERKEHRT";
710 FOR DL=0 TO 500 : NEXT DL
720 PRINT@ 416,STRING$(18," ");:PRINT STRING$(14," ");
730 PRINT STRING$(19," ");
740 GOTO 420
800 CLS:PRINT@ 320,"RICHTIG ; GUT GEMACHT !"
810 INPUT "NOCHMAL (J/N) ?";R$
820 IF LEFT$(R$,1)<>"J" THEN END
830 LET WD=RND(10) : IF A(WD)=1 THEN GOTO 830
840 LET LI=0:LET SC=0:LET D=0:LET A(WD)=1
1000 REM WORT NICHT ERRATEN
1010 PRINT@ 416,"SCHADE , VERKEHRT";
1020 FOR DL=0 TO 500 : NEXT DL
1040 LET SC=SC + 1
1050 ON SC GOTO 1060,1070,1080,1090,1100,1110,1120,1130,1140,1150
1060 PRINT@ 342,CHR$(128)+CHR$(128);:RETURN
1070 FOR N=2 TO 9:PRINT@ (32*N+22),CHR$(128);:NEXT N:RETURN
1080 PRINT@ 87,CHR$(128)+CHR$(128)+CHR$(128)+CHR$(128)+CHR$(133);:RETURN
1090 PRINT@ 154,CHR$(138)+CHR$(133);:RETURN
1100 PRINT@ 186,CHR$(128)+CHR$(128);:PRINT@ 218,CHR$(128)+CHR$(128);:RETURN
1110 PRINT@ 185,CHR$(128);:PRINT@ 217,CHR$(133);:RETURN
1120 PRINT@ 188,CHR$(128);:PRINT@ 220,CHR$(138);:RETURN
1130 PRINT@ 250,CHR$(133);:PRINT@ 281,CHR$(140)+CHR$(133);:RETURN
1140 PRINT@ 251,CHR$(138);:PRINT@ 283,CHR$(138)+CHR$(140);:RETURN
1150 CLS:PRINT@ 320,"KLONG ! SIE SIND TOT !"
1160 PRINT "DIE ANTWORT WAR ";A$(WD)
1170 LET D=1
1180 RETURN
1500 REM BUCHSTABE PASST
1510 LET LI=1 : LET B$(X)=L$
1520 FOR X=1 TO W
1530 PRINT@ (228+X),B$(X);
1540 NEXT X
1550 RETURN
```

Zeilenweise Anmerkungen zu »Hangman«

30– 100 Matrizen und Variablen initialisieren.

110– 220 Einleitung auf dem Schirm.

320– 410 Drucken einer Reihe von Strichen, um Wortlänge anzuzeigen.

440 Löschen von Meldung und geratenen Buchstaben.

500– 530 Alle Buchstaben des Wortes prüfen.

540	Wenn nichts paßt, Aufruf »Hangman«-Unterprogramm.
550	Wenn Mann vollständig, Teil für neues Spiel.
560	Wenn nichts paßt, nächsten Buchstaben raten.
600– 640	Wenn paßt, Prüfen des Wortes auf Vollständigkeit.
650	Wenn Wort vollständig, Teil für »Gut gemacht«.
660– 700	Wenn paßt und Wort nicht komplett, Wort raten.
720– 730	Löschen des geratenen Wortes und Meldung.
810– 850	Neuspiel bzw. Programmende.
1000–1180	»Hangman« zeichnen wie folgt.
1060	Unteres Gerüst.
1070	Mittleres Gerüst.
1080	Oberes Gerüst.
1090	Kopf.
1100	Körper.
1110–1140	Linker Arm, rechter Arm, linkes Bein, rechtes Bein.
1150	Tot.
1500–1550	Ändern von B$, wenn passender Buchstabe gefunden.

Würfeln

Dies ist ein anderes Spiel mit guter grafischer Anzeige, das Sie vielleicht nicht so gut kennen. Der Grundgedanke ist, zu wetten, ob Sie eine 6 würfeln oder nicht. Wenn Sie eine andere Zahl würfeln, wird diese Ihrem Punktestand zuaddiert; wenn es eine 6 wird, fallen Ihre Punkte auf Null. Der Grundgedanke ist einfach, aber das Spiel läuft gut, und mit einem attraktiven Bildaufbau macht es Spaß zu spielen.

Der Bildschirmaufbau

Dies ist wahrscheinlich das Beste am ganzen Spiel, deshalb wollen wir uns ihm zuerst zuwenden (siehe auch Seite 113). Da unser Computer nur schwarze Buchstaben auf grünem Hintergrund drucken kann, haben wir versucht, das als positive Eigenschaft auszunutzen, indem wir Schreibfelder auf einen roten Hintergrund gelegt haben. Wenn kein Text erscheint, werden sie mit Leerzeichen gefüllt.

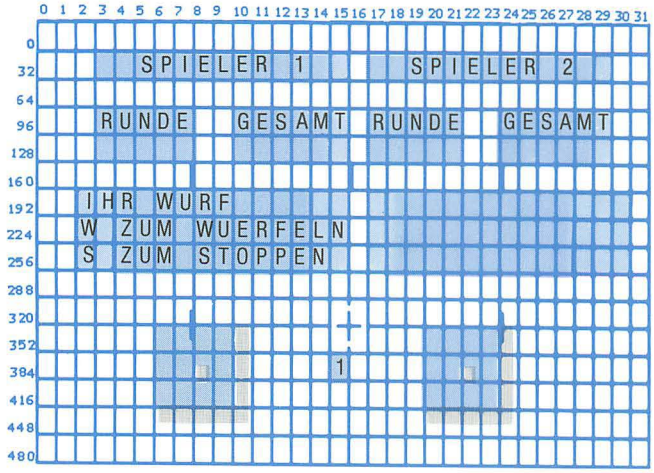
Der Rundenzähler wertet die Punkte des Spielers für jede seiner 4 Runden aus, der Summenzähler die Gesamtpunktzahl. Die Zahl unten in der Mitte gibt an, wie viele Runden schon vorbei sind.
Wir sind besonders stolz auf die Würfel, von denen jeder Spieler einen hat. Wenn er »geworfen« wird, erscheint eine Folge verschiedener Punktmuster, bis der Würfel schließlich bei einem davon anhält. Dies ist etwas umständlicher zu programmieren als einfache Zahlen, aber es sieht besser aus. Blockgrafikquadrate in Schwarz auf Grün (Weiß, falls Ihr Computer es hat, könnte noch besser aussehen) stellen die Punkte auf dem Würfel dar. Kreise wären eine weitere Verbesserung.

Das Spiel

Jeder Spieler hat 4 Runden, und wir benutzen die Variable T für eine FOR ... NEXT-Schleife, die viermal wiederholt wird. D zeigt an, welcher Spieler an der Reihe ist, und dieser Spieler hat so oft die Möglichkeit zu würfeln, wie er es möchte, außer es erscheint eine 6. R wird benutzt, um den Würfel anzustoßen, S um die Runde zu beenden. Am Ende der Runde (aber nicht früher) wird das Rundenergebnis des Spielers zu seiner Gesamtpunktzahl hinzuaddiert.
Den Würfel zu werfen ist eine komplexe Operation. Deshalb haben wir ein Unterprogramm daraus gemacht. Dieses erhält zuerst eine zufällige Zahl und zeigt dann die entsprechenden Punkte auf dem Würfel an. Das passiert fünfmal, um das Rollen des Würfels zu simulieren; dann bleibt der Würfel mit seiner letzten Seite stehen. Dieser letzte Wert X (die zufällige Würfelzahl) wird dann dem Programm zurückgegeben und zum Punktestand addiert. Wenn es eine 6 ist, werden die Punkte auf Null zurückgesetzt.

Die Spieler

Wir besprachen bereits ein Spiel für 2 Spieler – das »Nimm-Spiel« –, aber da war der zweite Spieler immer der Computer. In diesem Spiel kann der zweite Spieler entweder der Computer oder eine andere Person sein. Der Bildschirm zeigt die Namen der Spieler (falls nötig abgekürzt, damit sie in die Felder passen).
Das Programm hat 2 Zweige, einen für einen menschlichen Gegenspieler und einen für das Spiel des Computers. Wir



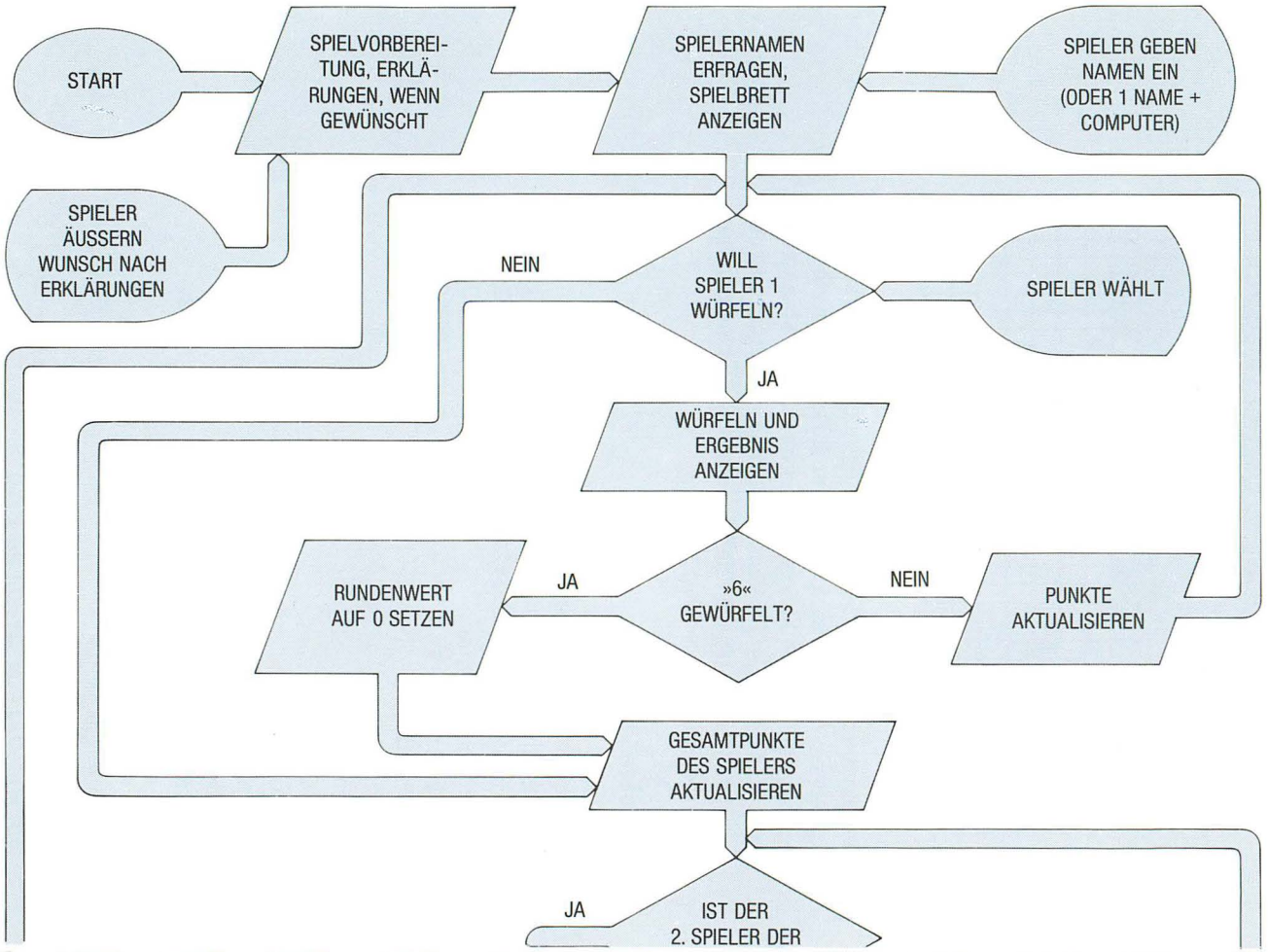
Würfeln –
Der Bildschirmaufbau

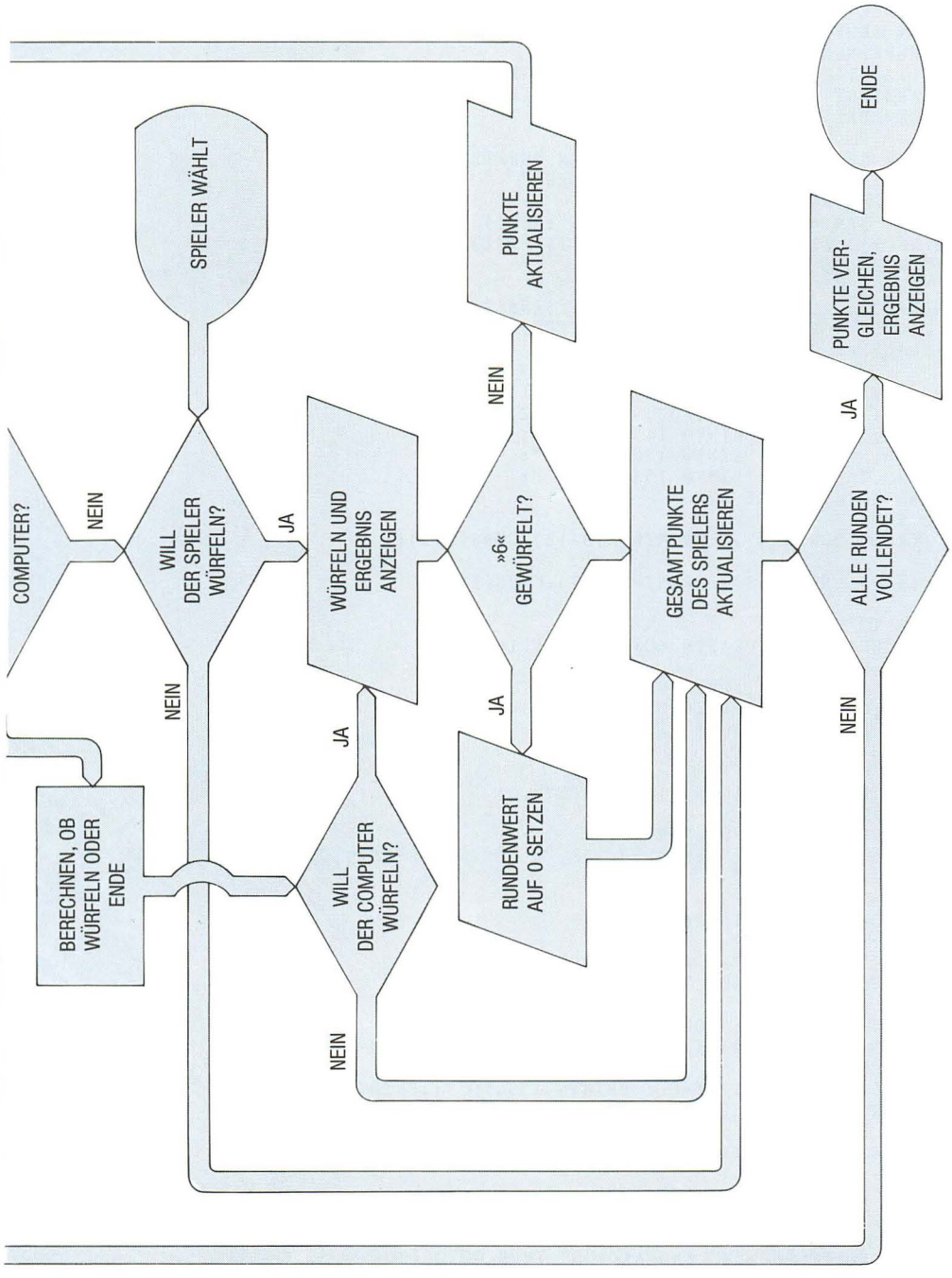
wollen die Strategie des Computers nicht ausführlich diskutieren. Wenn Sie sich die Zeilen 710 bis 730 anschauen, erkennen Sie eine Mischung aus gesundem Menschenverstand und einer Portion Zufall. In diesem Spiel gewinnt nicht nur reine Logik, so daß Sie eine gute Chance haben, den Computer zu schlagen.

Benutzte Variablen

Bildschirmseite der Spieler	D
Sieben diverse Punkte	N(7)
Koordinaten der Punkte (2*7)	SP(2,7)
Allgemeine Fragen	Q$, R$, G$
Name der Spieler	N$(2)
Schleifenzähler	A, B, C, K, N
Rundenzähler	T
Gesamtpunkte	T(2)
Rundenpunkte	RT
Würfelzahl	X
Computergrenzwert	M
Verzögerungsschleife	DL
Gewinner	W$

```
10 REM WUERFELN
20 REM VON MARGARET NORMAN
30 DIM SP(2,7):DIM N$(2):DIM T(2):DIM N(7)
40 FOR D=1 TO 2:FOR N=1 TO 7
50 READ SP(D,N)
60 NEXT N:NEXT D
70 DATA 359,361,391,392,393,423,425,372,374,404
80 DATA 405,406,436,438
90 CLS:PRINT@ 45,"WUERFELN":PRINT
100 INPUT "INSTRUKTIONEN (J/N)";Q$
```


START
SPIELVORBEREI-
TUNG, ERKLÄ-
RUNGEN, WENN
GEWÜNSCHT
SPIELERNAMEN
ERFRAGEN,
SPIELBRETT
ANZEIGEN
SPIELER GEBEN
NAMEN EIN
(ODER 1 NAME +
COMPUTER)
SPIELER
ÄUSSERN
WUNSCH NACH
ERKLÄRUNGEN
NEIN
WILL
SPIELER 1
WÜRFELN?
SPIELER WÄHLT
JA
WÜRFELN UND
ERGEBNIS
ANZEIGEN
RUNDENWERT
AUF 0 SETZEN
JA
»6«
GEWÜRFELT?
NEIN
PUNKTE
AKTUALISIEREN
GESAMTPUNKTE
DES SPIELERS
AKTUALISIEREN
JA
IST DER
2. SPIELER DER


BERECHNEN, OB WÜRFELN ODER ENDE
COMPUTER?
NEIN
JA
WILL DER SPIELER WÜRFELN?
SPIELER WÄHLT
NEIN
JA
WILL DER COMPUTER WÜRFELN?
WÜRFELN UND ERGEBNIS ANZEIGEN
»6« GEWÜRFELT?
NEIN
JA
RUNDENWERT AUF 0 SETZEN
PUNKTE AKTUALISIEREN
GESAMTPUNKTE DES SPIELERS AKTUALISIEREN
ALLE RUNDEN VOLLENDET?
NEIN
JA
PUNKTE VERGLEICHEN, ERGEBNIS ANZEIGEN
ENDE

```
110 IF LEFT$(Q$,1)<>"J" THEN GOTO 240
120 CLS: PRINT "DIES IST EIN SPIEL FUER ZWEI SPIELER"
130 PRINT "(EINER DAVON KANN DER COMPUTER SEIN)"
140 PRINT "JEDER SPIELER HAT 4 RUNDEN"
150 PRINT "IN JEDER RUNDE KANN ER SO OFT"
160 PRINT "WUERFELN , WIE ER MOECHTE ."
170 PRINT "1,2,3,4 ODER 5 WIRD ZU IHRER PUNKTZAHL"
180 PRINT "ADDIERT . EINE 6 BEWIRKT 0 PUNKTE UND"
190 PRINT "BEENDET DIE RUNDE ."
200 PRINT
210 PRINT "DER SPIELER MIT DER HOECHSTEN PUNKTZAHL"
220 PRINT "GEWINNT . DRUECKEN SIE 'RETURN'"
230 INPUT "UM WEITERZUMACHEN .";R$
240 CLS:INPUT "NAME DES ERSTEN SPIELERS";N$(1)
250 IF LEN(N$(1))>12 THEN LET N$(1)=LEFT$(N$(1),12)
260 PRINT "NAME DES ZWEITEN SPIELERS"
270 INPUT "ODER 'C' FUER COMPUTER";N$(2)
280 IF LEN(N$(2))>12 THEN LET N$(2)=LEFT$(N$(2),12)
290 IF N$(2)="C" THEN LET N$(2)="COMPUTER"
300 REM BILDAUFBAU
310 CLS 4
320 PRINT@ (35+INT((12-LEN(N$(1)))/2)),N$(1);
330 PRINT@ (49+INT((12-LEN(N$(2)))/2)),N$(2);
340 PRINT@ 131,"      ";:PRINT@ 145,"      ";
350 PRINT@ 106,"TOTAL";:PRINT@ 120,"TOTAL";
360 PRINT@ 138,"      ";:PRINT@ 152,"      ";
370 FOR A=358 TO 422 STEP 32
380 FOR B=0 TO 3
390 PRINT@ (A+B),CHR$(143);:PRINT@(A+B+13),CHR$(143);
400 NEXT B
410 PRINT@ (A+4),CHR$(138);:PRINT@(A+17),CHR$(138);
420 NEXT A
430 FOR C=454 TO 457
440 PRINT@ C,CHR$(140);:PRINT@ (C+13),CHR$(140);
450 NEXT C
460 PRINT@ 458,CHR$(136);:PRINT@ 471,CHR$(136);
470 PRINT@ 392,CHR$(141);:PRINT@ 405,CHR$(141);
480 LET T(1)=0 : LET T(2)=0 : LET RT=0
500 REM SPIELBEGINN
510 FOR T=1 TO 4
520 PRINT@ 99,"RUNDE";:PRINT@ 113,"RUNDE";
530 PRINT@ 397,T;
540 LET D=1 : REM ERSTER SPIELER
550  PRINT@ 196,"IHR WURF ";:PRINT@ 228,"W ZUM WUERFELN";:PRINT@ 260,"S ZUM
    STOPPEN";
560 FOR A=0 TO 2
570 PRINT@ (210+32*A),STRING$(10," ");
580 NEXT A
590 LET A$=INKEY$:IF A$<>"W" AND A$<>"S" THEN GOTO 590
600 IF A$="S" THEN GOTO 640
610 GOSUB 1000
620 IF X=6 THEN LET RT=0 : SOUND 20,5 : GOTO 640
630 SOUND 200,3 : LET RT=RT+X : PRINT*132,RT;:GOTO  590
640 REM ENDE DER RUNDE
650 LET T(1)=T(1)+RT : LET RT=0 : PRINT@ 133,"0 ";:PRINT@ 139,T(1);
660 LET D=2 : REM ZWEITER SPIELER
670 FOR A=0 TO 2
680 PRINT@ (196+32*A),STRING$(10," ");
690 NEXT A
700 IF N$(2)="COMPUTER" THEN GOTO 770
710 PRINT@ 210,"IHR WURF ";:PRINT@ 242,"W ZUM WUERFELN";:PRINT@ 274,"S ZUM
   STOPPEN";
720 LET A$=INKEY$:IF A$<>"W" AND A$<>"S" THEN GOTO 720
730 IF A$="S" THEN GOTO 860
740 GOSUB 1000
750 IF X=6 THEN LET RT=0 : SOUND 20,5 : GOTO 770
760 SOUND 200,3 : LET RT=RT+X : PRINT@ 146,RT;:GOTO 720
770 REM COMPUTER
780 PRINT@ 210,"ICH BIN DRAN"
790 IF T<4 THEN GOTO 810
```

```
800 IF (T(2)+RT)<T(1) THEN GOTO 820 ELSE GOTO 860
810 LET M=15+RND(10): IF RT<M THEN GOTO 820 ELSE GOTO 860
820 GOSUB 1000
830 FOR DL=1 TO 100 : NEXT DL
840 IF X=6 THEN LET RT=0 : SOUND 20,5 : GOTO 860
850 SOUND 200,3:LET RT=RT+X:PRINT@ 146,RT;:GOTO 790
860 REM RUNDENENDE
870 LET T(2)=T(2)+RT:LET RT=0:PRINT@ 147,"0 ";:PRINT@ 153,T(2);
880 NEXT T
890 FOR DL=1 TO 1000 : NEXT DL
900 REM PUNKTEVERGLEICH
910 IF T(1)>T(2) THEN LET W$=N$(1)
920 IF T(2)>T(1) THEN LET W$=N$(2)
930 CLS
940 IF T(1)=T(2) THEN PRINT@ 235,"UNENTSCHIEDEN" ELSE PRINT@ 230,W$;" GEWINNT"
950 INPUT "NOCH EIN SPIEL (J/N) ";G$
960 IF LEFT$(G$,1)="J" THEN RUN
970 END
1000 REM WUERFEL ROLLEN
1010 FOR K=1 TO 5
1020 LET X=RND(6)
1030 FOR A=1 TO 7
1040 LET N(A)=143
1050 NEXT A
1060 ON X GOTO 1120,1090,1110,1080,1100,1070
1070 LET N(3)=141 : LET N(5)=141
1080 LET N(2)=141 : LET N(6)=141
1090 LET N(1)=141 : LET N(7)=141 : GOTO 1130
1100 LET N(1)=141 : LET N(7)=141
1110 LET N(2)=141 : LET N(6)=141
1120 LET N(4)=141
1130 FOR A=1 TO 7
1140 PRINT@ SP(D,A),CHR$(N(A));
1150 NEXT A
1160 FOR DL=1 TO 100 : NEXT DL
1170 NEXT K
1180 FOR DL=1 TO 200 : NEXT DL
1190 RETURN
```

Zeilenweise Anmerkungen zu »Würfeln«

40–80	Würfelmuster in Matrix laden.
100–110	Bedingter Sprung, um Einleitung zu übergehen (nützlich für weitere Spiele).
250, 280	Namen begrenzen, damit sie in die Felder passen.
320–330	Namen in der jeweiligen oberen Bildhälfte zentrieren.
340	Grüne Blanks unter dem Rundenzähler.
360	Grüne Blanks unter dem Gesamtzähler.
390	Grüne Würfelflächen.
410	Schwarzer Streifen rechts am Würfel.
440	Schwarzer Streifen unten am Würfel.
460	Untere rechte Ecke des Würfels.
470	Einzelner Punkt in Würfelmitte.
510	4 Runden für jeden Spieler.
570	Löschen der Texte für den rechten Spieler, wenn der linke an der Reihe ist.
600	Sprung zum Ende, wenn gewünscht.
620	Kurzer tiefer Ton, wenn 6 gewürfelt.
630	Kurzer heller Ton, wenn anderes Ergebnis.
650	Addiert Summe zu Gesamtsumme, löscht Rundenwert; erhöht Rundenzähler.
790	Andere Computerstrategie in der letzten Runde.
800	Computerzug der letzten Runde bei Gewinn oder Verlust.
810	Computerzug Runden 1 bis 3.
820	Verzögerung, wenn der Computer spielt, um das Resultat einen Moment stehen zu lassen (kein INKEY$).
1010	K steuert 5 Würfeldrehungen (Würfel kommt auf der fünften Seite zur Ruhe).
1030–1050	Alle Würfelpunkte grün.
1060–1120	Verschiedene Punkte schwarz, während der Würfel rollt.
1130–1150	Dann werden grüne und schwarze Punkte gezeichnet.

Anfang und Ende des Programms

Der Einleitungsteil ist diesmal etwas länger. Er gibt dem Spieler die Möglichkeit, die Erklärungen zu überspringen, und erfragt den Namen des Spielers. Wenn der Name des

Spielers zu lang ist, wird er abgeschnitten. Eine nette Möglichkeit wäre etwa, zurückzugehen und nach einer Kurzform zu fragen.
Der Abschlußteil muß die Punkte vergleichen und den Sieger ermitteln. Er gibt dann das Resultat bekannt, das natürlich auch ein Unentschieden sein kann. Die Frage nach einem neuen Spiel vervollständigt das Programm.

Mögliche Verbesserungen

Auch hier gibt es nicht viel Spielraum für Verbesserungen. Es ist kein vollendetes Spiel, und das Programm erfaßt die meisten Möglichkeiten. Sie könnten möglicherweise mehr Geräusche oder aufregendere Anfangs- und Endbildschirme einbauen.

Künstler

Wir haben Ihnen schon ein Programm gezeigt, das Ihren Computer Melodien spielen läßt. Nun, hier ist eines, das es Ihnen erlaubt, Bilder zu zeichnen. Der »Künstler« kettet Kommandos für hochauflösende Grafik an INKEY$-Befehle, genau wie der »Orgelspieler« es mit den Tonbefehlen gemacht hat.
Auf unserem Computer gibt es eine Unmenge Kommandos für die hochauflösende Grafik, und unser Programm läßt dem Benutzer eigentlich jede Möglichkeit offen. So ist natürlich der »Arbeits«-Teil des Programms ein wenig länger und komplizierter als in dem Orgelprogramm.

Das eigentliche Programm

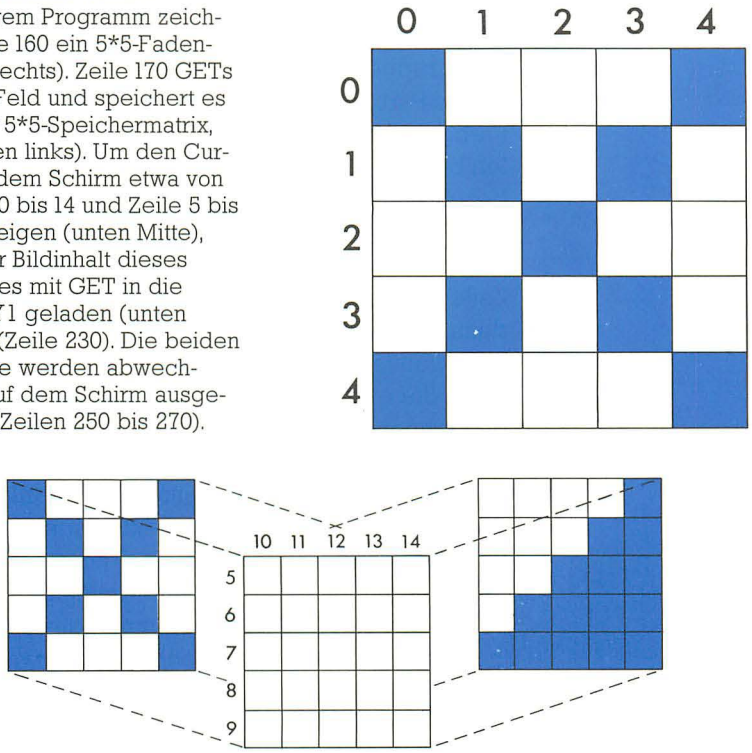
Das Flußdiagramm auf den Seiten 124/125 zeigt die verschiedenen Möglichkeiten, die wir in dem Programm zugelassen haben. Der Benutzer kann Linien in jeder horizontalen, vertikalen oder diagonalen Richtung ziehen, in jeder Größe und mit Hintergrund und Vordergrund beliebiger Farbe, und er kann den Schirm löschen und neu beginnen, ohne das Spiel neu starten zu müssen.
Sie erinnern sich vielleicht daran, daß unser Beispielcomputer für dieses Kapitel 2 verschiedene Farbsätze in der

hohen Auflösung kennt. Es gibt je eine Standardfarbe für Vorder- und Hintergrund. Das Programm beginnt mit einer festen Auswahl, die der Benutzer dann ändern kann, wenn er es wünscht. (Der Schirm muß gelöscht werden, damit man eine andere Hintergrundfarbe wählen kann.)
Die INKEY$-Befehle sind so geordnet, daß die am häufigsten vorkommenden zuerst stehen und der Computer schnell darauf reagieren kann. Der Benutzer wird wohl seltener den Schirm löschen als eine Linie ziehen, so daß diese Möglichkeit zuletzt kommt. Sobald ein Kommando ausgeführt worden ist, springt das Programm wieder zurück zum INKEY$-Teil.
Wir haben ein Hilfskommando eingebaut, weil sonst nur zu leicht vergessen wird, was das Programm alles kann, wenn es läuft. Eingebaut sind ferner Fehlerprüfungen mittels INSTR. Falls Ihr Computer Integervariablen kennt, vereinfacht deren Benutzung die Fehlerprüfung.

Der Cursor

Ein Teil des Programms, der verwirren könnte, sind die Zeilen 160 bis 280. Wir wollen uns einmal genau anschauen, was sie bewirken.
Die meisten Computer haben irgendeine Art von Cursor – ein kleines blinkendes Quadrat oder einen Pfeil, der anzeigt, wo man sich auf dem Schirm befindet. Er erscheint, wenn der Computer bereit ist, etwas Neues zu tun, aber nicht, wenn ein Programm läuft. Deshalb müssen besondere Schritte unternommen werden, wenn im Verlauf eines Programms ein Cursor angezeigt werden soll. Dieser Programmteil ist hauptsächlich damit beschäftigt.
Ein Problem, das bald auftaucht, ist, daß mit dem Zeichnen eines Cursors auf dem Bildschirm sehr wahrscheinlich etwas gelöscht wird, das vorher an derselben Stelle gezeichnet war. Wegen dieses Problems können einige schlechte Zeichenprogramme keine Linie von links nach rechts zeichnen. Wir haben dieses Problem mit einem blinkenden Cursor umgangen, der zwischen Cursor und Untergrund wechselt. Beachten Sie, daß das etwas anderes als die blinkenden Farben in den Farbsätzen mancher Computer ist! Dieses Blinken ist kein Farbblinken, sondern ein programmierter Effekt.
Wir haben dies mit GET- und PUT-Kommandos in unserem BASIC erreicht. Die grafischen Details unseres Fadenkreuz-Cursors sind in der Matrix X1 gespeichert und die grafischen Einzelheiten der entsprechenden Bildschirmstelle in der Matrix Y1. Das Programm wechselt dann hin

Der blinkende Cursor
In unserem Programm zeichnet Zeile 160 ein 5*5-Fadenkreuz (rechts). Zeile 170 GETs dieses Feld und speichert es in einer 5*5-Speichermatrix, X1 (unten links). Um den Cursor auf dem Schirm etwa von Spalte 10 bis 14 und Zeile 5 bis 9 anzuzeigen (unten Mitte), wird der Bildinhalt dieses Bereiches mit GET in die Matrix Y1 geladen (unten rechts) (Zeile 230). Die beiden Bereiche werden abwechselnd auf dem Schirm ausgegeben (Zeilen 250 bis 270).



und her, wobei eine kurze Verzögerung die Wechsel besser erkennbar macht. Zeile 160 zeichnet den Cursor links oben auf dem Bildschirm (Punkt 0 bis 4 in beiden Richtungen). Zeile 170 speichert ihn in X1. Zeile 230 verfährt genauso mit der Zeichnung, und die Zeilen 240 bis 270 wechseln zwischen den beiden Bildern.
GET und PUT sind in dieser Funktion ziemlich einzigartig für unseren BASIC-Dialekt. Wenn Ihrer verschieden ist, müssen Sie eine andere Möglichkeit finden, dieses Problem zu lösen. Manche BASICs kennen ein OVER-Kommando, mit dem Bilder überlagert werden können; manche können Farben mit »EXKLUSIV ODER« verarbeiten, was einen ähnlichen Effekt bewirkt.

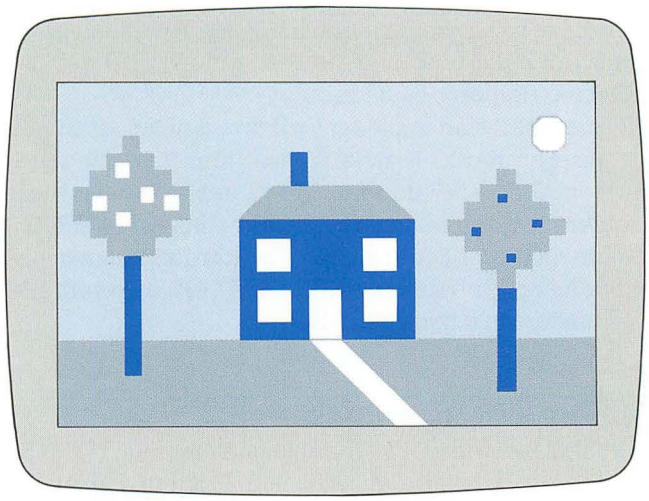
Andere wichtige Punkte

Beachten Sie, wie wir die Befehle im Programm behandelt haben! Der Hauptbefehlssatz am Programmanfang ist

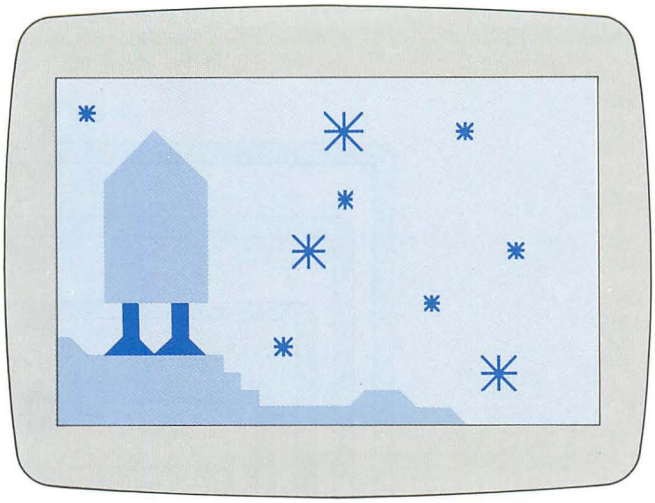
wahlfrei und kann, wie schon früher erwähnt, aufgerufen werden. Die »Farb-Einleitung« zeigt ein Beispiel jeder Farbe auf dem Schirm (in niedriger Auflösung, in der alle erscheinen können, um dem Benutzer die Auswahl einer Farbgruppe zu erleichtern. Durch die Benutzung von A$ für die Farbauswahl kann der gleiche Programmteil beide Farbgruppen verarbeiten, obwohl unterschiedliche Zahlen benutzt werden, um die Farben in den beiden Sets zu beschreiben (das heißt, im zweiten Set werden die Zahlen 5 bis 8 benutzt, um die Farben zu beschreiben; sie werden nicht anders behandelt als die Farben 1 bis 4).

Zeile 210 verwirrt Sie vielleicht anfangs. Warum sollte jedesmal der Schirm gelöscht werden, wenn der Benutzer kein neues Kommando gegeben hat? Heißt das nicht, wenn Sie eine Linie gezeichnet haben und sich dann zurücklehnen, um den nächsten Schritt zu überlegen, daß der ganze Schirm wieder gelöscht wird? Nein, das ist nicht der Fall. CLS löscht den Textschirm. Die Grafikversion davon wäre CLG. Das Löschen des Textschirmes mit CLS beeinflußt die Grafik nicht; es werden nur die Textmeldungen gelöscht. Wenn so etwas auf Ihrem Computer nicht möglich ist, müssen Sie PRINT@-Kommandos für die Eingaben in den Zeilen 400, 430 und so weiter verwenden.

Beachten Sie, daß das »SCALE«-Kommando eine einfache Möglichkeit ist, die Zeichengeschwindigkeit von Linien zu beeinflussen! Sie können das auch recht einfach als eigenes Kommando programmieren, falls es in Ihrem BASIC kein fertiges »SCALE«-Kommando gibt. Sie behandeln S als Variable, und jedesmal, wenn ein Befehl zum Zeichnen gegeben wird, multiplizieren Sie mit dem aktuellen Wert von S.



Künstlerprogramm – Ein Bild auf dem Schirm mit Farbset 1 (Dieses Bild und das auf der nächsten Seite zeigen den Aufbau, aber nicht den vollen Farbbereich der Anzeige; mit Abtönungen konnten wir die einzelnen Farben verdeutlichen.)



Künstlerprogramm – Ein Bild auf dem Schirm mit Farbset 2

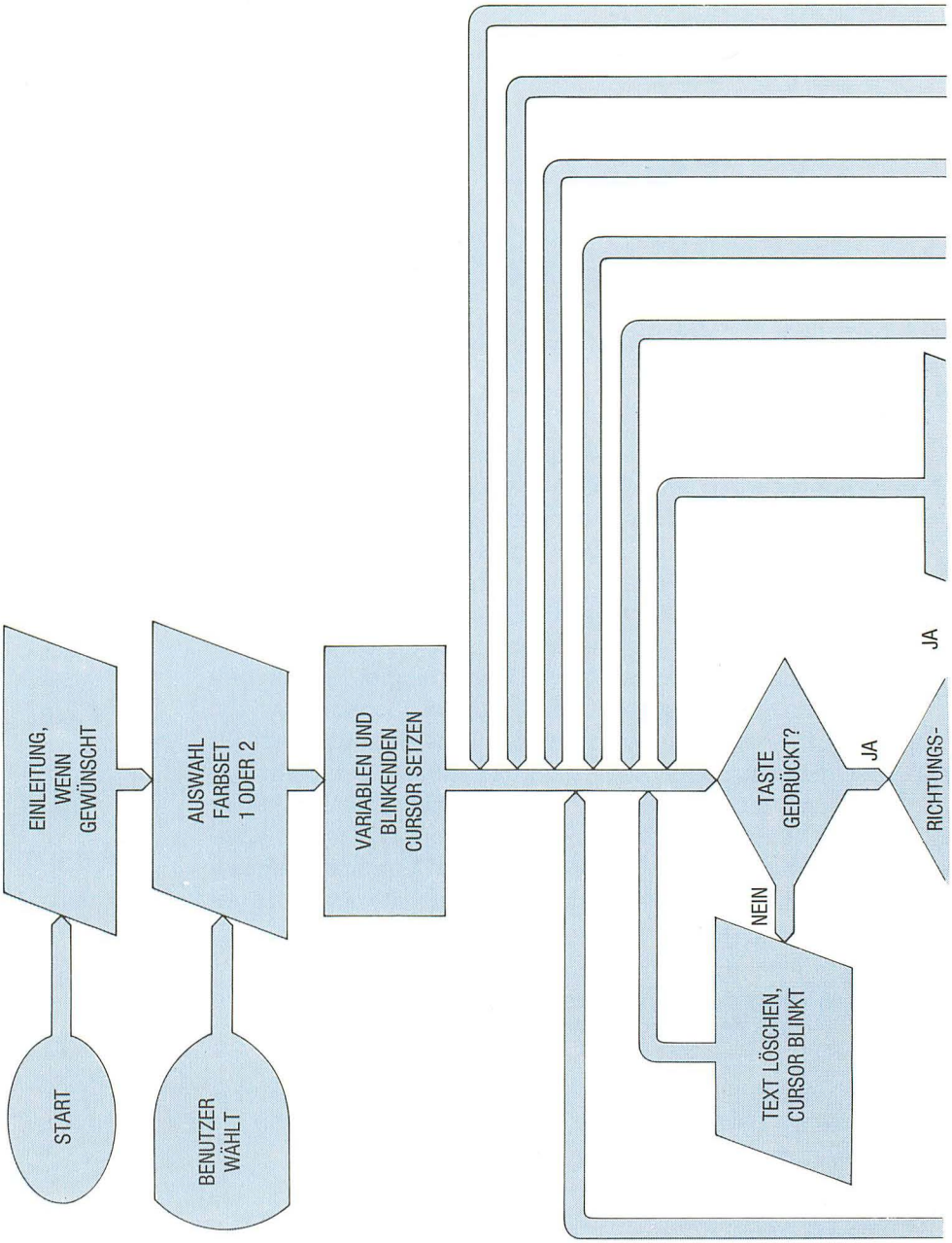
Unser Computer hat sowohl absolute als auch relative Kommandos zum Zeichnen. Sie können entweder die neuen Koordinaten angeben, zu denen Sie eine Linie zeichnen möchten, oder Sie sagen ihm, eine bestimmte Strecke nach oben oder unten zu gehen. Relative Befehle sind normalerweise zum Erstellen von Zeichnungen günstiger, und das Programm ist so geschrieben, daß der Benutzer seine Kommandos auf diese Art und Weise eingibt. Das spiegelt sich in den grafischen Kommandos des Programms wider, bei denen wir die relativen Befehle verwenden. Wenn Ihr Computer diese Möglichkeit nicht bietet, kann dem Benutzer trotzdem dieser Eindruck vermittelt werden, indem die Instruktionen des Benutzers mit solchen Programmbefehlen gekoppelt werden, welche die Koordinaten aktualisieren. Ein Befehl etwa der folgenden Art wäre dafür geeignet:

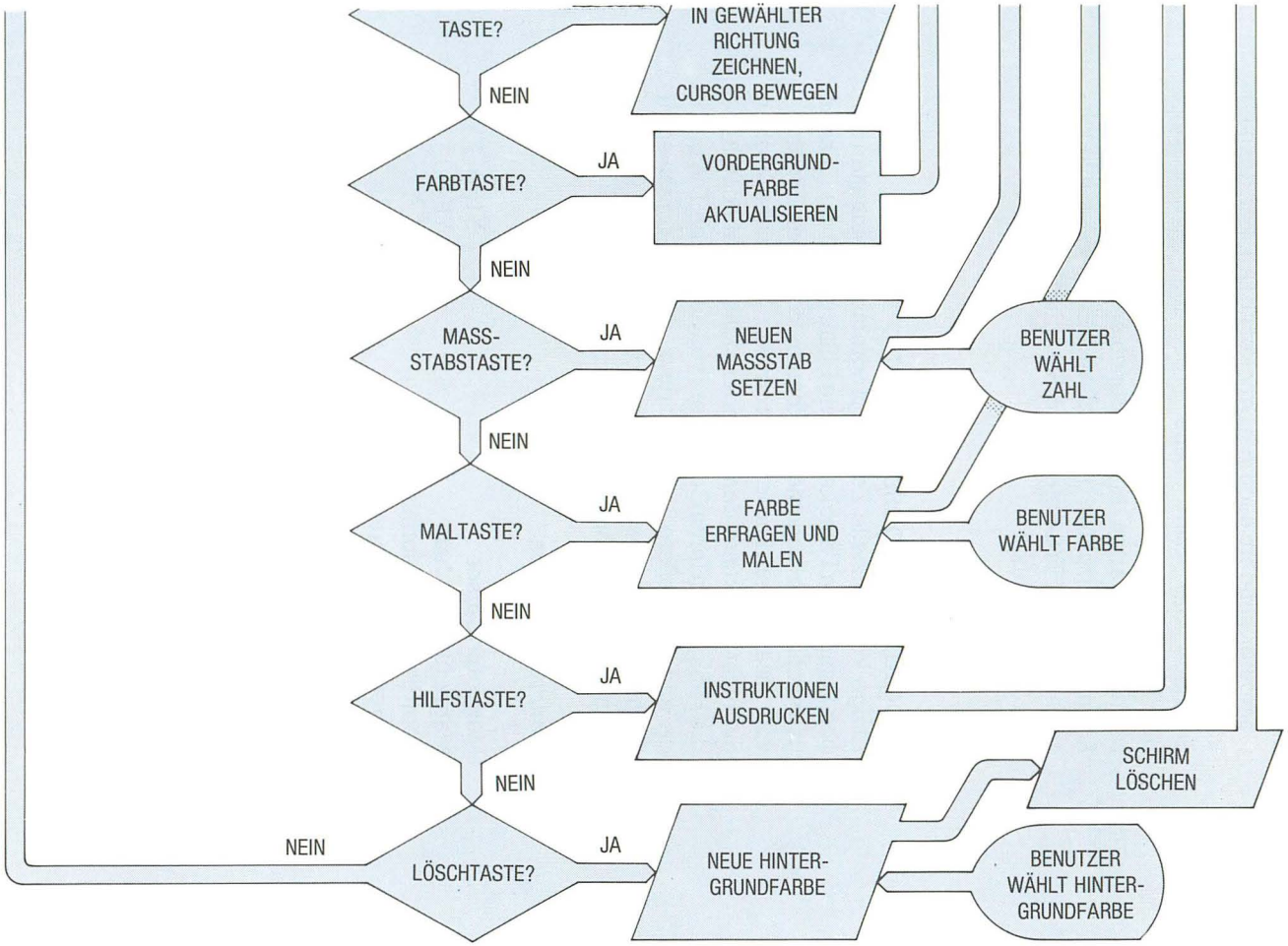
```
290  IF T$ = "L" AND X > S + 1 THEN LET X = X-S:
     DRAW X,Y: GOTO 200
```

Beachten Sie, daß wir solche Befehle ohnehin in unserem Programm haben, um immer die Übersicht über die genaue Cursorposition zu behalten!

Anfang und Ende des Programms

Es gibt in diesem Programm kein natürliches Ende: Es läuft so lange, bis Sie es mit BREAK anhalten. Eine Verbesserung, die auch ein empfehlenswertes Ende darstellte,


START
BENUTZER WÄHLT
EINLEITUNG, WENN GEWÜNSCHT
AUSWAHL FARBSET 1 ODER 2
VARIABLEN UND BLINKENDEN CURSOR SETZEN
TASTE GEDRÜCKT?
JA
NEIN
RICHTUNGS-
JA
TEXT LÖSCHEN, CURSOR BLINKT


TASTE?
IN GEWÄHLTER RICHTUNG ZEICHNEN, CURSOR BEWEGEN
NEIN
FARBTASTE?
JA
VORDERGRUND-FARBE AKTUALISIEREN
NEIN
MASS-STABSTASTE?
JA
NEUEN MASSSTAB SETZEN
BENUTZER WÄHLT ZAHL
NEIN
MALTASTE?
JA
FARBE ERFRAGEN UND MALEN
BENUTZER WÄHLT FARBE
NEIN
HILFSTASTE?
JA
INSTRUKTIONEN AUSDRUCKEN
NEIN
SCHIRM LÖSCHEN
NEIN
LÖSCHTASTE?
JA
NEUE HINTER-GRUNDFARBE
BENUTZER WÄHLT HINTER-GRUNDFARBE

könnte ein sogenannter SCREEN DUMP darstellen – eine Routine, die den Bildschirminhalt auf einen Drucker, eine Kassette oder in ein Diskettenfile ausgibt. Die dafür nötigen Kommandos variieren nicht nur von Computer zu Computer, sondern auch von Drucker zu Drucker; deshalb können sie hier nicht im Detail beschrieben werden.
Schauen Sie sich Zeile 750 an! »DRUECKEN SIE ›RETURN‹, UM FORTZUFAHREN« ist eine einfache Methode, zu warten, bis der Benutzer fertig ist, ehe der Kommandobildschirm gelöscht wird. Es ist ein wenig kürzer als ein INKEY$-Befehl, und Sie müssen trotzdem nur eine Taste betätigen.

Programmverbesserungen

Einige kommerzielle Zeichenprogramme beinhalten solche komplexen Fähigkeiten wie Füllroutinen, die Flächen unterschiedlich schattieren können oder den Eindruck von zusätzlichen Farben hervorrufen. Vielleicht streben Sie nicht unbedingt danach, aber sicherlich möchten Sie die Fähigkeiten Ihres Computers voll ausnutzen. Die einzige andere größere Verbesserung, die wir vorschlagen können, ist die Erweiterung um eine »SCREEN-DUMP«-Routine, die ja gerade erwähnt wurde.

Benutzte Variablen

Matrix für Cursor	X1(4)
Matrix für Bildinhalt	Y1(4)
Koordinaten Cursormitte	X, Y
Allgemeine Antworten	Q$
Gewählte Farbe	Z
String für mögliche Farben	A$
Maßstab	S
Mögliche Kommandos	B$
Aktuelle Auswahl	T$
Hintergrundfarbe	C$
Zeichen- und Randfarben	P1, P2, P1$
Verzögerungsschleife	DL

```
10 REM KUENSTLER
20 REM VON MARGARET NORMAN
30 PCLEAR 4:DIM X1(4,4):DIM Y1(4,4)
40 CLS:PRINT@ 75,"KUENSTLER";:
50 PRINT@ 416,"INSTRUKTIONEN ? (J/N)"
60 INPUT Q$:IF Q$="J" THEN GOSUB 600
70 CLS:PRINT@ 34,"FARBSET 1    FARBSET 2"
80 PRINT@ 98,"1=GRUEN      5=BEIGE    ";CHR$(207);
90 PRINT@162,"2=GELB    ";CHR$(159);"  6=CYAN     ";CHR$(223);
100 PRINT@ 226,"3=BLAU    ";CHR$(175);"  7=MAGENTA  ";CHR$(239);
110 PRINT@ 290,"4=ROT     ";CHR$(191);"  8=ORANGE   ";CHR$(255);
120 PRINT@ 355,"WELCHES MOECHTEN SIE ?":INPUT Z:IF Z<1 OR Z>2 THEN GOTO 120
130 PMODE 3,1 : PCLS : ON Z GOTO 140,150
140 SCREEN 1,0:LET A$="1234": GOTO 160
150 SCREEN 1,1:LET A$="5678"
160 DRAW "BM 0,0;F4;BU4;G4"
170 GET(0,0)-(4,4),X1,G:PCLS
180 LET X=2:LET Y=2:DRAW "BM 2,2":LET S=1
190 LET B$=A$+"SPCILRUDEFGH"
200 LET T$=INKEY$:IF T$<>"" THEN GOTO 280
210 CLS: IF Z=1 THEN SCREEN 1,0 ELSE SCREEN 1,1
220 REM FLASH CURSOR
230 GET (X-2,Y-2)-(X+2,Y+2),Y1,G
240 FOR DL=1 TO 25:NEXT DL
250 PUT (X-2,Y-2)-(X+2,Y+2),X1,PSET
260 FOR DL=1 TO 25 : NEXT DL
270 PUT (X-2,Y-2)-(X+2,Y+2),Y1,PSET:GOTO 200
280 IF INSTR(1,B$,T$)=0 THEN GOTO 200
290 IF T$="L" AND X > S+1 THEN LET X=X-S : DRAW "L" : GOTO 200
300 IF T$="R" AND X < 255-S THEN LET X=X+S : DRAW "R" : GOTO 200
310 IF T$="U" AND Y > S+1 THEN LET Y=Y-S : DRAW "U" : GOTO 200
320 IF T$="D" AND Y > 191-S THEN LET Y=Y+S : DRAW "D" : GOTO 200
330 IF T$="E" AND X < 255-S AND Y>S+1 THEN LET X=X+S :LET Y=Y-S :
    DRAW "E" : GOTO 200
340 IF T$="F" AND X > 255-S AND Y<191-S THEN LET X=X+S : Y=Y+S :
    DRAW "F" : GOTO 200
350 IF T$="G" AND X > S+1 AND Y<191-S THEN LET X=X-S : Y=Y+S :
    DRAW "G" : GOTO 200
360 IF T$="H" AND X > S+1 AND Y>S+1 THEN LET X=X-S :Y=Y-S :
    DRAW "H" : GOTO 200
370 REM FARBE AENDERN
380 IF INSTR(1,A$,T$)<>0 THEN DRAW "C"+T$: GOTO 200
390 REM MASSSTAB AENDERN
400 IF T$="S" THEN INPUT "MASSSTAB (1-15)";S:IF S<1 OR S>15 THEN GOTO 400
    ELSE DRAW "S"+STR$(S*4): GOTO 200
410 REM AUSMALEN
420 IF T$="P" THEN GOTO 430 ELSE GOTO 460
430 INPUT "FUELLFARBE";P1$:IF INSTR(1,A$,P1$)=0 THEN GOTO 430
440 LET P1=VAL(P1$):LET P2=PPOINT(X,Y)
450 PAINT(X+1,Y),P1,P2: PAINT (X,Y+1),P1,P2:GOTO 200
460 REM ERKLAERUNGEN
470 IF T$="I" THEN GOSUB 600: GOTO 200
480 REM SCHIRM LOESCHEN
490 IF T$="C" THEN INPUT "HINTERGRUNDFARBE";C$:IF INSTR(1,A$,C$)=0 THEN
    GOTO 490 ELSE PCLS VAL(C$):GOTO 200
500 GOTO 200
600 REM ERKLAERUNGEN
610 CLS:PRINT "ZEICHNEN SIE EIN BILD AUF DEN SCHIRM"
620 PRINT "BENUTZEN SIE L,R,U,D,E,F,G, ODER H"
630 PRINT "UM DEN CURSOR ZU BEWEGEN ."
640 PRINT "ZUM AENDERN DER ZEICHENFARBE WAEHLEN"
650 PRINT "SIE DIE FARBNUMMER (SET1:1-4,SET2:5-8)
660 PRINT "AENDERN SIE DEN MASSSTAB MIT 'S'"
670 PRINT "UND EINER ZAHL VON 1 - 15 ."
680 PRINT "ZUM AUSMALEN ZEICHNEN SIE LINIEN IN"
690 PRINT "DER RANDFARBE UND DRUECKEN SIE 'P'"
700 PRINT "SOWIE DIE GEWUENSCHTE FARBNUMMER"
```

```
710 PRINT "FUER INSTRUKTIONEN DRUECKEN SIE 'I'"
720 PRINT "ZUM SCHIRMLOESCHEN DRUECKEN SIE 'C'"
730 PRINT "UND DIE HINTERGRUNDFARBE ."
740 PRINT
750 INPUT "WEITER MIT 'RETURN'";Q$
760 RETURN
```

Zeilenweise Anmerkungen zu »Künstler«

30 Reserviert Platz für Grafik, initialisiert Matrix.

70–120 Instruktionen wahlfrei, Farbwahl immer mit CHR$-Blöcken in jeder Farbe.

130 Vierfarb-Grafikmodus mit ausgewählten Farben.

140–150 Aufbau Prüfstring mit erlaubten Farben.

160–170 Fadenkreuz-Cursor in X1 gespeichert, Schirm löschen. G für zu speichernde grafische Einzelheiten.

180 Anfangswerte für Cursor-Position und Scale.

190 Erlaubte Kommandos an Farbstring anhängen.

200 Input über INKEY$; wenn positiv, Sprung nach 280, sonst zur Routine für Cursor-Blinken.

230 Legt Bild »unter« Cursor nach Y1.

240, 260 Verzögerung, damit Blinken deutlicher wird.

250 Nimmt Cursor aus X1, und blinkt an aktueller Position.

270 Hintergrund mit Cursor austauschen.

280 Gültiges Kommando?

290–360 Zeichnen? Wenn ja, zeichnen in Vordergrundfarbe, neue Cursorposition.

380 A$ ist Farbstring; neue Farbauswahl?

400 Neuer Scale-Factor? (Siehe Pendant dazu in Zeilen 660–670)

430 Farbauswahl als String, Prüfung gegen A$, Umwandlung ...

440 ... in eine Zahl; Farbe der Stelle, auf der sich der Cursor befindet, wird zur Randfarbe.

480–490 Löschen des Schirmes mit neuer Hintergrundfarbe.

Breakout

Nun zu unserem letzten Spiel – einer Version des bekannten Favoriten »Breakout«! Für alle, die sich nicht genau erinnern können: Dies ist das Spiel, in welchem eine Mauer niedergerissen werden muß, indem mit einem Ball Steine »herausgehauen« werden. Der Ball springt zurück und muß mit einem Schläger zurückgeschlagen werden.

Das Spiel an sich

Das hört sich einfach an, nicht wahr? Aber in Wirklichkeit ist das ganz schön verzwickt zu programmieren. Sie müssen den Ball bewegen und prüfen, ob der Spieler den Schläger bewegen möchte; Sie müssen den Schläger bewegen oder auch nicht (wie gewünscht); Sie müssen prüfen, ob der Ball die Wand oder den Schläger oder den Rand des Bildschirms berührt, und müssen angemessen darauf reagieren, indem der Punktestand aktualisiert wird, die Ballrichtung geändert wird, ein Fehlschlag gemeldet wird und so weiter. Das Flußdiagramm auf den Seiten 134/135 zeigt, wie all diese Aktionen in einen zusammenhängenden Spielplan gepackt werden können.

Es ist wichtig, »Breakout« zu einem schnellen Spiel zu machen. Sie sollten experimentieren, um zu sehen, wie Ihr Computer am schnellsten reagiert. Wir haben es als gute Methode für dieses Programm angesehen, das grafische Zeichen, das den Ball darstellt (ein violettes Quadrat), an die gewünschte Stelle zu POKEn. Dies gab uns auch die Möglichkeit, in diesem Kapitel ein Programm zu bringen, welches PEEK und POKE benutzt! Wenn das auf Ihrem Computer nicht schnell genug funktioniert, können Sie als Alternative einmal PRINT versuchen. Den Ball in hoher Auflösung zu zeichnen wäre sicherlich zu langsam.

Unsere Methode erreicht eine annehmbare Geschwindigkeit, führt allerdings nicht zu einem superschnellen Spiel. Um es in Stufe 2 zu beschleunigen, haben wir darauf zurückgegriffen, den ganzen Computer mit dem speziellen POKE-Kommando schneller zu machen.

Da Sie PEEK und POKE vielleicht nicht allzugut kennen, wollen wir diese beiden Kommandos erst einmal kurz besprechen, ehe wir fortfahren. Mit POKE können Sie Werte direkt in vorgegebene Speicherzellen des Computers schreiben. Sie können Codes in beliebige beschreibbare Speicherplätze des Computers ablegen, einschließlich der Stellen, in denen das Programm gespeichert wird, und

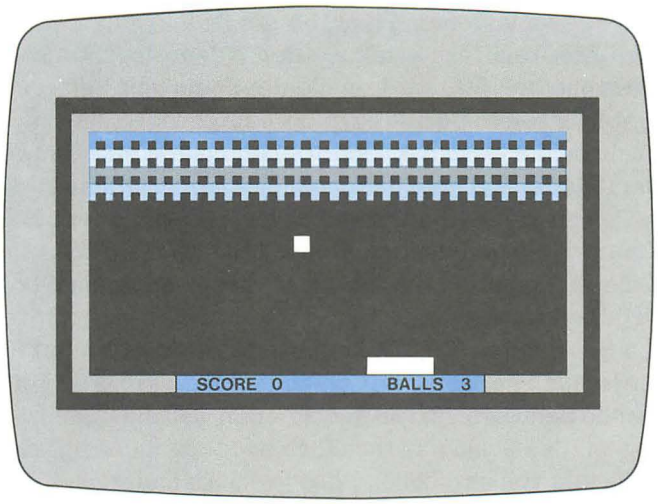
diverser Arbeitsregister. Aber seien Sie gewarnt: Wenn Sie die falschen Stellen erwischen, »läuft« Ihr Computer eventuell »Amok«. Ein Flüchtigkeitsfehler beim Tippen kann Ihren Computer zum Absturz bringen. Der Hardware als solcher kann dabei nichts passieren, aber um sie wieder zum Laufen zu bringen, müssen Sie einen »Reset« auf der Maschine machen oder den Stromkreis unterbrechen. Und so verlieren Sie Ihr Programm im Speicher. Und die Moral von der Geschicht'? SAVEn Sie immer ein Programm mit POKEs, bevor Sie es laufen lassen (nachdem Sie es so sorgfältig wie möglich überprüft haben)! Sonst haben Sie viel Zeit mit Tipparbeit vertan.

Die Werte, die wir POKEn, sind ASCII-Werte und die ASCII-Erweiterungen für grafische Zeichen. POKEn erfordert nicht die CHR$-Funktion, die normalerweise dem Computer mitteilt, daß ein ASCII-Code folgt.

Wohin POKEn wir die einzelnen Werte? Natürlich in den Speicherbereich des Computers, der für den Bildschirm zuständig ist. In unserem Computer reicht dieser von den Adressen 1024 bis 1535. Dies ist auf Ihrem Computer wahrscheinlich anders, so daß Sie Ihr Manual zu Rate ziehen müssen, falls Sie die Adressen nicht bereits wissen. Vergewissern Sie sich auch, wie der Zusammenhang mit den Schirmpositionen ist! Nicht alle Computer (wenn auch die meisten) numerieren zeilenweise durch, so wie das PRINT@-Kommando, das wir an anderer Stelle in diesem Buch verwendet haben.

PEEKen ist die ergänzende Technik, die den Inhalt bestimmter Speicherzellen liefert. Für den Bildschirmspeicher kann sie angeben, welches Zeichen an einer bestimmten Stelle steht. Wir PEEKen die Stelle, wo der Ball hinfliegen wird (es ist sinnlos zu PEEKen, wo er gerade ist – dort befindet sich, wie wir wissen, ein lilafarbenes Quadrat), um herauszufinden, ob er einen Stein treffen wird, oder den Schläger oder den Rand. Zur leichteren Durchführung dieser Idee haben wir einen beigefarbenen Rand um das Spielfeld gezogen. Eine andere Möglichkeit, den Ball zurückzuschlagen, wäre es, die aktuellen Koordinaten zu prüfen und diese gegebenenfalls als Spielfeldgrenze zu behandeln.

Obwohl die Zahlen für PEEK und POKE nicht die gleichen sind wie die PRINT@-Zahlen, können sie genauso behandelt werden, wenn wir einen Faktor dazuaddieren. Um etwa das Äquivalent für PRINT@ 133 zu erhalten, POKEn wir in die Zelle 1024 (Beginn des Bildspeichers) +133 und so weiter. So können wir Schleifen und Variablen verwenden, um die Richtung und Lage des Balls zu beeinflussen.



Breakout –
Der Bildschirmaufbau

Grafikzeichen zum Aufbau der Mauer

Da Sie nur auf Textstellen PEEKen können und nicht auf den Grafikschirm (nun, Sie können, aber es geht völlig anders), ist es angebracht, die Mauer mit grafischen Zeichen aufzubauen und nicht mit hochauflösenden Befehlen. Wir haben das oben rechts gezeigte grafische Zeichen verwendet.
Eines zum Schluß! Wir merkten, daß das Spiel in seiner ersten Version ein paar Steine ganz am Rand übrig ließ und daß es schwer war (unsere Tester meinten: unmöglich), diese letzten Steine zu treffen. Um auch diese zu löschen, gibt es einen Zufallsteil, der bei Punkten über 6000 beschleunigt (das heißt, wenn die Mauer zu 80% demoliert ist), und der den Ball seitwärts bewegt, um 2 Steine auf einmal zu treffen. Sie können das Spiel auch ohne diese Erweiterung testen und sie nur einbauen, wenn sie auf Ihrem Computer nötig ist.

Der Gebrauch von Joysticks

Für »Breakout« müssen nicht unbedingt Joysticks verwendet werden: Sie können das INKEY$-Kommando verwenden, wie wir es bereits in den früheren Spielen getan haben. Aber diese Version wurde für Joysticks geschrieben, um Ihnen deren Programmierung zu verdeutlichen.
Die Joysticks auf unserem Computer (es können 2 sein, aber für dieses Spiel wird nur einer benutzt) liefern 2 Werte. Einer zeigt die Position an: dies ist eine Zahl zwischen 0

und 63. Der andere zeigt an, ob der Druckknopf betätigt wird. Dies kann festgestellt werden, indem diejenige Speicherzelle gePEEKt wird, an die der Knopf sein Signal abgibt. In unserem Computer erfaßt eine einzige Zelle die Signale beider Joystick-Knöpfe. Der normale Wert ist entweder 127 oder 255, aber es können sich verschiedene andere Zahlen ergeben, abhängig davon, ob einer oder beide Knöpfe gedrückt wurden. Durch AND mit dem Wert 1 in Zeile 280 erfahren Sie, ob einer dieser anderen Werte gegeben wurde.
Ein besonderes JOYSTK-Kommando liefert den Wert für die Joystick-Position. Die Programmzeilen, die dieses Kommando benutzen, sind so geschrieben, daß die Geschwindigkeit des Schlägers (natürlich auch die Richtung) sich abhängig von der Stellung des Joysticks ändert.
Nicht alle BASICS haben ein spezielles Joystick-Kommando. Eine Alternative ist es, in die zusätzliche(n) Speicherzelle(n), in denen die Joystickpositionen zu lesen sind, zu PEEKen.
Obwohl das Grundprinzip der Benutzung von Joysticks bei allen Computern ähnlich ist, sind die genauen Werte und Befehle spezifisch für unseren Computer ausgelegt. Wenn Sie Joysticks benutzen, sollten Sie Ihr Manual sorgfältig in Verbindung mit unseren Befehlen studieren.
Wenn Sie statt dessen INKEY$-Befehle benutzen, müssen Sie eventuell die Antwortzeit Ihrer Tastatur verkürzen oder den Puffer leeren, wenn sie eine automatische Wiederholung hat. Ihr Manual wird Ihnen dabei behilflich sein.

Der Bildschirm

Wir haben bereits viele Aspekte des Bildschirmaufbaus diskutiert, weil dieser sehr wichtig für ein Spiel ist. Verglichen mit all den »Breakout«-Spielen finden wir, daß unseres recht attraktiv ist mit seiner guten Farbausnutzung und Punktanzeige. Die beigefarbene Grenze ist eine recht gute Idee, besonders wenn Ihr Computer eine breite »Schirmgrenze« hat, die dieselbe Farbe wie das Spielfeld annehmen kann. Sonst könnte es verwirrend sein, wenn der Ball die Grenze des nutzbaren Schirms trifft!
Obwohl Farbe attraktiv ist, ist sie nicht notwendig (wie Sie wissen, wenn Sie schon einmal »Breakout« auf einem Schwarzweiß-Bildschirm gespielt haben). Sie müssen jedoch besonders vorsichtig mit den Ballkoordinaten sein, wenn Sie sich auf schwarzweiße Zeichen beschränken.
(Wenn Sie von Ihrem Hersteller keine Joysticks für Ihren

Computer erhalten, wenden Sie sich an unabhängige Lieferanten.)

Ändern des Schwierigkeitsgrads

Genauso wie wir das Spiel in Stufe 2 beschleunigen, haben wir eine Option eingebaut, um die Schlägergröße zu bestimmen. Natürlich ist das Spiel mit einem kleineren Schläger schwerer.

Anfang und Ende des Spiels

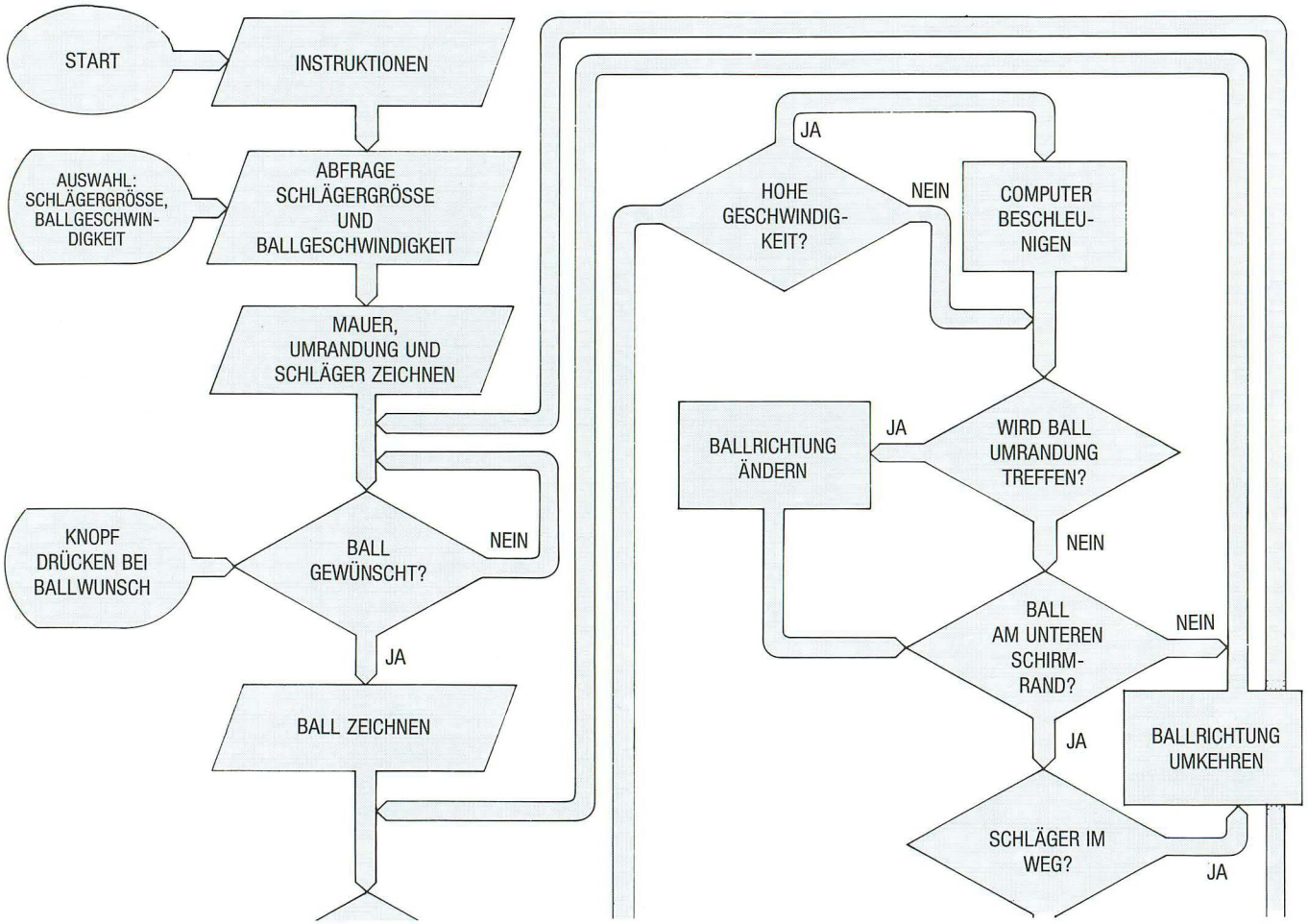
Der Spieler schafft es nicht immer, bis zum Spielende die ganze Mauer einzureißen. Wenn er dieses Ziel jedoch erreicht, erhält er eine zusätzliche Belohnung in Form einer besonderen Gratulation.
Falls Sie irgendwo am Timing Ihres Computers gedreht haben, um das Spiel schneller ablaufen zu lassen, müssen Sie unbedingt daran denken, vor Spielende den Normalzustand wiederherzustellen. Außer am Programmende (Zeile 470) ist es ratsam, die notwendigen Befehle auch noch in eine Fehlerroutine zu packen, wenn Ihr Computer so etwas unterstützt. Fehlerroutinen werden sowohl durch normale Programmfehler als auch durch die »BREAK«-Taste angesprochen. Ein Befehl wie etwa

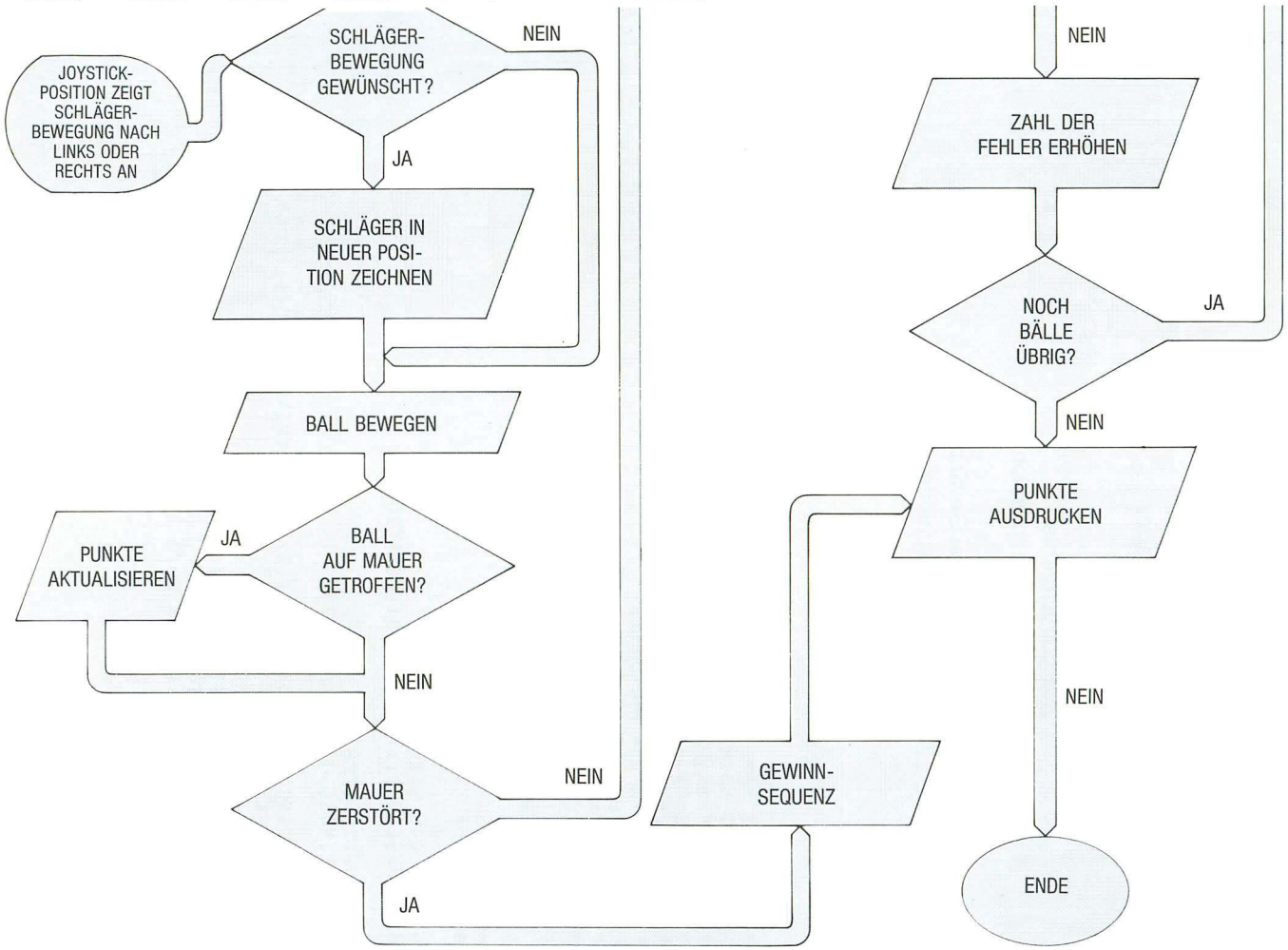
```
ON ERROR POKE & HFFD6,0: END
```

reicht aus. Eigentlich hat unser BASIC gar kein ON ERROR-Kommando, so daß das Beispiel rein theoretisch ist. Sie müssen die entsprechenden Verlangsamungskommandos Ihres eigenen Computers hier einsetzen.

Verbesserungen des Spiels

Wir sind eigentlich mit unserer Version von »Breakout« recht zufrieden, aber wie bei all unseren Programmen ist es schwer zu sagen, wie leicht die Übertragung auf einen anderen Computer gelingt. Wir überlassen es Ihnen, unser »Breakout« zurechtzustutzen und an Ihrer eigenen Version zu arbeiten, bis sie Ihnen gefällt.
(Das Flußdiagramm, das Programm und die Anmerkungen zu »Breakout« finden Sie auf den folgenden Seiten.)


START
INSTRUKTIONEN
AUSWAHL: SCHLÄGERGRÖSSE, BALLGESCHWIN-DIGKEIT
ABFRAGE SCHLÄGERGRÖSSE UND BALLGESCHWINDIGKEIT
MAUER, UMRANDUNG UND SCHLÄGER ZEICHNEN
KNOPF DRÜCKEN BEI BALLWUNSCH
BALL GEWÜNSCHT?
NEIN
JA
BALL ZEICHNEN
JA
HOHE GESCHWINDIG-KEIT?
NEIN
COMPUTER BESCHLEU-NIGEN
BALLRICHTUNG ÄNDERN
JA
WIRD BALL UMRANDUNG TREFFEN?
NEIN
BALL AM UNTEREN SCHIRM-RAND?
NEIN
JA
BALLRICHTUNG UMKEHREN
SCHLÄGER IM WEG?
JA


SCHLÄGER-
BEWEGUNG
GEWÜNSCHT?
NEIN
JOYSTICK-
POSITION ZEIGT
SCHLÄGER-
BEWEGUNG NACH
LINKS ODER
RECHTS AN
JA
SCHLÄGER IN
NEUER POSI-
TION ZEICHNEN
BALL BEWEGEN
PUNKTE
AKTUALISIEREN
JA
BALL
AUF MAUER
GETROFFEN?
NEIN
MAUER
ZERSTÖRT?
NEIN
JA
GEWINN-
SEQUENZ
NEIN
ZAHL DER
FEHLER ERHÖHEN
NOCH
BÄLLE
ÜBRIG?
JA
NEIN
PUNKTE
AUSDRUCKEN
NEIN
ENDE

Benutzte Variablen

Schlägergröße	BS
Geschwindigkeit	SP
Fehlschläge	M
Punkte	SC
Schleifenzähler	I, J
Schlägerposition	BX
String mit Schläger	B$
Ballposition	BL
Ballrichtung	V
Zwischenvariable	K
Inhalt neuer Ballposition	A
Zufallszahl	C
Schirmfarbe bei Gewinn	CL
Verzögerungsschleife	D

```
10 REM *** BREAKOUT ***
20 REM VON MARGARET NORMAN
30 CLS: PRINT@ 10,"BREAKOUT":PRINT
40 PRINT "ZERSCHIESSEN SIE DIE MAUER "
50 PRINT "MIT DREI BAELLEN ."
60 PRINT "BEWEGEN SIE DEN SCHLAEGER "
70 PRINT "MIT DEM JOYSTICK . DRUECKEN"
80 PRINT "SIE DEN KNOPF FUER EINEN BALL"
90 INPUT "SCHLAEGERGROESSE (3-7) ";BS
100 IF BS<3 OR BS>7 THEN GOTO 90
110 INPUT "BALLGESCHWINDIGKEIT (1,2)";SP
120 IF SP<1 OR SP>2 THEN GOTO 110
130 LET M=0 : LET SC=0
150 REM MAUER AUFBAUEN
160 CLS 0 : PRINT@ 0,STRING$(32,207);
170 FOR I=32 TO 448 STEP 32
180 PRINT@ I,CHR$(207);:PRINT@ I+31,CHR$(207);
190 NEXT I
200 FOR J=33 TO 129 STEP 32
210 PRINT@ J,STRING$(30,126+(J-1)/2);
220 NEXT J
230 PRINT@ 480,STRING$(5,207)+STRING$(22,143)+STRING$(4,207);:POKE 1535,207
240 PRINT@ 487,"PUNKTE";SC;"   BAELLE";3-M;
250 REM SCHLAEGER,WARTEN AUF BALL
260 LET BX=10 : LET B$=STRING$(B$,239)
270 PRINT@ 449+BX,B$;:GOSUB 1000
280 IF (PEEK(65280) AND 1) > 0 THEN GOTO 270
300 REM BALL ZEICHNEN
310 LET BL=1194+RND(10):POKE BL,239:LET V=33
320 GOSUB 1000
330 GOSUB 1500
340 IF SC=7500 THEN GOTO 600
350 IF SP=2 THEN POKE &HFFD7,0
360 REM REFLEKTION AN WAND
370 IF PEEK(BL+V)<>207 THEN GOTO 400
380 SOUND 50,1 : IF V=33 OR V=-31 THEN LET V=V-2 ELSE LET V=V+2
390 IF BL<1088 THEN LET V=ABS(V)
400 IF BL<1440 THEN GOTO 320
```

```
410 REM BALL GETROFFEN
420 IF PEEK(BL+32)<>239 THEN GOTO 440
430 SOUND 50,1:IF V=33 THEN LET V=-31 : GOTO 320 ELSE LET V=-33 : GOTO 320
440 REM BALL VERFEHLT
450 SOUND 20,10:LET M=M+1:IF M<>3 THEN POKE BL,128:PRINT@ 505,3-M;:GOTO 280
460 REM ENDE DES SPIELS
470 POKE &HFFD6,0
480 CLS:PRINT@ 35,"IHRE PUNKTE ";SC:PRINT
490 INPUT "NOCH EIN SPIEL (J/N)";Q$
500 IF Q$="↑" THEN END ELSE GOTO 90
600 PLAY "T30V3104BAGFEDCO-BAGFEDCO-BAGFEDC"
610 FOR CL=0 TO 8
620 CLS CL:PRINT@ 235,"GUT GEMACHT!";
630 FOR D=0 TO 100 : NEXT D
640 NEXT CL
650 GOTO 460
1000 REM SCHLAEGER BEWEGEN
1010 LET K=BX
1020 LET BX=BX+(JOYSTK(0)<10 AND BX>0)-(JOYSTK(0)>50 AND BX<(30-BS)):
     GOSUB 1100
1030 LET BX=BX+(JOYSTK(0)<30 AND BX>0)-(JOYSTK(0)>30 AND BX<(30-BS)):
     GOSUB 1100
1040 RETURN
1100 IF BX>K THEN PRINT@ (448+BX),CHR$(128)+B$; ELSE PRINT@ (449+BX),B$;
1110 IF BX<K THEN PRINT@ (449+BX+BS),CHR$(128);
1120 RETURN
1500 REM BALL BEWEGEN
1510 POKE BL,128:LET BL=BL+V:LET A=PEEK(BL):POKE BL,239
1520 IF A=142 THEN LET SC=SC+100 : GOTO 1600
1530 IF A=158 THEN LET SC=SC+75 : GOTO 1600
1540 IF A=174 THEN LET SC=SC+50 : GOTO 1600
1550 IF A=190 THEN LET SC=SC+25 : GOTO 1600
1560 IF BL>=1088 THEN RETURN
1600 LET V=32+(V=-33)-(V=-31)-(V=33)+(V=31)
1610 SOUND 100,1:PRINT@ 493,SC;
1620 LET C=RND(5): IF C=5 OR (C<3 AND SC>=6000) THEN GOTO 1630 ELSE RETURN
1630 LET K=BL+(V-32):LET A=PEEK(K)
1640 IF A=207 THEN RETURN
1650 POKE BL,128:LET BL=K:POKE BL,239:GOTO 1520
```

Zeilenweise Anmerkungen zu »Breakout«

90 Schlägergröße hängt von der Anzahl grafischer Zeichen ab.

100 Bei Geschwindigkeit 2: Maschine durch POKE in 350 beschleunigen.

160 Hintergrund schwarz, beige Grenze oben. Beachten Sie, daß CHR$ nicht im STRING$-Befehl enthalten sein muß.

170– 190 Beige Grenze links und rechts.

200– 220 4 Reihen verschiedenfarbiger Steine für Mauer.

230 Beige Grenze und grüne Punktanzeige unten. Zeichen werden eingePOKEt, nicht gePRINTet, damit der Schirm nicht scrollt.

260 239 ist der Code für violette Schlägerblocks.

280 Wird der Knopf des Joysticks gedrückt?

310 Zufällige Ballposition setzen und lila Quadrat für Ball zeichnen. V bei 33 zeichnet den Ball eine Zeile tiefer und eine Spalte weiter rechts nach jeder Schleife.

340 7500 maximale Punktzahl. Belohnung dafür.

350 Beschleunigung der Maschine (Dragon-spezifisch). Wird in Zeile 470 rückgängig gemacht.

370 Nächste Ballposition Feldgrenze? Sonst Reflektion überspringen.

380 Korrigiert Richtung des Balles (siehe Text).

390 Prüfung, damit Ball nicht vom Schirm verschwindet.

400 Wenn noch nicht unten, Ball eine Reihe tiefer zeichnen.

420 Prüfung, ob Quadrat unterhalb Ball Teil des violetten Schlägers ist. Wenn nicht, Ball verfehlt.

430 Kurzer Ton, wenn Ball Seiten oder Schläger trifft.

450 Langer tiefer Ton, wenn Ball verfehlt. Punkte, neuer Ball, bis alle Bälle verbraucht.

470 POKE für normale Maschinengeschwindigkeit.

600 Belohnungsmelodie, wenn ganze Mauer weg.

610– 640 Belohnung ebenfalls Schirmblinken in diversen Farben.

1010 Speichert Schlägerposition vorübergehend in Variable K.

1020–1110 Lesen Joystick (0...63) und Bewegen des Schlägers (siehe Haupttext).

1510 Löschen Ball (schwarz, Code=128). Fügt Bewegungsvariable zur Ballpositionsvariablen hinzu. Neuer Code für neuen Block nach Variable A. Ball an dieser neuen Stelle drucken.

1520–1550 Punkte aktualisieren, abhängig vom Wert des getroffenen Wandsteines.

1560 Rückkehr zum Hauptprogramm, wenn Ball am oberen Bildschirmrand.

1600 Umkehr Ballrichtung.

1610 Kurzer heller Ton: Treffer, Punkte aktualisieren.

1620 Zufallselement beschleunigt, gegen Ende das Spiel, um alle Steine zu treffen.
1630 Wenn Zufallsfolge aktiviert, Prüfen des Elements direkt über dem Ball.
1640 Beenden der Folge, wenn beige obere Grenze erreicht.
1650 Ball eine Reihe nach oben.

Der Kauf von fertigen Programmen

Auch wenn Sie lernen sollten, Ihre eigenen Programme zu schreiben, möchten Sie sicher einige professionelle kaufen. Hier sind ein paar Hinweise, die Sie dabei beachten sollten.

Welche Medien werden benutzt?

Programme sind hauptsächlich in 5 verschiedenen Formen erhältlich:

- ROM-Kassetten und ROM-Chips
- Kompaktkassetten
- Floppy-Disks
- Listings in Zeitschriften und Büchern
- »Tele-Software«, vermittelt durch Teletext oder elektronisch durch ein Netzwerk.

ROMs sind in der Entwicklung teuer, und so werden sie nur von größeren Software-Häusern produziert, die erwarten können, davon größere Stückzahlen zu verkaufen. Als Folge davon ist die Qualität meistens – wenn auch nicht immer – sehr hoch. Die Kosten sind allerdings ebenfalls hoch, manchmal geradezu lachhaft im Vergleich zu den geringen Herstellungskosten solcher Kassetten oder ROMs in der Massenproduktion. Dennoch sind die Programme in dieser Form robust und leicht zu benutzen. Sie sind außerdem schwer zu kopieren, was ein Grund für den Trend in diese Richtung ist!

Programme auf Kompaktkassetten oder Floppy-Disks sind leicht herzustellen; und so gibt es auf diesem Sektor – abgesehen von vielen großen und angesehenen Unternehmen – eine Menge etwas dubioser Firmen. Programme auf Disketten sind einfach zu benutzen und schnell im Zugriff. Kassetten sind langsam; aber wenn Sie kein Diskettenlaufwerk haben, werden Sie noch darauf angewiesen sein. Beide Formen sind empfindlich, und Sie müssen sie sorgfältig behandeln. Nach Möglichkeit ziehen Sie sich eine Sicherungskopie. Wenn Sie es nicht schaffen, weil das Listing gesichert ist, sollte der Programmhersteller bereit sein,

Ihre Kopie einzutauschen, wenn sie einmal nicht mehr funktioniert.
Listings in Zeitschriften und Büchern sind die unbequemste Form der Programme: *Sie* sind es, der das Programm in den Computer eintippen muß! (Wenn Sie es einmal eingetippt haben, werden Sie es natürlich auf Kassette oder Diskette sichern.) Sie müssen auch selbst die Fehler im Programm finden, wenn Sie sich irgendwo vertippt haben oder wenn sich – was recht oft vorkommt – Fehler im gedruckten Listing befinden.
Die Listings sind normalerweise in BASIC oder einer anderen höheren Programmiersprache geschrieben; in bezug auf die Geschwindigkeit lassen sie sich meistens nicht mit einer Maschinencodeversion vergleichen. Das Eintippen von Listings ist jedoch eine gute Möglichkeit, sich billig und recht schnell eine Programmbibliothek aufzubauen.
Tele-Software und Software, die über Netzwerke verkauft wird, sind neue Entwicklungen, die sich offensichtlich ganz gut anlassen. Im Moment ist die angebotene Netzwerk-Software nicht gerade billig. Sie müssen eine Mitgliedsgebühr bezahlen, außerdem die Programme, die Sie sich aussuchen. Wenn jedoch die Benutzerzahl steigt, sollte der Preis sinken. Tele-Software ist stark ausbildungsorientiert. Netzwerk-Software dagegen allgemeiner. Von der Theorie her ist das Laden der Software einfach, aber in der Praxis kann es eine Weile dauern, ehe man damit »klarkommt«.

Software-Auswahl für Ihren Computer

In welcher Form auch immer die Software vorliegt: Es ist wichtig, sich zu vergewissern, daß sie auf Ihrem Computer läuft. Prüfen Sie nicht nur, für welches Computermodell das Programm geschrieben wurde, sondern auch die Notwendigkeit von Extras (mehr Speicher? Joysticks?) und die Betriebssystemversion – wenn Sie wissen, daß es mehrere für Ihren Computer gibt! Im Zweifelsfalle fragen Sie vor dem Kauf. Wenn Sie ein Listing anpassen wollen – wie jene aus dem lezten Kapitel – prüfen Sie die verwendeten Kommandos! Benutzt Ihr BASIC sie auch, oder wissen Sie, wie Sie den gleichen Effekt mit anderen Kommandos erreichen können? Versuchen Sie, genau zu verstehen, was in den

einzelnen Programmzeilen gemacht wird und wo Sie beispielsweise eine andere Bildschirmgröße anpassen müssen.
Programme anzupassen ist nicht immer leicht, und wir raten Ihnen nicht, es bei sehr maschinenspezifischen Listings oder bei irgendwelchen langen Programmen zu versuchen. Sie sollten keine Programme anpassen, die nicht als gedrucktes Listing vorliegen.
Ein großes Software-Angebot ist ein starkes Verkaufsargument für einen bekannten Computer. Unbekannte oder sehr neue Geräte finden keine so gute Unterstützung. Glauben Sie den Computeranzeigen nicht! Schauen Sie selbst, ob es Programme gibt, ehe Sie einen Computer auswählen! Einen brandneuen Computer zu kaufen, für den noch keine Programme entwickelt wurden, kann eine Wartezeit von 6 Monaten oder mehr bedeuten, ehe die erste Auswahl auf dem Markt auftaucht. Einen unbekannten Computer zu kaufen, bedeutet, daß nie mehr als nur ein paar Programme erhältlich sein werden.

Die Kosten von Programmen

Wieviel Sie bezahlen sollten? Das hängt von der Art des Programmes und Ihrem Computer ab. Kommerzielle Programme können viel teurer als Spiele sein. ROMs kosten meistens mehr als Kassetten oder Disketten. Spiele für den einen Computer können unerklärlicherweise doppelt so teuer sein wie jene für einen anderen. Prüfen Sie die Preise! Hüten Sie sich vor »unglaublichen Rabatten« – aber die teuersten Programme sind andererseits nicht unbedingt die besten!

Wo können Sie kaufen?

Immer häufiger werden Programme über die Ladentheke von Computergeschäften verkauft, aber trotzdem sind die meisten nach wie vor über den Postweg erhältlich. Der Postweg bedingt allerdings eine Verzögerung zwischen Bezahlung und Lieferung. Alle Computerzeitschriften enthalten Unmengen von Anzeigen für Programme. Sie sollten sie studieren.

Von wem sollten Sie kaufen?

Immer mehr große Firmen erscheinen auf dem Spielemarkt, aber trotzdem gibt es noch sehr viel kleinere. Qualität und Preis variieren enorm, und die großen Unternehmen haben den kleinen bestimmt nichts voraus. Einige der bekanntesten Programme sind das Produkt eines Einzelgängers. Auch wenn Sie vielleicht denken, daß große Namen gleichbedeutend seien mit einem professionellen Hintergrund: Die großen Gesellschaften kaufen oft Software von Amateuren. Lesen Sie nur einmal die Anzeigen für Programmierer! Es sind Leute wie Du und ich, die darauf antworten, und das Material, das sie liefern, ist nicht besser, als Sie es produzieren könnten! (Entschuldigung, wenn Sie selbst ein Profi sein sollten!) Wollen Sie wirklich einen Haufen Geld für Programme wie die in diesem Buch bezahlen? Bis der ganze Markt sich stabilisiert hat, ist es ratsam, vorsichtig vorzugehen und Besprechungen von Programmen zu lesen, ehe man sie kauft.
Die Zahl der Rezensionen »explodiert« gegenwärtig geradezu. Die meisten Zeitschriften testen jetzt Spiele, während bis vor kurzer Zeit sich nur die auf Spiele spezialisierten damit beschäftigten. Besonders wenn Sie kein routinierter Spieler sind, sollten Sie diese Besprechungen sorgfältig lesen. Ein Spiel kann für die eine Person aufregend und voller Bewegung sein, während es für jemand anders praktisch nicht spielbar ist.
Schauen Sie sich vor dem Kauf auch genau die Programmverpackung und den Aufbau der Anzeigen an! Oft – aber, wie gesagt, nicht immer – ist eine armselige Verpackung und eine Anzeige voller Rechtschreibfehler gleichbedeutend mit schlampiger Programmierung.

Computerklubs

Mit »Computerklub« meinen wir echte Benutzergruppen, nicht diese Pseudoklubs, die von Software-Verkäufern gegründet werden. Wenn Sie einem Computerklub beitreten, werden Sie Leute treffen, die Ihren Computer auch benutzen, und können mit ihnen Gedanken austauschen. Aber widerstehen Sie der Versuchung, Programme, die diese Leute gekauft haben, illegal zu kopieren! Anderer-

seits sollten Sie ungehemmt Listings Ihrer eigenen Programme mit denen der anderen Mitglieder austauschen. Probieren Sie deren Spiele aus, um zu sehen, welche es wert sind, gekauft zu werden! Kopieren Sie sich Programme, die Sie aus Zeitschriften oder aus Büchern eingetippt haben! Vergewissern Sie sich, daß Sie Vorteile aus irgendwelchen Mengenrabatten ziehen, die der Klub ausgehandelt hat!

Computerzeitschriften

Neue Zeitschriften erscheinen fast jede Woche, so daß es schwer ist, Ihnen eine aktuelle Liste mit Empfehlungen zu geben. Schauen Sie bei Ihrem Zeitschriftenhändler vorbei, und prüfen Sie besonders jene, die Listings für Ihren Computer enthalten! Aber ignorieren Sie die anderen nicht ganz! Sie sind sicher keine Geldverschwendung und enthalten oft nützliche Artikel und Tips. Es ist darüber hinaus sinnvoll, die Soft- und Hardware-Anzeigen zu lesen, sowie die Programmbesprechungen und andere Artikel. Sie werden recht schnell herausfinden, welche Zeitschriften sorgfältig geprüfte, gut dokumentierte und ganz allgemein *gute* Programme veröffentlichen und welche andererseits offenbar alles veröffentlichen, was ihre Leser einsenden, ob es nun gut, durchschnittlich oder vollkommen unbrauchbar ist.
Möchten Sie Ihre eigenen Bemühungen veröffentlicht sehen? Prüfen Sie die veröffentlichten Listings! Auf diese Weise bekommen Sie bald eine Vorstellung vom erwarteten Standard. Beinahe alle Zeitungen akzeptieren unverlangte Listings von ihren Lesern und bezahlen bei Veröffentlichung auch ein kleines Honorar.

Fachwortverzeichnis

Algorithmus Eine genaue Anleitung, um Schritt für Schritt ein Problem zu lösen oder ein Computerprogramm aufzubauen.
ASCII American Standard Code for Information Interchange. Dieser Code weist jedem Buchstaben, jedem Zahlenwert und jeder Taste eine Nummer zu, häufig auch grafischen Zeichen. Bei der Verwendung des ASCII-Codes in BASIC-Programmen wird die CHR$-Anweisung benutzt.
BASIC Die Programmiersprache, in der die Programme in diesem Buch geschrieben sind.
binär Eine Möglichkeit, Informationen zu verschlüsseln oder Rechnungen auszuführen, wobei nur 0 und 1 benutzt werden.
Bit Eine Binärziffer (0 oder 1), die der Computer benutzt (Binary Digit).
Byte Normalerweise eine Gruppe von 8 Bits, ein »Computerwort«.
Chip Ein elektronisches Bauteil in integrierter Form. Ein wichtiger Bestandteil von Mikrocomputern.
Cursor Kleine »Markierung« auf dem Bildschirm, die anzeigt, wo die nächste Ausgabe erscheinen wird.
Diskettenlaufwerk Das Gerät, das entweder mit Floppy-Disks oder mit einer Festplatte arbeitet und Informationen speichern und wiedergeben kann.
Floppy-Disk Flexible Scheibe mit magnetisierbarem Material, auf der Daten gespeichert werden; eine bequemere Alternative zu Kompaktkassetten.
Hexadezimalcode Arithmetisches System mit 16 Ziffern (0 bis 9 und A bis F). Wird von Maschinencodeprogrammierern benutzt. Wird häufig dazu verwendet, Computerzeichen zu definieren.
Hochsprache Leicht benutzbare Computersprache wie BASIC, COMAL, Pascal oder Forth.
kB oder *Kilobyte* (exakt 1024 Bytes). Eine Einheit, mit der die Speichergröße des Computers gemessen wird.
Maschinencode »Muttersprache« des Computers, in der programmiert werden kann. Effizient, aber schwer zu benutzen.
Kassette Computerzubehör. Ein Behälter, der in das Computergehäuse eingeschoben und mit der Hauptplatine verbunden wird. Enthält normalerweise ein Programm im ROM oder ein zusätzliches RAM.
Matrix Allgemeiner Ausdruck für einen geordneten Satz Stellen, Variablen oder Bildschirmplätze.

Modus Verschiedene Betriebsarten des Bildschirms.
Pixel Kleinstes adressierbares Bildschirmelement.
RAM Beschreibbarer Speicher im Computer (Random Access Memory, Schreib-Lese-Speicher).
Register Speicherplätze des Computers, die besondere Funktionen des Computers steuern.
ROM Nur lesbarer Speicher; vom Hersteller verwendet, um feste Informationen im Computer zu speichern (Read Only Memory).
Seite Fest definierter Speicherbereich, beispielsweise zur Speicherung des Bildschirminhalts.
Software Allgemeines Wort für Programme des Computers.
Speicher Elektronische Plätze im Computer, an denen Daten »gemerkt« werden können.
Teletext Digitale Informationen, die über Fernsehkanäle gesendet werden.
Videoschirm Bildschirm für den Output des Computers. Entweder ein Fernsehschirm oder ein spezieller Monitor.

Register

Computer verständlich

Computerspiele, Grafik & Musik

Ein Buch für alle, die ihren Computer über die normalen Anforderungen hinaus einsetzen wollen. Ob privat, im Schulbereich oder im kommerziellen Einsatz: Intelligente Spiele, Grafiken und Musik eröffnen eine neue Dimension der Anwendung. Detailliert wird auf die verschiedenen Darstellungsmöglichkeiten eingegangen. Grundsätzliche Ausführungen zur Hardware, zum Kauf und zur Erweiterung der Peripherie klären die Zusammenhänge von Programmen und notwendiger Ausrüstung.

Neben lauffähigen Programmen für die Bereiche Spiele, Grafik und Musik wird Schritt für Schritt erklärt, wie man seine eigenen Programme erstellen kann. Fehlerkatalog und ausführliches Fachwörterverzeichnis erhöhen den praktischen Gebrauchswert des Buches.

ISB N 3-8068-4305-8

T 1-28-99

FALKEN VERLAG