3 64'er AMIGA SONDERHEFT

Markt&Technik

DM 16,–
ÖS 120,–/Sfr 16,–
Lit 16000/hfl 21,–/dkr 75,–

Kurse für Aufsteiger

Basic perfekt

- Der Schlüssel zum Betriebssystem
- Effektiver programmieren mit Basic-Modulen

Kaufhilfen

- Alle Basic-Dialekte im Vergleich
- Was leistet der AC-Compiler wirklich?

Alle Spiele in Amiga-Basic

10 Spitzen-Spiele zum Abtippen

- Anpfiff: Fußball-Manager mit allen Extras
- Pingpong: 3D-Sportspiel in Perfektion
- Broker: Börsenfieber am Amiga

Alle Programme auch auf Diskette erhältlich

Ran an die Amiga*-Spielekiste!

BESTSELLER von

technicSupport

Marketing und Verlag GmbH
Bundesallee 36-37, 1000 Berlin 31

Das Große Amiga-Spielebuch

von Axel Schmidt und Jens Hertwig (Hg.), Hardcover, 256 Seiten mit vielen farbigen Abbildungen,
ISBN 3-926847-02-6

Preis: DM 49,-

Kompetente Fachautoren beschreiben ihre Lieblingsspiele, geben Tips und verraten ihre Tricks zu u.a. Flight Simulator II, Championchip Golf, Bard's Tale I + II, Test Drive, Football Manager II, Roadwar 2000, Marble Madness, Jinxter, Plutos, Defender of the Crown, Super Huey, Ports of Call, Barbarian, Interceptor, Jet, Garrison und Shanghai.

Insgesamt 32 ausführliche Spiele-Beschreibungen.

* AMIGA ist eingetragenes Markenzeichen der Commodore Büromaschinen GmbH.

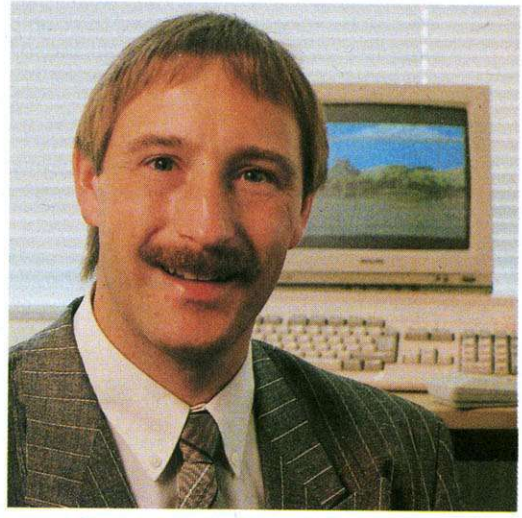
technicSupport-Bücher erhalten Sie in gut sortierten Buchhandlungen, im Computerfachhandel, bei Hertie-Fachabteilungen, im Versandhandel und bei telerent. Österreich: INTERCOMP; Schweiz: MICROTRON .

Mehr Spaß am Computer

Spielen und Programmieren — zwei Dinge, die nahezu alle Amiga-Besitzer begeistern. Sicher programmieren auch Sie vieles in Basic. Obwohl das Amiga-Basic sehr komfortabel ist, stößt man oft an Grenzen, da etliches in Basic gar nicht machbar ist. Ein ausführlicher Basic-Kurs zeigt Ihnen, wie Sie diese Grenzen zukünftig sprengen können. Durch geschickte Zugriffe auf die Betriebssystem-Routinen eröffnen Sie sich einen neuen Freiraum mit rasanten Geschwindigkeiten.

Ein zweiter Basic-Kurs zeigt, wie Sie sich eine Modul-Bibliothek zulegen. Die wichtigsten Routinen liefern wir gleich mit. Sparen Sie sich zukünftig viel Zeit beim Programmieren. Greifen Sie zu auf Ihr stets wachsendes Modul-Archiv.

Spielen macht Spaß. Eine besondere Faszination üben Computerspiele aus. Man sitzt vor dem Bildschirm und knobelt an extrem schwierigen Aufgaben oder wirbelt voller Action mit dem Joystick herum. Durch seine »Intelligenz« läßt sich ein Computer für nahezu alle Spielarten nutzen. Simulation von natürlichen Bewegungsabläufen auf dem Bildschirm oder knallhartes logisches Kombinationsvermögen sind nur zwei Beispiele dafür. Dem Amiga gaben seine Konstrukteure vor allem grafische Eigenschaften mit, die diesen Computer geradezu für Spiele prädestinieren.



Unsere Leser haben fleißig programmiert und ihre Ergebnisse eingeschickt. An dieser Stelle ein Dank an alle Amiga-Fans. Wir haben für Sie gute Spiele gesammelt und können Ihnen in dieser Ausgabe 10 tolle Spiele-Listings zur Verfügung stellen.

Ebenfalls bedanken möchten wir uns für die zahlreichen ausgefüllten Fragebögen (aus dem AMIGA-Sonderheft 2), die Sie einsandten. Im nächsten Sonderheft (Nummer 4) erfahren Sie, wer das Doppellaufwerk für den Amiga gewinnt. Durch die Auswertung der Fragebögen werden wir zukünftig die Sonderhefte noch mehr nach den Wünschen der Leser gestalten können. Dennoch sind wir auch weiterhin an Ihrer Meinung zu den Sonderheften interessiert. Schreiben Sie doch an die Redaktion »Sonderhefte«.

Ihr
Gottfried Knechtel
(Stellv. Chefredakteur)

Gottfried Knechtel

SPIELE-ÜBERSICHT

6 DIE SPIELE-HITS FÜR DEN AMIGA
Die große Anzahl an Spielen für den Amiga erschwert die richtige Wahl beim Kauf. Wir stellen Ihnen die absoluten Top-Hits vor.

GRUNDLAGEN

13 COMPILER CONTRA INTERPRETER
Welche Vor- und Nachteile besitzen Compiler und Interpreter? Neben der Antwort auf diese Frage stellen wir den AC-Basic-Compiler vor.

18 BASIC-DIALEKTE
Vier Basic-Dialekte kämpfen um die Gunst der Amiga-Besitzer. Wir helfen bei der Entscheidung, welcher für Sie der geeignete ist.

KURSE

28 AMIGA-BASIC IM HÖHENFLUG
Nutzen Sie die Betriebssystem-Routinen mit Basic. Sie glauben gar nicht, mit welchem Tempo Ihre Programme ablaufen werden. Zudem stehen Ihnen durch diese Programmiertechnik viele neue Möglichkeiten offen.

63 MODULE FÜR IHRE BASIC-BIBLIOTHEK
Eine Bibliothek mit Unterroutinen sollte jeder Programmierer besitzen. Die vorgestellten Module legen den Grundstein oder erweitern eine bereits bestehende Sammlung.

SPIELE

75 BÖRSENFIEBER AM AMIGA
Die Faszination der Börse jetzt »live« auf dem Amiga. Dieses komplexe Wirtschaftsspiel führt Sie in die Welt des Großkapitals.

84 DER PFIFFIGE FUSSBALL-MANAGER
Ein Manager-Posten ist für Sie reserviert. Spieler-Transfers, Toto-Tips, Baumaßnahmen und viele weitere Faktoren sind bei dieser Bundesliga-Simulation zu steuern.

100 PATIENCE
Taktisches Denken und einen sicheren Blick benötigen Sie bei dieser Umsetzung des beliebten Kartenspiels.

104 PING-PONG PUR
Dreidimensionales Tischtennis mit zahlreichen Effekten. Das Sportspiel für alle Joystick-Artisten.

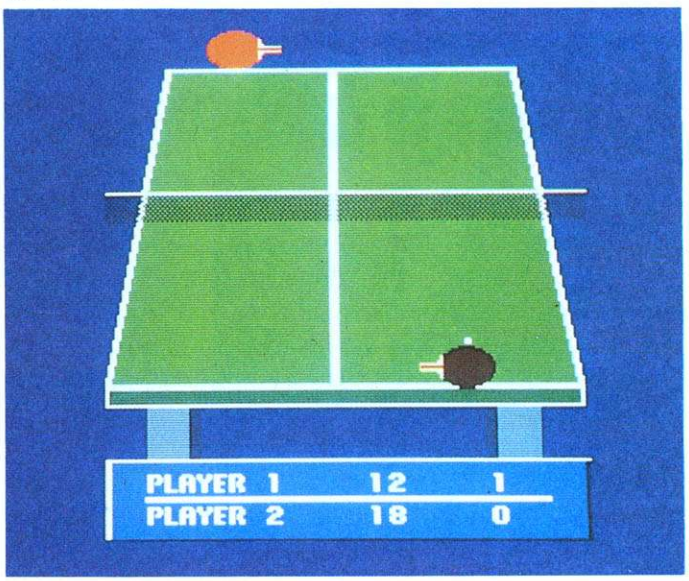
109 EIN KLASSIKER IM NEUEN GEWAND
Eine bewährte Spielidee mit ausgereifter Grafik: »Solitaire«

111 MODERNE ZEITEN
Als gestreßter Arbeiter suchen Sie den Ausgang aus einer Fabrik voller Schikanen. Mit dem Editor können Sie zusätzlich neue Level bauen.

120 GIBBER - TÖDLICHE LINIEN
Ein Reaktionsspiel mit superschneller Grafik. Entwischen Sie den Angriffen der Gibber.

Ping-Pong ist ein rasantes Sportspiel für zwei Kontrahenten. Schmettern Sie unerreichbar über's Netz oder schneiden Sie den Ball raffiniert an.

SEITE **104**



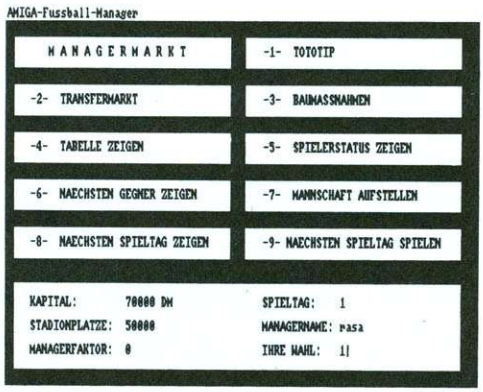
Amiga-Basic unterstützt modulare Programmierung. Erfinden Sie nicht das Rad für jedes Programm neu, bauen Sie sich Ihre Standard-Routinen.

SEITE **63**

INHALT

Mit diesem Kurs gelangen gelangen Sie direkt in Betriebssystem. Erleben Sie Basic im Höhenflug. Das Ergebnis: rasante Geschwindigkeit.

SEITE **28**

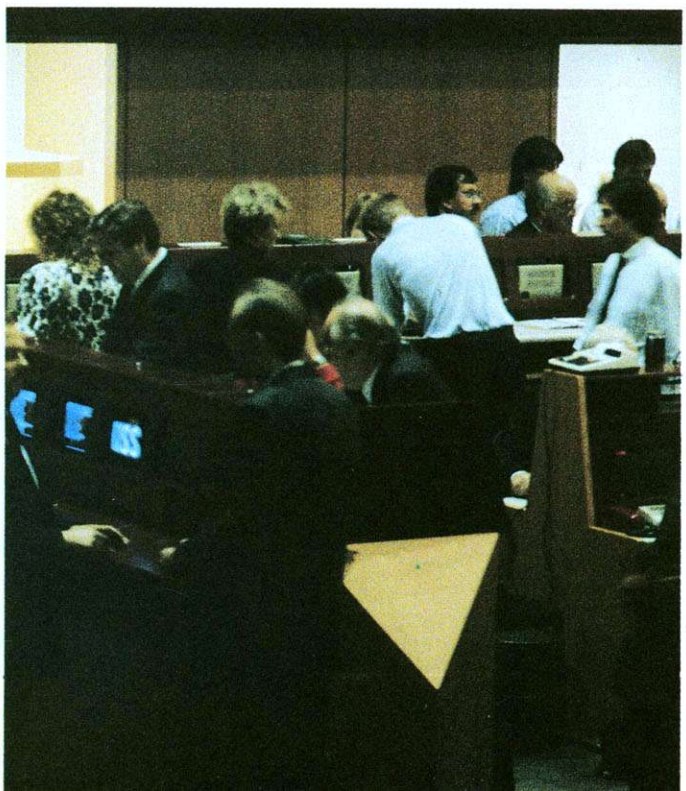


Übernehmen Sie den Posten eines Managers in der Fußball-Bundesliga

SEITE **84**

Die Börsen-Simulation für geschickte Finanzjongleure.

SEITE **75**



125 DIE RÄTSELMASCHINE
Finden Sie die im Textsalat versteckten Begriffe? Das Erstellen eigener Rätsel wird mit diesem Programm zum Spaß für die ganze Familie.

130 BODENLOS
Nur wer dem Gegner die richtige Falle stellt, bleibt bei »Hinterhalt« siegreich.

135 LAWINENGEFAHR
Bei diesem Spiel besteht erhöhte Lawinengefahr. Maximal drei Computer-Gegner machen Ihnen das Leben schwer.

TOOLS

140 VOLLBREMSUNG IM AMIGA
Die unentbehrliche Hilfe für Highscore-Jäger. Ein Tastendruck genügt, und schon ist das Spiel gestoppt.

141 DER BILDERKLAU
Grafiken aus kommerziellen Programmen sind häufig traumhaft schön. Bauen Sie solche Bilder mit diesem raffinierten Hilfsprogramm aus.

144 KREUZ & QUER
Verschaffen Sie sich eine präzise Übersicht über lange Basic-Programme. Labels, Konstanten und Variablen werden von »XRef« angezeigt.

TIPS & TRICKS

147 DIE SCHATZKISTE FÜR BASIC-FREAKS
Zahlreiche hilfreiche Kniffe verhelfen zur optimierten Programmierung.

154 SCHUMMELN SIE MIT
Viele Spiele sind nur mit kleinen Schummeleien zu bewältigen. Hier finden Sie eine ganze Reihe davon.

SONSTIGES

3 EDITORIAL

157 GURU-MEDITATION

158 BÜCHER

159 CHECKSUMMER

162 IMPRESSUM

162 INSERENTENVERZEICHNIS

Alle Programme aus Artikeln mit einem -Symbol finden Sie auch auf der Programmservice-Diskette zu diesem Sonderheft

Bei den zahlreichen Spielen für den Amiga verliert man leicht den Überblick. Herausragende ältere Titel sind wegen zahlreicher Neuerscheinungen schnell vergessen. Wir stellen Ihnen die Spiele vor, die unserer Meinung nach die absolute Elite darstellen.

Die Spiele-Hits für

Kein anderer Computer ist für Spiele besser geeignet als der Amiga. Die Hardware bietet zahlreiche Leckerbissen, die für die Programmierer von Spielen geradezu ideal sind. Wenn dann entsprechend gute Software entsteht, ist Begeisterung vorprogrammiert. Jeder, der einmal ein Superspiel auf dem Amiga erlebt hat, kennt das Ergebnis: Faszination.

Das Angebot an Spielen ist im letzten Jahr gewaltig angewachsen. Das hat für die Amiga-Besitzer zwei Seiten: Zum einen ist es toll, daß ein riesiges Angebot besteht. Zum anderen ist leider fast unmöglich, den Überblick über die Spieleflut zu behalten.

Genau dieses Problem möchten wir Ihnen erleichtern. Wir haben uns zusammengesetzt und die Spiele ausgewählt, die unserer Meinung nach die absolute Spitze darstellen. Dabei kam es uns nicht unbedingt darauf an, daß ein Spiel brandneu ist. Im Vergleich zu einigen »müden« Neuerscheinungen können Klassiker, wie »Shanghai« durchaus überzeugen.

Bei der Auswahl haben wir Spiele aus verschiedenen Sparten berücksichtigt. Nur so können wir sicher sein, daß für jeden Geschmack ein idealer Kandidat vorgestellt wird. Adventures, Geschicklichkeit, Ballerspiele, Sportspiele und Simulationen haben wir untereinander verglichen. Die Auswahl muß trotzdem subjektiv bleiben, bei Spielen gibt es für uns nunmal ein Hauptkriterium zur Bewertung: den Spaß. Aber auch Bedienungskomfort, Verständlichkeit des Spielprinzips, und Abwechslung sind Merkmale, die bei dieser subjektiven Auswahl den Ausschlag gaben.

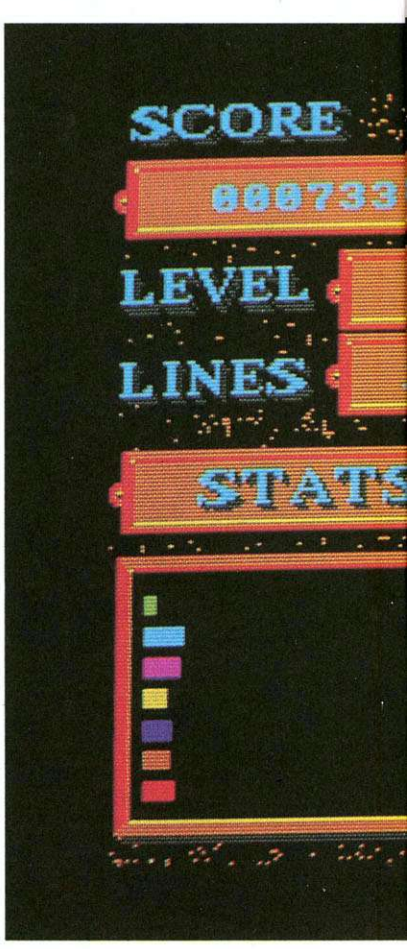
Einige zusätzliche Tips möchten wir Ihnen für den Spielekauf geben. Sie sollten auf jeden Fall die Möglichkeit nutzen, im Geschäft den Wunschkandidaten auszuprobieren. Oft merkt man sehr schnell, ob das Spiel wirklich das richtige ist. Vielleicht nehmen Sie bei Action- und Geschicklichkeitsspielen einfach den eigenen Joystick mit. Es gibt freundliche Händler, die einen Test mit dem eigenen Equipment gestatten.

Bei Adventures sollten Sie den Wortschatz genau untersuchen. Versteht der »Parser« (der sorgt dafür, daß ein Programm Ihre Eingaben versteht) komplizierte Vokabeln, werden ganze Sätze erkannt und korrekt analysiert? Außerdem ist auch bei Abenteuerspielen eine gelungene Grafik oft das erfreuliche Quentchen mehr.

Natürlich kommen fast täglich neue Spiele für den Amiga heraus. So erreichte uns nach Redaktionsschluß eine Umsetzung von »Elite« sowie verschiedene andere potentielle Kandidaten für unsere Übersicht. Leider konnten wir diese nicht mehr berücksichtigen.

Wenn Sie sich über die aktuellen Highlights informieren möchten, bieten die Kollegen vom AMIGA-Magazin Unterstützung. Dort werden in jedem Monat interessante Neuerscheinungen unter die Lupe genommen. Zusätzlich bietet die Zeitschrift Happy-Computer ebenfalls monatlich mit der Beilage »Powerplay« zahlreiche Informationen über den gesamten Spielemarkt. Auch dort findet jeder begeisterte Spieler eine hervorragende Unterstützung bei der Wahl eines Spitzentitels aus dem reichen Angebot.

(Andreas Lietz/rs)



den Amiga

Interceptor

Wer schon immer einmal ein Kampfflugzeug mit Höchstgeschwindigkeit durch die Lüfte steuern wollte, kommt an »Interceptor« nicht vorbei. Dieses Spiel simuliert wahlweise eines der beiden Kampfflugzeuge »F/A-18« und »F-16«. Beide Flugzeuge kann man entweder einfach zum Spaß über der Bucht von San Francisco fliegen lassen oder mit ihnen einen Luftkampf mit feindlichen Flugzeugen ausfechten, wozu dem Spieler ein Maschinengewehr und verschiedene Lenkraketen zur Verfügung stehen. Vor dem Kampf heißt es allerdings erst einmal üben, üben und nochmal üben. Ein zweites Flugzeug (Trainer) zeigt dem unerfahrenen Piloten, wie er die verschiedenen Flugmanöver meistert.

Als Start- und Landeplätze stehen ein Flugzeugträger sowie verschiedene Flughäfen bereit. Dort kann auch nachgetankt und Munition aufgefüllt werden.

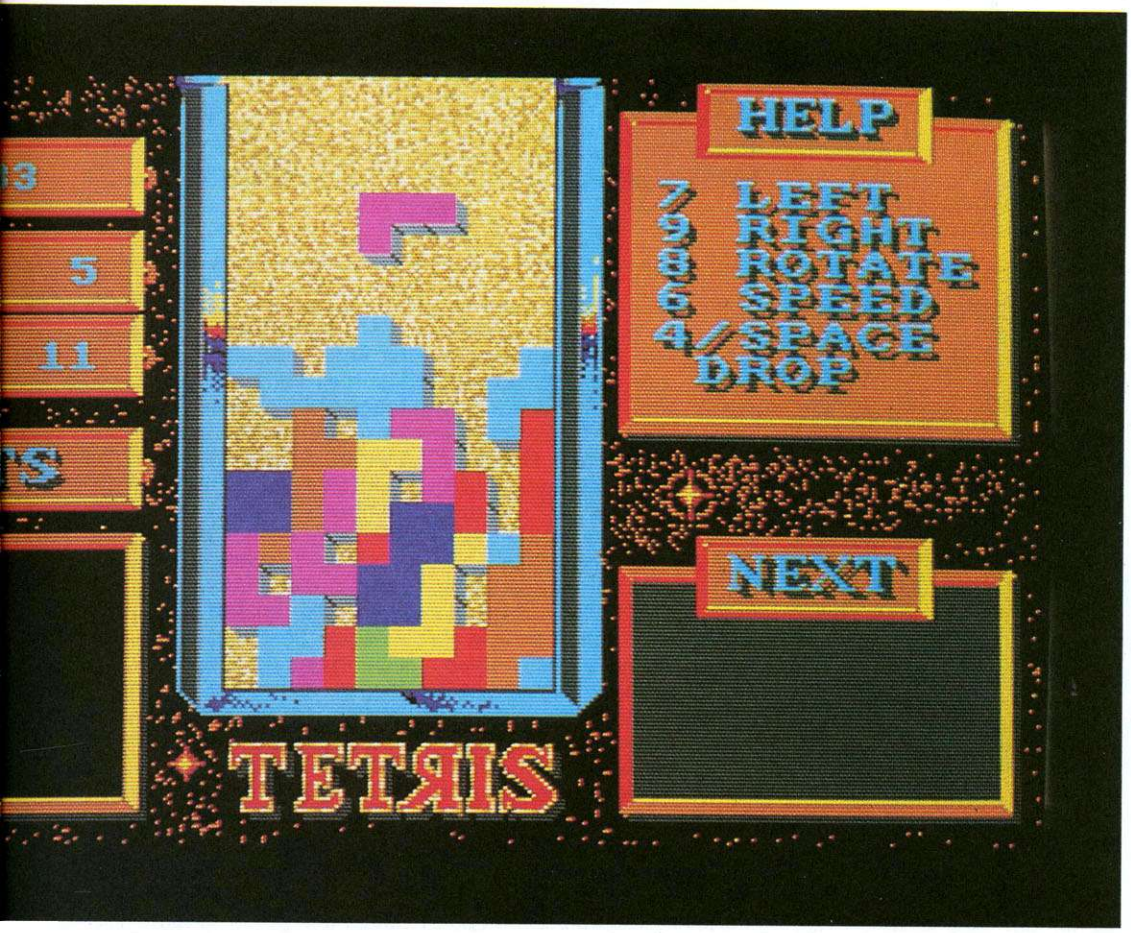
Das Wichtigste an Interceptor ist die geradezu unglaublich schnelle Grafik, die den »Flight Simulator II« um Längen schlägt. Im Gegensatz zu einem richtigen Piloten kann sich der Spieler sein Flugzeug sogar von außen aus verschiedenen Blickwinkeln anschauen. Mit Hilfe eines »Head-Up-Displays« hat der Pilot die wichtigsten Daten des Flugzeuges ständig im Blick. Unterschiedlich schwere Missionen müssen der Reihe nach erfüllt werden. Bei Erfolg gibt's einen Eintrag in das persönliche Logbuch. Etwas gewöhnungsbedürftig sind am Anfang die vielen verschiedenen Tastaturbefehle, die zur Steuerung aller Funktionen nötig sind. Insgesamt ist Interceptor aber eine rundum gelungene Kampfflugzeug-Simulation.

Tetris

Wie viele begeisternde Spiele basiert »Tetris« auf einem ganz einfachen Spielprinzip. Vom oberen Bildschirmrand fallen verschiedene geometrische Objekte, also zum Beispiel ein Stab, ein T-Stück oder ein Quadrat. Jeder dieser Körper sinkt langsam nach unten, bis er den unteren Rand des Spielfeldes erreicht. Der Spieler muß nun jedes dieser Objekte so in die bereits vorhandenen einpassen, daß zwischen diesen keine Lücken frei bleiben. Dazu steuert er den gerade herunterfallenden Körper mit dem Joystick an eine günstige Position. Durch Druck auf den Feuerknopf lassen sich die Objekte zusätzlich drehen, um sie richtig einzupassen.

Das Spielfeld ist in Zeilen aufgeteilt. Ist nun eine dieser Zeilen lückenlos gefüllt, verschwindet diese vom Bildschirm. Das ist auch dringend nötig, denn die herunterfallenden Objekte wachsen zu einem immer höheren Berg, der die Zeit zum Nachdenken immer kürzer werden läßt... und im nächsten Level fallen die Objekte schon schneller. Für jede verschwundene Zeile gibt es Punkte.

Tetris ist ein interessantes Spiel, das rasches Denken und blitzschnelle Reaktion erfordert — langweilig wird einem bei diesem Spiel jedenfalls nicht.



Hybris

Mit »Hybris« kommt Spielhallenstimmung auf: Dieses Actionspiel läßt mit seiner hervorragenden Grafik so manchen Spielautomaten »alt aussehen«.

Das Ziel des Spiels ist einfach: Der Spieler fliegt mit seinem Raumschiff über eine (von oben dargestellte) Landschaft und schießt auf alles, was das Raumschiff gefährdet. Angegriffen wird er vom Boden und von verschiedenen Raumschiffen aus; die Bodenstationen lassen sich zwar mit der Bordkanone zerstören, sind aber oft mit »Laser-Stoppern« ausgestattet, die dem Spieler das Leben schwermachen.

Gelegentlich bekommt man Extra-Waffen, die auch dringend nötig sind: Die Monster, die an manchen Stellen lauern, sind mit dem normalen Bordlaser fast unmöglich zu besiegen.

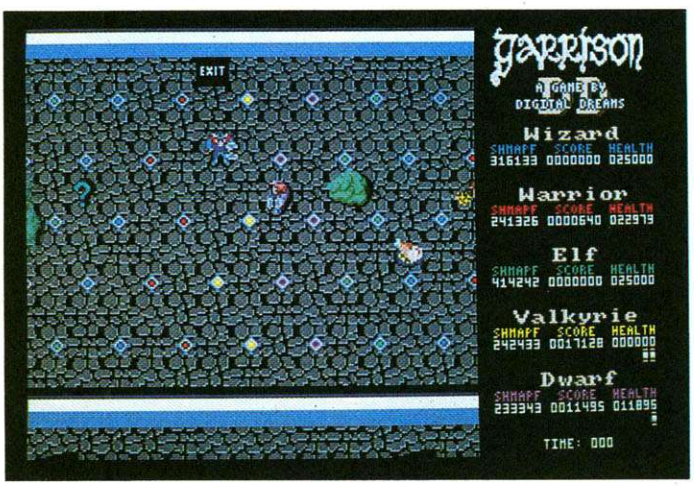
»Hybris« bietet eine große Vielfalt von Gegnern, bei denen man jeweils unterschiedlich vorgehen muß. Zusammen mit dem perfekten Scrolling tragen die gute Musik und die Soundeffekte dazu bei, daß dieses Spiel jedem Ballerspiel-Fan eine Menge Spaß machen wird.

Dieses reinrassige Action-Spiel besitzt sogar eine »Continue«-Funktion, mit der man nach dem Spielende weiterspielen kann — fast möchte man am Amiga nach dem Geldeinwurf suchen!

Wizball

Um die vom bösen Zark ihrer Farben beraubte Wizwelt wieder einzufärben, muß der Spieler, als kleiner Ball getarnt, durch die verschiedenen Ebenen springen, um die passenden Farben zu finden. Nun gibt es aber viele garstige Kobolde, die etwas gegen die Einfärbung der Wizwelt haben und dem Ball das Leben schwer machen. Dies ist der Handlungsrahmen bei »Wizball«. Die Steuerung der kleinen Kugel, die ständig auf- und abspringt, ist eigentlich ganz einfach: Drückt man den Joystick nach rechts oder links, so dreht sie sich in die entsprechende Richtung. Wenn die Kugel den Boden berührt, springt sie also in die jeweilige Drehrichtung. Auf die Kobolde kann der Ball schießen, aber manchmal ist eine Kollision unvermeidbar, da die Kugel nach dem Absprung ihre Richtung nicht ändern kann...

So richtig interessant wird das Ganze, wenn man die Kobolde durch einen gezielten Schuß in Kristallkugeln verwandelt, die der Kugel zusätzliche Fähigkeiten verleihen. Auf diese Weise holt man sich auch die Katze »Catalite« herbei, die beim Einsammeln der Farben sehr hilfreich ist. Am meisten Spaß macht das mit zwei Spielern. Die sehr gute Grafik und schöne Soundeffekte runden dieses gelungene Action-und Geschicklichkeitsspiel ab.



Garrison

In diesem Action-Spiel geht es um die Rettung einer schönen Prinzessin namens Angelique. Das Wundermittel zu ihrer Rettung und Heilung einer schweren Krankheit ist in der Burg eines finsteren Magiers zu finden. Also brechen fünf Helden auf, sich durch 128 verschiedene Etagen voller Monster zu kämpfen. Das Überleben in dem Schloß ist ganz schön schwierig, besonders wenn man dabei allen möglichen unliebsamen Gestalten begegnet. »Garrison« bietet deshalb etwas Einmaliges: Spielt man es zu zweit, können sich die beiden gerade spielenden Figuren gegenseitig helfen. Sie können aus den fünf verschiedenartigen Spielfiguren ausgesucht werden, die alle ihre Stärken und Schwächen haben. Das ist auch dringend nötig, denn einige Gegner lassen sich mit roher Gewalt bekämpfen, andere aber nur mit einem Zauberspruch. Besonders wichtig sind für die Spieler auch die verschiedenen im Schloß zu findenden Zaubersprüche und Schlüssel. Die unterschiedlichen Eigenschaften der Kämpfer kann man durch das Aufsammeln von Flaschen voll magischem Elixier verbessern. Man sollte einen Trank zur Steigerung der Schußkraft zum Beispiel seinem Elf zukommen lassen, da dieser zwar sehr schnell zu Fuß, aber auch sehr schwach ist. Während Speed-Potions eher für den schweren und langsamen Krieger bestimmt sind. Die Übermacht der Gegner macht dieses Actionspiel an vielen Stellen recht schwierig. Die 128 verschiedenen Level, die nach dem Programmstart in einer zufälligen Reihenfolge geladen werden, lassen keine Langeweile aufkommen. Eine High-Score-Liste auf Diskette für jede einzelne Spielfigur ergänzt den guten Eindruck.

Profilaufwerk 3,5"

Metallgehäuse ● einstellbare Lauwerknummer mit Displayanzeige ● digitale Trackanzeige ● Write Protect am Laufwerk schaltbar ● abschaltbar ● durchgeschleifter Bus
1 Jahr Garantie
Super ALCOMPreis **329,-**

Laufwerk 5,25"

40/80 Track ● Laufwerksbus durchgeschleift ● abschaltbar ● einstellbare Adressen ● MS- DOS- kompatibel ● mit Diskchange

Super ALCOMPreis	**298,-**
HD 1,6 MB (umschaltbar)	**318,-**
Amigafarbene Blende	**+10,-**
Write Protect Schalter	**+15,-**

Gemischtes Doppel 3,5/5,25"

einzeln ein-/abschaltbar ● einstellbare Laufwerksnummern mit Anzeige ● durchgeschleifter Bus ● bei 5,25" 40/80 Tracks umschaltbar ● Metallgehäuse ● 1 Jahr Garantie
Super ALCOMPreis **548,-**

500er Speichererweiterung

Für 512k zusätzliches RAM ● alle RAM-s gesockelt ● selbstkonfigurierend ● abschaltbar ● Uhrenschaltung auf Platine mit Akku- bzw. Batteriepufferung nachrüstbar

Komplett mit 512k	**Preis auf Anfrage**
Superpreis mit Uhr	**Preis auf Anfrage**
Bauteilesatz für Uhr ohne Akku	**24,-**
Leerplatine mit Stecker	***39,-**

*mit Schaltplan und Bestückungsliste

Laufwerkanschlußkabel

Zum Anschluß von Laufwerken an alle Amigas ● mit Ansteuerelektronik

Für 3,5" Laufwerk	**39,-**
Für 5,25" Laufwerk	**49,-**

Steckplatzerweiterung 3-fach für Laufwerke

Jeder Steckplatz abschaltbar und einstellbare Laufwerksnummer ● Steckplatzerweiterung direkt am Amigagehäuse ● Dadurch keine Kabellängenprobleme
Anschlußfertig zum Super ALCOMPreis **49,-**

Soundsampler

Für alle Amiga's mit Software ● Type bei Bestellung bitte angeben ● 8-Bit Datenbreite ● Betrieb am Parallelport (Druckerport) ● Mit Vorverstärker für Micro-Anschluß (Cinch-Buchsen) ● Musik- und Sprachdigitalisierung möglich ● Arbeitet mit fast allen Digitizer-Programmen ● Formschönes Gehäuse
Super ALCOMPreis **79,-**

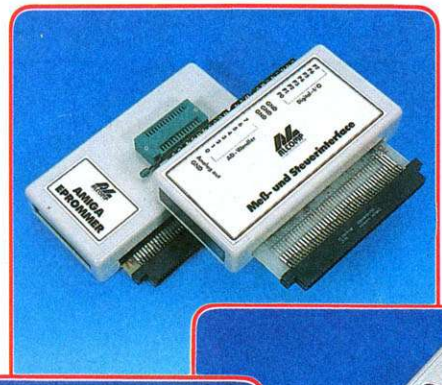
Sampler Studio

● Professionelles Sampler-Programm ● 4-Kanal-Technik ● speichern auf 4 Disketten hintereinander möglich ● alle gängigen Formate (IFF, Data, Future) ● Echtzeitdisplay mit Zoomfunktion ● viele Verfremdungsmöglichkeiten ● Echo, Hall, Reverse **69,-**
Paket: Sampler + Software **129,-**

MIDI - Interface

4 Kanäle einschließlich 1 Thru ● Optische Datenanzeige ● Formschönes Gehäuse
Wahnsinnspreis von nur **89,-**

ausgereifte Ingenieurleistung ● 14 Tage Umtauschrecht ● fast alle IC's gesockelt ● nur professionelle Leiterplatten ● Bauteile namhafter Hersteller ● mit Bedienungsanleitung



3,5" Laufwerk

Für alle Amiga's ● einstellbare Gerätenummer ● abschaltbar ● Metallgehäuse ● superflach ● 1 Zoll (2,54cm) ● durchgeschleifter Bus ● TEAC Laufwerk
1 Jahr Garantie

komplett anschlußfertig	**239,-**
Amigafarbene Blende	**+10,-**

Basislaufwerke

1 Jahr Garantie

TEAC FD 135 FN 3,5" 1MB superslimline	**218,-**
TEAC FD 55 GFR 5,25" 40/80 Tracks	**239,-**
Amigafarbene Blende	**+10,-**
1,6 MB Diskchange	**259,-**
3,5" Gehäuse	**25,-**
5,25" Gehäuse	**25,-**
Gehäuse für "Gemischtes Doppel"	**65,-**

Bootselector **19,90**

Amiga Eprommer

- Für A 500/1000
- Expansionsportanschluß
- Für EPROM's 2764-27011 (8K-128K) Alle A-Typen und CMOS-Typen
- Funktionen:
  LEERTEST LADEN VON DISK
  VERGLEICHEN SPEICHERN AUS DISK
  AUSLESEN HEXDUMP
  BRENNEN
- vier Programmieralgorithmen 50mS/Byte - Superschnell 64K-1,5 min
- Programm zum Generieren und Brennen von Kickstarts direkt von Diskette oder aus ROM
- Mit Software + Gehäuse **225,-**

Meß- und Steuerinterface

- 8 ADC-Kanäle 0-2,55V in 0,01V Stufe
- 1 DAC-Kanäle 0-2,55V in 0,01V Stufe Genauigkeit- 1,5 LSB
- 8 frei programmierbare TTL-I/O Kanäle
- Mit Gehäuse, Anschlüße auf Schraubklemmen
- interne Referenzspannung
- Expansionsanschluß
- Einfache Programmierung in Basic möglich Multitasking tauglich
- incl. DEMO-Software auf 3,5" Diskette

239,-

Trackanzeige

Für DF0-DF3 einstellbar ● für alle Laufwerke (3,5"/5,25") ● Laufwerkbus durchgeschleift ● mit Gehäuse
Super ALCOMPreis **69,-**

Einführungsangebot Amiga - Harddisks

HD-Interface A 2000	**198,-**
HD-Interface A 500/A 1000	**249,-**
Platte 20 MB A 2000	**798,-**
30 MB A 2000	**898,-**
40 MB A 2000	**1 098,-**
65 MB A 2000	**1 348,-**
Platte A 500/A 1000	
20 MB	**898,-**
30 MB	**998,-**
40 MB	**1 248,-**
65 MB	**1 498,-**

Vokabeltrainer

2500 englisch-deutsche Vokabeln incl. Hilfssatz ● Merkfunktion ● komfortabler Editor zur Vokabelverwaltung ● Wörterbuch zum Dateiendurchsuchen **59,-**

Selbstbootende Harddisk für Amiga ohne PC-Karte!

Die Amiga-Festplatte von ALCOMP:
● Selbstbootend wie "Card" oder "Rad"! ● Als Einbau-Festplatte für den "Amiga 2000" ● Als Externe Einheit für den "Amiga 500" und 1000 mit Gehäuse, eigenem Netzteil und Erweiterungsanschluß ● Erhältlich mit 20, 30, 40 und 65 Megabyte ● Kopiert 1 Megabyte in unter 4 Sekunden ● Speichert schneller als "1.2-Ramdisk" ● Läuft mit "FastFileSystem" ● Einfach einstecken, Formatieren, "Mountlist" und "Startup-Sequence" ändern und los geht's!
Entwickler: Stephan und Stefan
Für den Selbstbau: Harddisk-Interface incl. Steuersoftware ● Anschluß mit Slot für Omti-Controller

Frischer Wind auf dem Hardwaresektor

ALCOMP
COMPUTERHARDWARE

Kickstartumschaltung

Bauen Sie die anderen Kickstart-Versionen in Ihren Amiga 500 ● Einfacher Einbau ohne Löten ● für Original-Kickstart-ROM und 2 zusätzliche Versionen auf EPROM ● EPROM-Programmierservice auf Anfrage

SuperALCOMPreis	**59.-**
Kickstartversion auf EPROM's	**120.-**

Userport + Experimentierkarte für Expansionport

Mit Lochraster und 2 x 6522 Ports

Leer	**59.-**
komplett aufgebaut	**89.-**

Wir suchen ständig Hardware-Entwicklungen. Wir garantieren gute Umsatzprovisionen und ehrliche Abrechnung

kostenloses Info anfordern!!!

Bestellung und Versand

ALCOMP
A. Lanfermann
Lessing Str. 46
5012 Bedburg
Tel. 0 22 72/15 80

Nachnahmeversand NN-Spesen 7.50 DM b. Vorkasse 3.- DM. Auslandsbestellungen: Nachnahmeversand NN-Spesen 10.- DM b. Vorkasse 5.-DM. Wir liefern Ihnen auf Ihre Rechnung und Gefahr zu den Verkaufs- und Lieferbedingungen des Elektronikgewerbes.
Postgiroamt Köln
(BLZ 370 100 50) 275 54-509

Entwurf + Druck: PLUM, Düren-Oberzier, TEL. 0 24 28/7 11

Black Lamp

Jack, der Hofnarr des Königs Maxim von Allegoria, hat Liebeskummer. Nicht, daß ihn die Angebetete nicht wollte, aber sie ist ausgerechnet die Tochter seines Chefs — die königliche Hoheit Prinzessin Grizelda. Doch bevor Jack sie heiraten kann, muß er eine schwierige Aufgabe erfüllen. Er soll die neun Wunderlampen wieder zurückbringen, die von bösen Mächten geraubt wurden und von Monstern und Drachen bewacht sind.

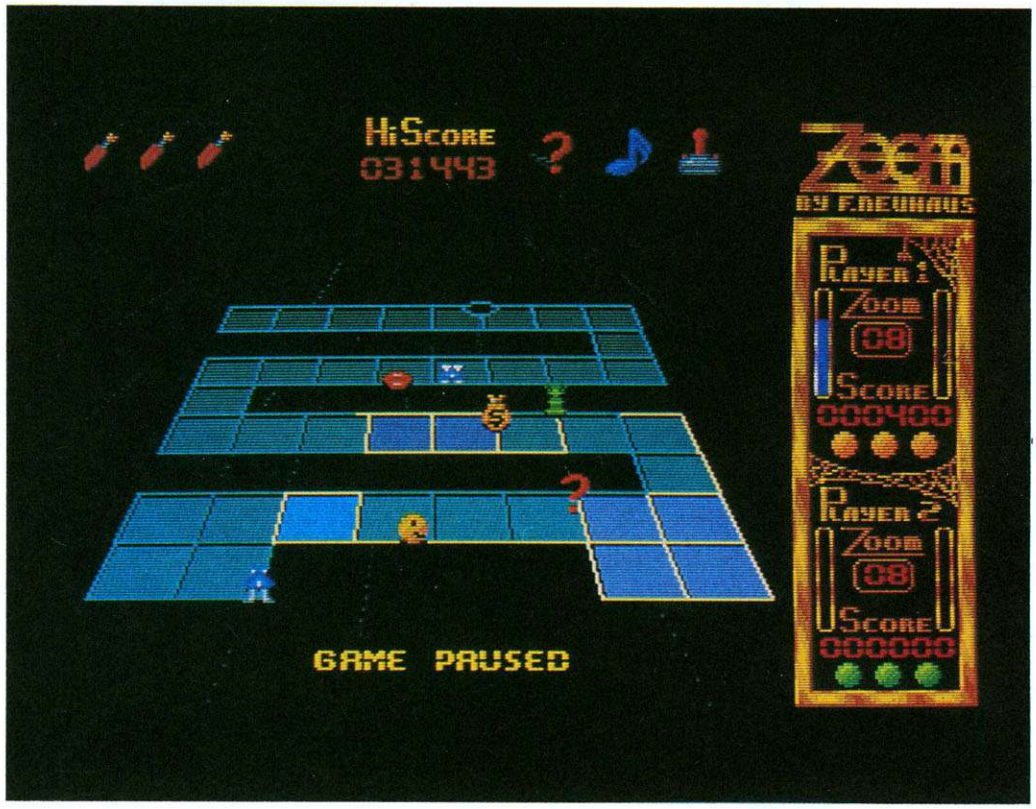
»Black Lamp« ist ein »Jump-and-run«-Spiel. Der Held Jack muß in einer Vielzahl von verschiedenen Szenen herumlaufen, springen und sich mit Schüssen aus seinem magischen Gürtel gegen die Monster verteidigen. Dabei ist er natürlich ständig auf der Suche nach einer der Wunderlampen. Wenn Jack getroffen wird, verliert er Energie — findet er etwas zu Essen oder zu Trinken, wird sie wieder aufgefüllt. Am schwierigsten ist es, an die schwarze Lampe zu kommen: Sie wird von mindestens einem Drachen unerbittlich verteidigt.

Grafisch ist »Black Lamp« recht gut gemacht, die verschiedenen Monster sorgen für reichlich Abwechslung.

Zoom

Eigentlich ist »Zoom« ganz einfach zu spielen: Bei diesem Geschicklichkeitsspiel muß die Spielfigur, die ein wenig an den berühmten »Pacman« erinnert, auf einem Linienmuster umhergesteuert werden, um die Quadrate zwischen den Linien einzufärben. Doch leider bewegen sich auf dem Gitter eine Menge merkwürdiger Gestalten, die dem Spieler an den Kragen wollen. Wirklich unangenehm ist ein roter Mund, der der Spielfigur ständig hinterherläuft. Erwischt der Mund die Spielfigur, »frißt« er diese einfach auf! Zu allem Überfluß färbt ein grüner Stengel die Linien wieder um. Aber es gibt auch Gestalten, mit denen man gerne zusammentrifft — einige bremsen beispielsweise die Bösewichte, andere bringen zusätzliche Punkte.

Die dreidimensionale Darstellung des Spielfeldes ist gut gelungen. Den Vorspann sollten Sie zudem einmal in voller Länge genießen.



Summer Olympiad

In der Kategorie Sportspiele tat sich auf dem Amiga lange Zeit wenig. Magere Umsetzungen von anderen Computern boten nicht den Spaß, der mit den allseits bekannten und geschätzten Qualitäten des »Grafik-Wunders« zu erwarten war.

Die Sommer-Olympiade behebt diesen Mangel. Die Wettbewerbe bei dieser Olympiade sind: Turmspringen, Dreisprung, Fechten, Hürdenlauf und Tontaubenschießen. Überdurchschnittliche Grafiken zeichnen die einzelnen Übungen sowie die Eröffnungssequenz aus.

Herausragend ist die Darstellung des Hürdenlaufes. Ein »Kameraschwenk« vor Beginn des Rennens verwischt die Grenzen zwischen Spiel und Fernseherlebnis. Spielerisch sind besonders das Fechten und das Tontaubenschießen gelungen. Nur mit viel Übung beherrschen Sie das technisch schwierige Tontaubenschießen.

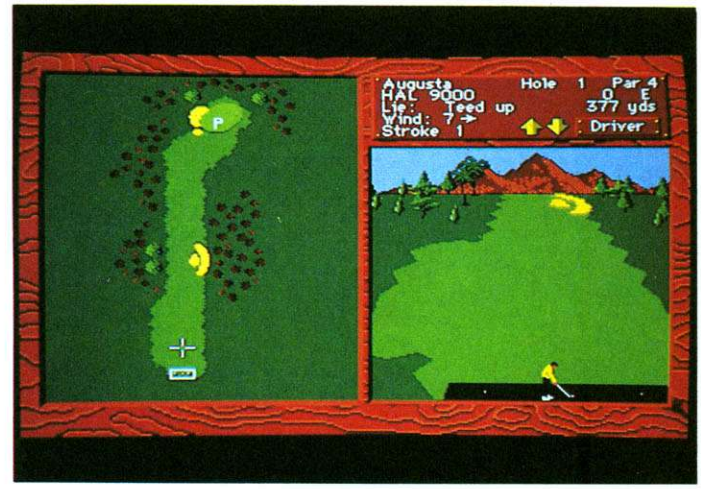
Wer Sportspiele liebt, liegt hier genau richtig.

World Tour Golf

Nicht jeder kann es sich leisten, zu den exklusivsten Golfplätzen der Welt zu fahren und dort eine Runde zu spielen. Viel billiger ist »World Tour Golf« für den Amiga.

Sobald der Spieler am Abschlagpunkt steht, sieht er in sehr guter Grafikdarstellung auf dem Bildschirm, wohin er den Ball befördern muß. Das Fadenkreuz gibt dabei die Richtung des Schlages an. Nun heißt es, erst einmal den richtigen Schläger aussuchen und dann mit einer Folge von vier gut gesetzten Mausklicks den Ball zu schlagen, was am Anfang einige Übung erfordert. Auf dem Golfkurs selbst begegnen einem eine Vielzahl von Hindernissen. Da heißt es, über Wasserflächen zu spielen, Bäume zu umgehen und gelegentlich den Ball aus einem Sandbunker zu holen.

Besonders interessant ist es natürlich, selber einen Golfkurs zu entwerfen. Mit Hilfe eines »Golfplatz-Editors« geht das schnell und komfortabel. Insgesamt ist World Tour Golf also nicht nur für Golf-Freunde sehr zu empfehlen.



Ports of Call

Bei »Ports of Call« werden Sie zum Reeder, der mit 5 Millionen Dollar Startkapital soeben in die Schiffahrt eingestiegen ist und nun mit dem Transport von Seefracht viel Geld verdienen will. Eine Menge Entscheidungen sind da zu fällen. Nicht jeder Heimathafen ist gleich gut und zwischen den angebotenen Schiffen bestehen große Unterschiede.

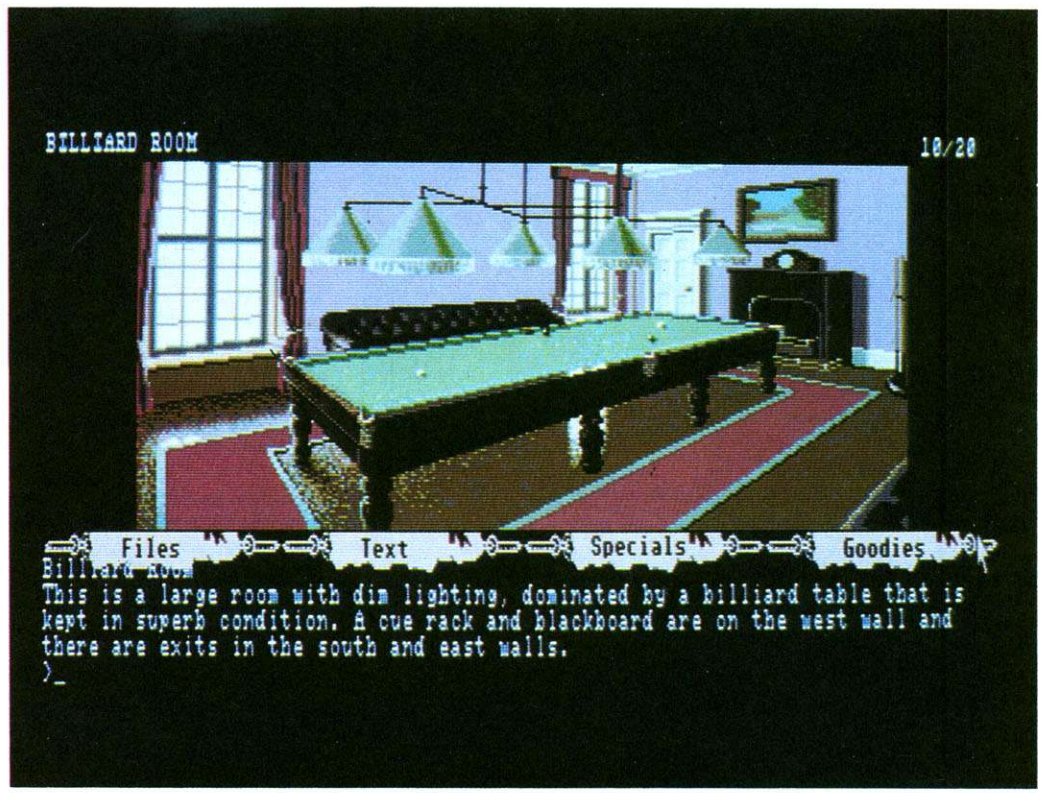
Auf dem Weg ist man vor Überraschungen nicht sicher. Von der Rettung eines Schiffbrüchigen bis zum nötigen Umfahren eines Eisberges kann alles passieren. Wer gut ist, kann viel Geld verdienen und seine Flotte ausbauen.

Die wirklich hervorragende Grafik und die Liebe zu kleinen Details machen Ports of Call zu einer sehr realistischen Simulation. Besonders interessant ist das Spiel mit mehreren Spielern (bis zu vier Spieler sind möglich), die einander natürlich harte Konkurrenz machen.

Das »Ein- und Ausparken« in den Häfen wird allerdings nach einiger Zeit etwas langweilig. Gegen Bezahlung kann man sich eines Schleppers bedienen. Sehr praktisch ist, daß Spiel und Anleitungen in Deutsch verfaßt sind, was den Einstieg um einiges erleichtert.

Guild of Thieves

Um in die Gilde der Diebe in Kerovnia aufgenommen zu werden, müssen Sie Ihre Fähigkeiten als Meisterdieb unter Beweis stellen. Sie brechen auf zu einer Insel, die an den verschiedensten Orten Schätze beherbergt. Diese Schätze sind jedoch nicht immer auf den ersten Blick als solche erkenntlich. Sie müssen alle wertvollen Gegenstände auffinden und entführen, ohne daß ihre Besitzer etwas davon merken. Doch das ist gar nicht so einfach, denn in ein Schloß muß man erst einmal hineinkommen, und nicht alle Wesen auf der Insel sind Ihnen freundlich gesinnt. Die einzelnen Schätze kann man natürlich nicht ständig mit sich herumtragen und muß sie deswegen in einem Banksafe deponieren. Letzte Aufgabe bevor Sie dem Meisterdieb die Erfüllung ihrer Aufgabe bestätigen können, ist es dann eben diese Bank auszuräumen.



Pinball Wizard

Wer keine Lust hat, in einen richtigen Flipper einen Stapel Markstücke zu investieren, sollte den »Pinball Wizard« ausprobieren. Diese Amiga-Flippersimulation bereitet außerordentlich viel Spaß. Das Wichtigste bei einem Flipper ist natürlich, die Kugel im Spiel zu halten — mit zwei großen und einem kleinen Flipper ist das mit etwas Übung durchaus zu schaffen. Doch die hohe Kunst ist es, durch geschicktes Lenken der Kugel die vielen verschiedenen Spezialfunktionen von »Pinball Wizard« zu aktivieren. Geübte Spieler können sogar mit zwei Bällen gleichzeitig spielen. Flipper-Profis sind vor allem auf Bonuspunkte aus, die die Punktzahl in ungeahnte Höhen treiben können.

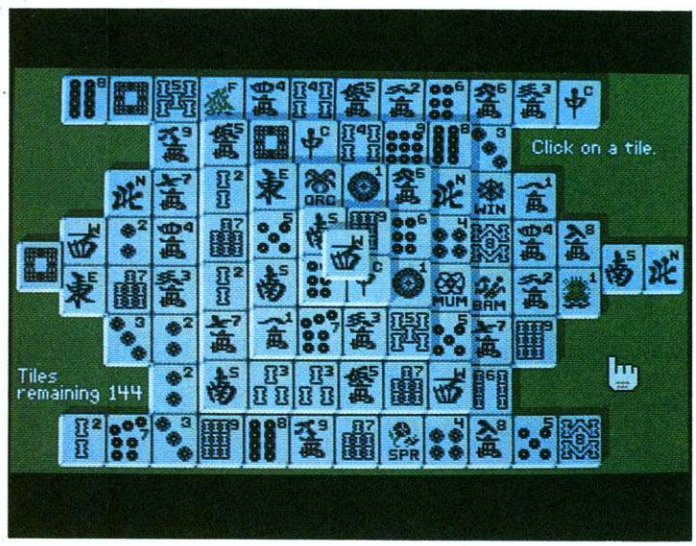
Die Grafik und vor allem die digitalisierten Sounds des Programms sind hervorragend — vieles hört sich wie bei einem echten Flipper an. Die Kugel bewegt sich mit einer ganz schönen Geschwindigkeit, was schnelle Reaktionen unbedingt nötig macht.

Durch die verschiedenen »Spezialeffekte« bleibt das Spiel auch für lange Zeit interessant. Besonders erfreulich ist, daß die gesamte PAL-Auflösung des Amiga ausgenutzt wird.

Die Drachen von Laas

Haben Sie sich schon immer über die komplizierten englischen Texte von Adventures geärgert? Dann ist »Die Drachen von Laas« genau das Richtige für Sie. Dieses Adventure, das auch Rollenspiel-Elemente enthält, ist vollständig in deutsch geschrieben und versteht über 2000 Wörter. Damit entfällt für Amiga-Anwender, deren Englisch-Kenntnisse nicht perfekt sind, das Blättern im Wörterbuch.

Die beiden Freunde Smirga und Aszhanti halten es zu Hause nicht mehr aus. Sie wollen weg aus Hyllok, um etwas neues kennenzulernen und sich in Kampf und Magie ausbilden zu lassen. Um Geld zu verdienen, führen sie für verschiedene Leute Aufträge aus, die nicht ganz einfach zu meistern sind. Schließlich treffen sie auf die Drachen von Laas — hier stellt sich den beiden eine fast unüberwindbare Aufgabe. Nur mit guten Einfällen können die beiden Freunde diese Herausforderung meistern.



Shanghai

»Shanghai« ist die moderne Version des alten asiatischen Brettspiels Mah-Jongh. Am Anfang des Spieles hat der Spieler dabei einen regelmäßig angeordneten Stapel mit Spielsteinen vor sich, den sogenannten »Drachen«, der aus 144 Steinen besteht. Es gibt insgesamt 42 verschiedene Sorten von Spielsteinen. Der Spieler muß nun zwei gleiche Steine finden, die »frei« sind. Frei bedeutet in diesem Zusammenhang, daß der Stein von oben frei sichtbar ist und sich nach links oder rechts wegschieben läßt. Hat man ein Paar gleicher Steine gefunden, können beide weggenommen werden. Dies geschieht einfach dadurch, daß man erst einen Stein mit der Maus anklickt. Er erscheint dann angeleuchtet auf dem Bildschirm. Danach braucht der zweite passende Stein nur mit einem Doppelklick bedacht zu werden und schon sind beide Steine vom Bildschirm verschwunden. Ziel des Spieles ist es, am Ende möglichst wenig Spielsteine übrig zu haben. Wer es sogar schafft die Pyramide so aufzulösen, daß am Ende gar keine Steine mehr übrig sind, für den gibt es einen grafischen Leckerbissen. Ein Drache durchbricht die nun leere Spielfläche und schaut einen mit bösen Augen an. Dieses Denkspiel kann allein oder mit mehreren Spielern gespielt werden. Besonderen Spaß bereitet es, mit zwei Mäusen gegeneinander unter Zeitlimit zu spielen. Die Regeln sind leicht zu begreifen. Um aber alle Steine zu entfernen, brauchen auch schnelle Denker viel Erfahrung. Durch den zufälligen Aufbau der Pyramide wird das Spiel nie langweilig. Übrigens läßt sich jede Pyramide auflösen. Der Computer bedenkt, daß sich nicht alle Steine einer Sorte übereinander befinden dürfen.

In zahlreichen Spielen konnten wir folgende Taktiken erfolgreich einsetzen:

Versuchen Sie, die äußeren und obersten Steine so schnell wie möglich abzubauen. Sind die Außenpositionen entfernt, stehen mehr Möglichkeiten zur Wahl. Sobald die beiden obersten Lagen entfernt sind, sollten Sie im Zweifelsfall besser die Außensteine entfernen. Die unteren Reihen im Zentrum der Pyramide sollten erst ganz am Ende aufgedeckt sein.

Bei zwei Spielern im Wettbewerbsmodus sollten Sie anfangs genügend Zeit wählen. Die Eile bei 10 Sekunden führt, gerade bei ungeübten Spielern, zu vielen Reststeinen.

Wir wünschen Ihnen bei allen Varianten dieses ungewöhnlichen Spiels viel Freude.

Compiler contra Interpreter

Wenn es um Programmiersprachen geht, fallen häufig die Begriffe »Compiler« und »Interpreter«. Wir zeigen, was sich dahinter verbirgt. Zusätzlich stellen wir Ihnen den bislang einzigen Compiler für Amiga-Basic vor. Mit diesem werden Basic-Programme erheblich schneller.

Ist es Ihnen auch so gegangen? Sie haben sich den Amiga gekauft und dann eifrig mit dem dazugehörigen Amiga-Basic gearbeitet. Kaum sind die ersten Experimente zur Zufriedenheit beendet, entstehen immer umfangreichere Programme. Diese sollten nach Möglichkeit auch mit schneller Grafikausgabe versehen sein. Außerdem müssen alle Berechnungen in atemberaubender Geschwindigkeit ausgeführt werden. Die Wunschliste läßt sich selbstverständlich beliebig verlängern.

Grenzen des Amiga-Basic

Dabei werden jedoch schnell die Grenzen des Amiga-Basic erreicht. Begeistert anfangs noch die Vielfalt der angebotenen Funktionen, stellt sich ob der nicht gerade rasanten Ausführungsgeschwindigkeit der Programme bald eine gewisse Ernüchterung ein.

»Profis« und »Fachhändler« antworten auf die Frage, wie Basic »Beine gemacht« werden könnten, entweder gar nicht oder mit einem Verweis auf einen »Compiler«.

Mehr oder weniger überwältigt von der erschöpfenden Auskunft des Gesprächspartners darf sich der Computer-Freund nun auf die Suche nach einem »Compiler« machen. Fündig wird er dabei jedoch nur, wenn er weiß, was ein Compiler eigentlich ist und wozu er dient.

Dieser Artikel beantwortet die gestellten Fragen und zeigt Ihnen außerdem, warum Basic-Programme »langsam«, mit einem Compiler übersetzte Programme dagegen so »schnell« sind.

Dazu wollen wir uns zunächst mit Ihrem täglichen Arbeitsmittel, dem von Microsoft entwickelten Amiga-Basic, etwas genauer befassen. Die Programmiersprache Basic hat sich nicht nur aufgrund ihrer leichten Erlernbarkeit und Vielseitigkeit im Heimbereich durchgesetzt. An diesem Erfolg hat mit großer Wahrscheinlichkeit auch die Tatsache großen Anteil, daß die Programmierung in Basic (beziehungsweise die Bedienung eines Basic-Systems wie Amiga-Basic) verhältnismäßig einfach ist.

Auch kann der Basic-Programmierer sehr leicht Änderungen am Programm vornehmen. Die Eingabe des Startkommandos »RUN« genügt, und schon können die Ergebnisse einer Änderung auf dem Bildschirm bewundert werden.

So ist ein Arbeiten nach dem »Try and Error«-Verfahren (engl. für »Versuch und Irrtum«), das heißt eine interaktive Art der Programmierung möglich.

Dies ist nur ein Teil der Eigenschaften, die einen »Interpreter« kennzeichnen.

Was ist ein Interpreter?

Ein Interpreter ist ein Programm, das den Quelltext (ein Basic-Programm) Schritt für Schritt, Befehl für Befehl auswertet. Der Quelltext wird in für den Computer verständliche Befehlsfolgen umgesetzt. Die Umsetzung ist notwendig, weil der Computer Befehle wie »PRINT« eigentlich gar nicht kennt. Hinter dem Basic-Befehl »PRINT« verstecken sich eine ganze Reihe von Anweisungen, die ausgeführt werden müssen, um etwas auf den Bildschirm auszugeben. Damit Sie sich nicht die zahlreichen Einzelanweisungen merken müssen, die zur Bildschirmausgabe notwendig sind, wird Ihnen von Basic das leicht zu merkende Wort »PRINT« zur Verfügung gestellt.

Betrachten Sie beispielsweise folgendes kurze Programm:

```
10 INPUT "Ihr
   Name bitte :",name$
20 PRINT "Hallo,
   liebe(r) ";name$
```

Starten Sie dieses Programm mit RUN, so wird zunächst eine Zeichenkette eingelesen und dann mit dem Vorsatz »Hallo, liebe(r) « wieder ausgegeben. Das Wort RUN startet den Interpreter. Dieser holt sich den ersten ihm bekannten Befehl. In unserem Beispiel ist das INPUT. Er sucht dann anhand einer Befehlstabelle, ob dieses Wort überhaupt Bestandteil der Sprache Basic ist. Dann führt er eine entsprechende Folge von Maschinenbefehlen aus. Im Beispiel würde eine Befehlsfolge erzeugt, die eine Zeichenkette ausgibt (»Ihr Name bitte : «) und dann eine Zeichenkette von der Tastatur einliest. Der nächste Befehl ist dann PRINT, das wie oben bereits angesprochen Maschinenbefehle erzeugt. Erst diese Maschinenbefehle geben die Zeichenketten auf dem Bildschirm aus.

Anhand dieses kurzen Beispieles wird das Prinzip eines Interpreters bereits deutlich (sehen Sie dazu auch Bild 1). Er sucht nach dem jeweils nächsten Befehl und vergleicht ihn mit einer Tabelle, in der alle ihm bekannten Funktionen enthalten sind. Kann der Befehl darin gefunden werden, führt der Interpreter den zu diesem Befehlstabelleneintrag gehörenden Maschinencode aus.

Dies wiederholt sich für jeden einzelnen Befehl des Basic-Programmes. Daraus resultiert auch die langsame Ausführungsgeschwindigkeit solcher Programme.

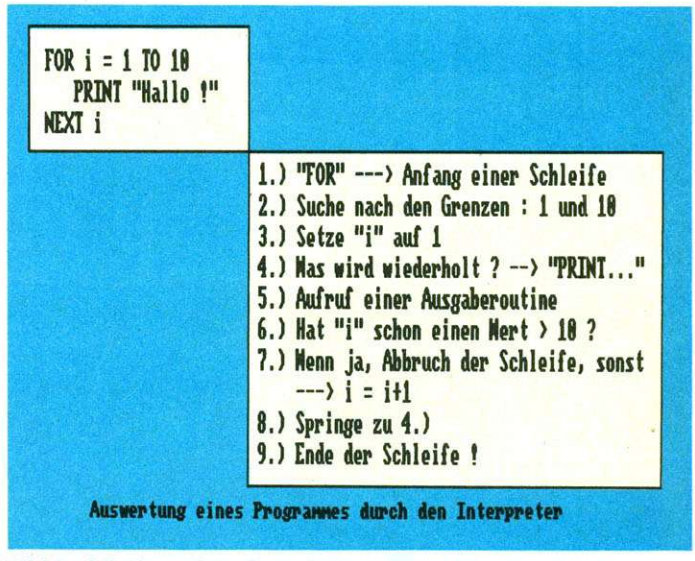


Bild 1. Arbeitsweise eines Interpreters: Er sucht nach dem jeweils nächsten Befehl und vergleicht ihn mit einer Tabelle, die alle ihm bekannten Funktionen enthält.

Da der Interpreter nicht weiß, ob sich eine Addition wie

```
d = i+j
```

auf reelle Zahlen bezieht, oder ob hier Integerzahlen (siehe INT, Basic-Handbuch) addiert werden sollen, muß er vor der Addition eine Prüfung vornehmen. Erst wenn er weiß, um welche Datentypen es sich handelt, kann er den richtigen Code zur Addition ausführen. Auch dies verlangsamt die Ausführung des Programmes.

Doch das Interpreter-Konzept bringt nicht nur den Nachteil der langsamen Arbeitsgeschwindigkeit mit sich, sondern — wie oben bereits angeführt — auch den Vorteil der leichteren Bedienbarkeit.

Dazu gehört auch die einfache Möglichkeit, in den Ablauf eines Programmes einzugreifen. Ein Interpreter (so auch Amiga-Basic) bietet in der Regel die Option, ein gerade laufendes Programm an jeder beliebigen Stelle zu unterbrechen. Der Programmierer kann sich dann den Inhalt verschiedener Variablen ansehen, die-

Einfache Fehlerbehandlung

se verändern und das Programm dann mit den neuen Werten weiterlaufen lassen. So kann auf einfache Art und Weise die Reaktion des Programmes auf bestimmte Werte überprüft werden.

Dadurch ist es relativ leicht, an Programmen, die nicht die gewünschten Ergebnisse liefern, Fehler zu erkennen und zu beheben.

Basic-Programme werden im allgemeinen mit einem speziellen Editor (Texterfassungsprogramm) eingegeben. Der Vorteil eines Interpreters wie zum Beispiel Amiga-Basic liegt nun darin, daß der Editor

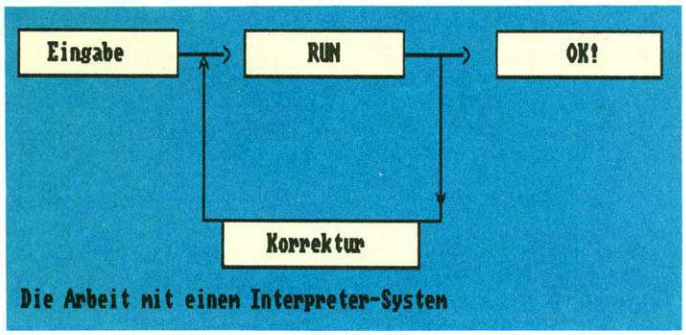


Bild 2. Sie müssen nicht mit verschiedenen Programmen arbeiten, um ein Basic-Programm zu erstellen. Sie brauchen die Programmierung nur einmal zu lernen. Hier sehen Sie alle nötigen Schritte.

ein Bestandteil des eigentlichen Basic-Systems ist. So genügt der Wechsel in das »LIST-Fenster«, um eine kleine Veränderung am Programm vorzunehmen. Sie müssen nicht mit verschiedenen Programmen arbeiten, um ein Basic-Programm zu erstellen. Sie brauchen die nötigen Schritte zur Programmierung nur einmal zu lernen. Bild 2 stellt diesen Zusammenhang dar.

Bei der Verwendung eines Compilers muß der Programmierer für den Vorteil der wachsenden Ausführungsgeschwindigkeit der Programme den Preis einer weniger einfachen Bedienung bezahlen.

Das bedeutet konkret, daß Sie erst den Quelltext des Programmes mittels eines Editors eingeben müssen. Nach der Eingabe ist dieser Quelltext im »ASCII«-Format abzuspeichern. ASCII ist die Abkürzung für ein Datenformat. Wird im ASCII-Format abgespeichert, so wird der reine Text gesichert. Etwaige Steuerzeichen (wie Sie beispielsweise von Textverarbeitungsprogrammen in den Text geschrieben werden) werden aus dem Text entfernt. Der Compiler erwartet die Daten genau in diesem ASCII-Format. Sollten Sie nicht im Besitz eines eigenen Programmeditors sein, so können Sie die Programme auch weiterhin im »LIST-Fenster« von Amiga-Basic eingeben. Abspeichern müssen Sie das Programm allerdings dann mit der »a«-Option (für ASCII, siehe Handbuch Amiga-Basic). Ist die Eingabe des Programmes beendet und der Quelltext erfolgreich abgespeichert, so wird der Editor verlassen. Jetzt ist der »Compiler« an der Reihe. Er liest den Quelltext ein und erzeugt daraus ein direkt ablauffähiges Maschinenprogramm. Dabei sucht er (meist gründlicher als ein Interpreter) nach syntaktischen Fehlern des Programmes. Sollten solche gefunden werden, weist der Compiler Sie darauf hin. Sie müssen dann den Compiler verlassen, wieder den Editor aufrufen und die Fehler beseitigen. Der korrigierte Quelltext wird dann abermals an den Compiler übergeben. Bild 3 zeigt diese Arbeitsschritte.

Schneller Compiler

Sind alle Fehler beseitigt, kann das Programm gestartet werden. Dazu wird der Compiler verlassen und das Programm (über CLI oder Workbench) aufgerufen.

Anhand dieser Beschreibung erkennen Sie bereits die etwas umständlichere Handhabung eines Compilers.

Doch nun zu den Grundlagen zurück. Ein »Compiler« ist nichts anderes als ein Programm zur Umsetzung eines Quelltextes in den Maschinencode eines speziellen Computers. Sie werden jetzt vielleicht sagen: Genau das macht ein Interpreter doch auch, oder?

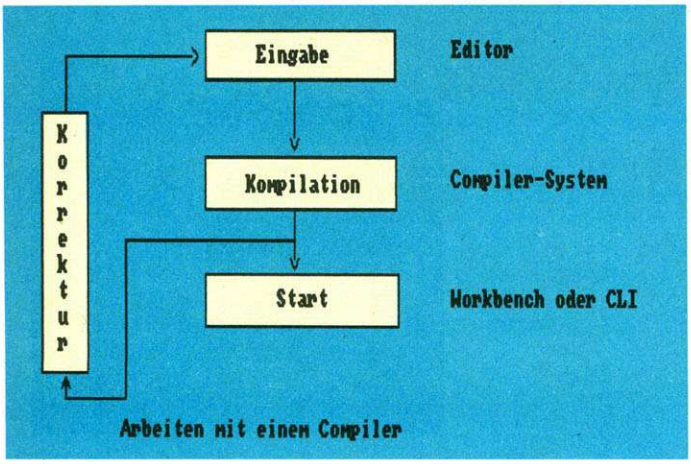


Bild 3. Der korrigierte Quelltext wird erneut an den Compiler übergeben

Richtig, nur geht der Compiler dabei etwas »geschickter«, sprich effektiver vor.

Im Gegensatz zum Interpreter (der das Programm während seiner Ausführung ständig »bearbeitet«) setzt der Compiler das Programm vor (!) seiner ersten Ausführung in den Maschinencode des Computers um. Das bedeutet aber auch, daß das Ergebnis der Compilation ein Programm ist, das völlig eigenständig läuft, ohne den »Klotz« Interpreter ständig am »Bein« zu haben.

Die einzelnen Programmzeilen werden nicht mehr der Reihe nach immer wieder aufs neue in Maschinencode übertragen, sondern nur noch einmal, nachdem das Programm (besser: der Quelltext des Programmes) fertiggestellt wurde. Dies bringt in vielen Fällen einen deutlichen Geschwindigkeitszuwachs mit sich.

Optimierende Compiler

Ein weiterer Vorteil der Compiler-Technik ist, daß der Compiler bei der Übersetzung des Programmes erkennen kann, ob bestimmte Berechnungen oder Anweisungen überhaupt benötigt werden. Sind überflüssige Anweisungen vorhanden, besitzen einige Compiler die Fähigkeit, diese aus dem erzeugten Programm zu entfernen. Diese Fähigkeit des Compilers wird »Optimierung« genannt. Bei einem Compiler, der diese Fähigkeit besitzt, wird von einem »optimierenden« Compiler gesprochen.

Optimierungen tragen sowohl zu einer weiteren Geschwindigkeitssteigerung als auch zu einer geringeren Größe des erzeugten Programmes bei. Dabei ist zu bemerken, daß die von einem Compiler erzeugten Programme (auf der Diskette und im Speicher) meist deutlich länger sind, als die entsprechenden (nicht-compilierten) Basic-Programme. Dies liegt daran, daß die nicht-compilierten Programme als reiner Text abgespeichert werden. Compilierte Programme dagegen enthalten eine Vielzahl von zur Ausführung derselben benötigten Programmteile, die im Quelltext selbstverständlich nicht erscheinen. Sie werden vom Compiler selbständig zum fertigen Programm hinzugefügt.

In diesem Zusammenhang fällt auch häufig der Begriff »Laufzeitsystem« (LZS) oder »Runtime-System«. Darunter wird eine Sammlung von Unterprogrammen verstanden, die zu jedem compilierten Programm mit gespeichert werden. Diese Unterprogramme unterstützen die Ausführung des Programms beziehungsweise ermöglichen diese sogar erst. Ein Teil dieses Laufzeitsystems könnten zum Beispiel Routinen zur Verarbeitung reeller Zahlen sein. Eine weitere Aufgabe des LZS ist meist die Erkennung und Meldung von Laufzeitfehlern. Dies bedeutet, daß bei einer Division durch Null der Computer sich nicht sofort zu einer Meditation mit dem Guru über dieses schwierige mathematische Problem zurückzieht, sondern eine Fehlermeldung der Art »Fehler : Division durch Null« ausgegeben wird.

Durch die vielen Funktionen des LZS, die an jedes Programm angehängt werden, erscheinen kleine Programme schnell »aufgebläht«. Die Länge der erzeugten Programme steht in keinem Verhältnis mehr zum Quelltext. Ein guter Compiler erkennt, welche Routinen des LZS benötigt werden und fügt nur diese zum erzeugten Programm hinzu.

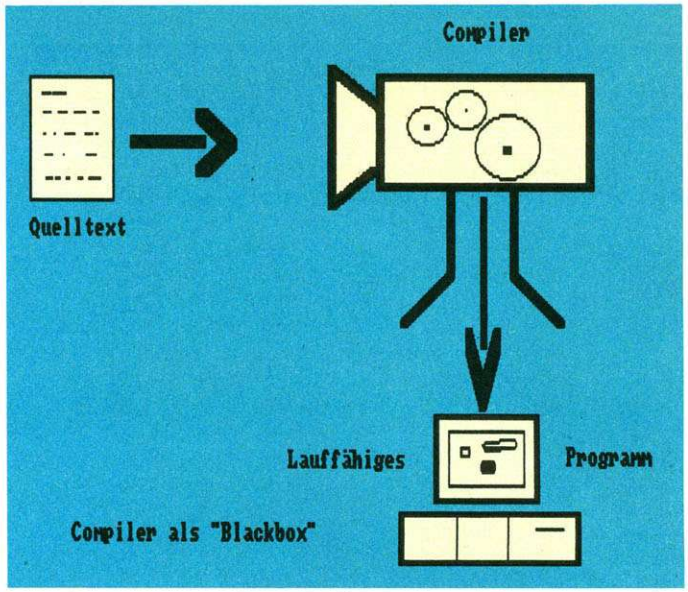


Bild 4. Der Compiler wird quasi zur »Blackbox«. Das Ergebnis (fertiges Programm) wird auf eine dem Anwender nicht bekannte Art und Weise erzielt.

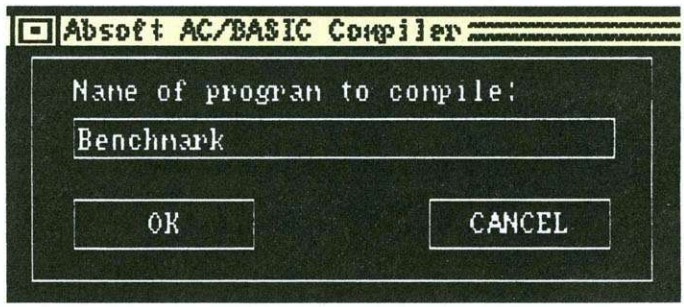


Bild 5. Das »Datei-Fenster« des AC-Basic. Hier geben Sie an, welches Programm Sie compilieren möchten.

Da wir gerade von den Fehlermeldungen sprachen: Wie weiter oben schon angedeutet, liest der Compiler den Quelltext eines Programms vor der Übersetzung ein. Danach wird dieser analysiert und in den Maschinencode übertragen. Bei der Analyse gehen Compiler häufig detaillierter auf etwaige Fehler im Quellprogramm ein, das heißt sie erzeugen oft mehr und/oder genauere Fehlermeldungen. So wissen Sie als Programmierer besser, woran es in der entwickelten Software noch »krankt«.

Genaue Fehlermeldungen

Doch nicht jedes Programm eignet sich zur Compilation. Dies trifft besonders auf Programme zu, die nicht von ihrer Geschwindigkeit leben (wie etwa Spiele), sondern sogar auf ein etwas gemächlicheres Tempo angewiesen sind. Als Beispiel lassen sich hier Programme anführen, die Musikstücke ausgeben. Werden diese Programme compiliert, so entsprechen die Ergebnisse meist kaum den Erwartungen. Durch die zusätzliche Geschwindigkeit werden die Musikstücke bis in die Unkenntlichkeit verstümmelt. Langwieriges Herumprobieren, mit welcher Geschwindigkeit (und Verzögerung) das Stück auch in der compilierten Version annehmbar klingt, ist die Folge. Gerade hierbei macht sich die Trennung von Editor, Compiler und ausführbarem Programm besonders negativ bemerkbar.

Der Compiler wird quasi zur »Blackbox« (Bild 4). Das Ergebnis (fertiges Programm) wird auf eine Ihnen nicht bekannte Art und Weise erzielt. Die Möglichkeiten zur Verifikation des erzeugten Codes sind gering und bleiben Assembler-Kennern vorbehalten.

Sie wissen nun, was ein Compiler macht. Er übersetzt den Basic-Quelltext in ein ablauffähiges Maschinencode-Programm.

Bevor Sie im nächsten Abschnitt einiges über den einzigen derzeit für den Amiga erhältlichen Basic-Compiler erfahren, wollen wir Sie noch auf einige wichtige Punkte auf-

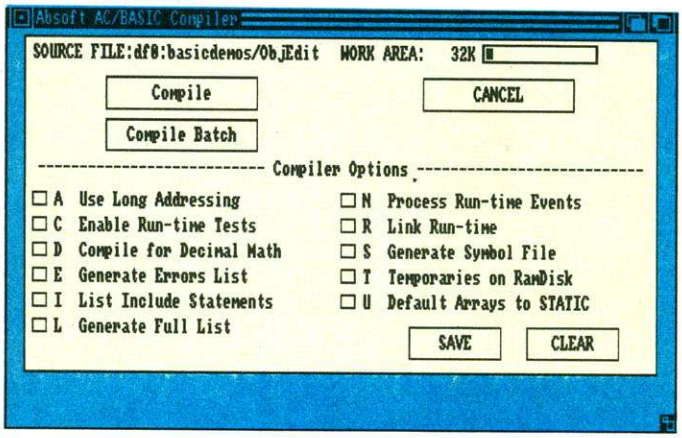


Bild 6. Das »Control Panel«, über das Sie Zugriff auf die Parameter bei der Compilation haben und den Compiler starten können

```
REM Bench-Test
PRINT "Primzahlsieb des Erathostenes"
PRINT "============================":PRINT
INPUT"Obergrenze der Primzahlenberechnung: ";pmax
d%=(pmax-1)/2:s%=(SQR(pmax)-1)/2
CLS:PRINT "Alle Primzahlen bis ";pmax":"
DIM pz%(d%):PRINT :PRINT  2;:a%=1:t1=TIMER
FOR i%=1 TO s%: IF pz%(i%)=0 THEN PRINT 2*i%+1;:a%=a%+1:IF a% MOD 10
=0 THEN PRINT
FOR j%=2*i%*(i%+1) TO d% STEP 2*i%+1:pz%(j%)=1:NEXT :NEXT :t2=TIMER
FOR i%=s%+1 TO d%:IF pz%(i%)=0 THEN PRINT 2*i%+1;:a%=a%+1:IF a% MOD
10=0 THEN PRINT
NEXT:t3=TIMER
PRINT :PRINT  "anzahl :";a%
PRINT "Rechenzeit :";t2-t1;"Sekunden"
PRINT "Zeit insgesamt";t3-t1;"Sekunden"
FOR t=1 TO 10000:NEXT t
```

Listing 1. Ein kurzes Basic-Programm, das wir für einen Zeitvergleich für Sie durch den Compiler »geschickt« haben. Das Programm berechnet Primzahlen.

Versuch	Amiga-Basic		AC/Basic	
	Berechnung	Ausgabe	Berechnung	Ausgabe
1	28,07	151,22	7,27	68,33
2	28,16	151,36	7,30	68,30
3	28,07	151,26	7,30	68,32
4	28,08	150,86	7,28	68,38
Mittel:	28,09	151,17	7,29	68,33

Tabelle 1. Die Zeiten für die Berechnung und Ausgabe aller Primzahlen bis zur Zahl 30000

merksam machen, die Sie beim Kauf eines Compilers beachten sollten.

Worauf Sie auf jeden Fall beim Kauf eines Compilers achten sollten :

— erkennt der Compiler alle Programme, die mit Ihrer Basic-Version erzeugt wurden?

— besitzt der Compiler Fehler? Wenn ja, sind diese dokumentiert (das heißt zum Beispiel im Handbuch aufgelistet)?

— wie schnell laufen die compilierten Programme ab?

— wie lange benötigt der Compiler zur Erzeugung eines ablauffähigen Programmes?

In diesem Abschnitt stellen wir Ihnen den einzigen zur Zeit der Erstellung dieses Artikels erhältlichen Basic-Compiler kurz vor.

Der Compiler für Amiga-Basic

Sein Name ist »AC/Basic« (ACB, Version 1.2). Er wird auf einer Amiga-Diskette zusammen mit einem recht umfangreichen Handbuch geliefert.

ACB ist ein Compiler für den Basic-Dialekt »Amiga-Basic«. Sie können also Programme übersetzen lassen, die mit dem zum Amiga ausgelieferten Basic-Interpreter erstellt wurden.

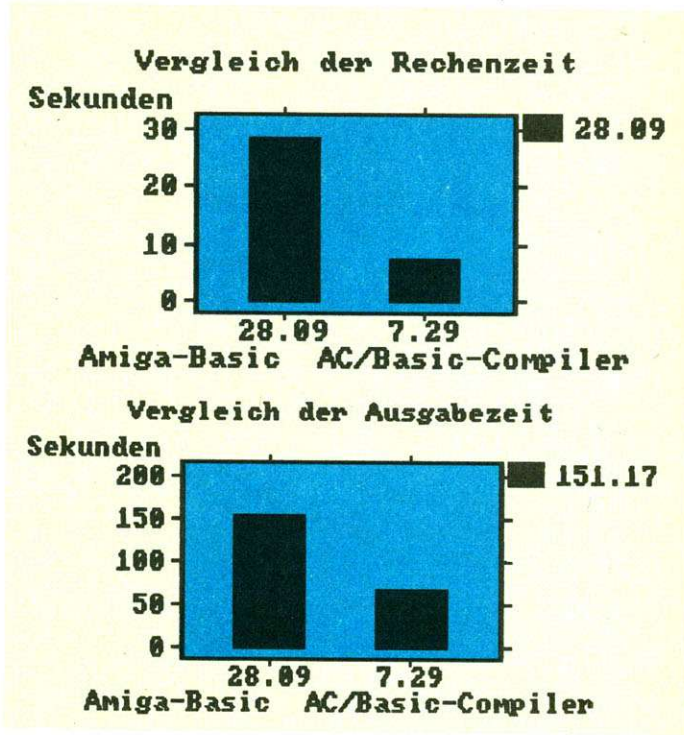


Bild 7. Die Vergleichswerte für die Ausgabe der Primzahlen bis zur Grenze 30000 als Balkengrafik

Sie müssen sich keinen neuen Interpreter zulegen, sondern können wahrscheinlich sogar alle Ihre Programme von diesem Compiler übersetzen lassen, ohne große Veränderungen daran vorzunehmen.

Das (englische) Handbuch erklärt auch die Unterschiede zwischen Amiga-Basic und AC/Basic.

Darunter fallen zum Beispiel Befehle wie CONT, DELETE, LIST, LLIST, LOAD, MERGE, NEW, SAVE, TRON, TROFF, die der Übersetzer überhaupt nicht kennt. Dies ist auch leicht einsichtig, da zum Beispiel über DELETE nur Zeilen des Quelltextes löschbar sind, nicht jedoch im ablauffähigen Programm.

In Bild 5 sehen Sie das »Datei-Fenster« des ACB. Hier geben Sie an, welches Programm Sie compilieren möchten. Danach erscheint ein Fenster (»Control Panel«, Bild 6), über das Sie Zugriff auf einige Parameter bei der Compilation haben und den Compiler starten können. Sie können über die einzelnen Schalter (Gadgets) zum Beispiel wählen, welche Adressierungsart für den erzeugten Code gewählt werden (lang/kurz) und wie der Compiler mit Feldern (Arrays) umgehen soll.

Der wichtigste Schalter in diesem Fenster heißt allerdings »Compile« — er startet den Compilationsvorgang. Sollte der Compiler einen Fehler entdecken, macht er Sie darauf aufmerksam. Unterbleibt eine derartige Fehlermeldung, können Sie den Compiler verlassen und das compilierte Programm über das zugehörige Bildsymbol (Icon), oder im CLI über den Programmnamen aufrufen.

Dabei ist zu beachten, daß der Compiler an den Namen des Quelltextes die Endung ».run« hängt. Das heißt, das compilierte Programm endet immer mit ».run«. Dadurch kann auf einfache Art und Weise zwischen Quell- und Objektdatei unterschieden werden.

In Listing 1 finden Sie ein kleines Basic-Programm, das wir für einen kleinen Zeitvergleich für Sie durch den Compiler »geschickt« haben. In Tabelle 1 sehen Sie die Ergebnisse.

Basic im Rausch der Geschwindigkeit

In diesem Artikel haben wir Ihnen alle Grundlagen zum Thema »Compiler und Interpreter« vermittelt, die Sie unbedingt vor einem Kauf berücksichtigen sollten.

Ob ein Compiler für Sie die richtige Ergänzung Ihres »Programmieralltages« ist, müssen Sie jedoch selbst entscheiden. Ziehen Sie dabei auf jeden Fall sowohl den erzielbaren Geschwindigkeitsgewinn als auch den mit der Verwendung eines Compilers verbundenen Verlust an Bedienungskomfort in Betracht. *(Ingolf Krüger/rs)*

BASIC

F-BASIC

GFA-BASIC

Für den Amiga gibt es einige Versionen der bekanntesten aller Programmiersprachen: Basic. Wir vergleichen die zur Zeit führenden vier Basic-Dialekte für den Amiga und sagen, welcher für verschiedene Ansprüche geeignet ist.

Zwar ist der Amiga durch sein in der Sprache »C« geschriebenes Betriebssystem eigentlich für diese Sprache prädestiniert. Doch vor allem Anfänger tun sich oft recht schwer mit dieser Programmiersprache. Wesentlich angenehmer ist es, Basic zu lernen, mit dem viele Umsteiger auf den Amiga schon vertraut sind. Doch wer sich gegen C oder auch Assembler entscheidet, steht vor der Qual der Wahl: Welcher Dialekt ist für ihn am besten geeignet? Ist es das mitgelieferte Amiga-Basic, True Basic, GFA-Basic oder F-Basic?

SIC EKTE

AMIGA-BASIC

TRUE-BASIC

Natürlich kann keiner der Dialekte als »der beste« bezeichnet werden, da jeder für sich seine ganz speziellen Besonderheiten und Sonderfunktionen hat. Zudem stellt jeder Programmierer andere Ansprüche, daher beschränkt sich unser Vergleich auf Hinweise, welcher Basic-Dialekt für verschiedene Anforderungen geeignet ist.

Der Klassiker: Amiga-Basic

Der größte Vorteil des **Amiga-Basic** ist wohl sein Preis, denn es wird jedem Amiga kostenlos beigelegt. Somit ist es seinen Mitstreitern schon einmal um ungefähr 200 Mark voraus — und das ist ja schließlich nicht gerade wenig.

Das von der Firma Microsoft entwickelte Amiga-Basic ist ein reiner Interpreter, das heißt, das eingegebene Programm wird Befehl für Befehl in Maschinensprache übersetzt (interpretiert) und dann ausgeführt. Solche Interpreter haben sowohl Vor- als auch Nachteile. Zum Beispiel startet das Programm unmittelbar nach der Eingabe von RUN (beim Compiler muß es erst komplett übersetzt werden). Natürlich weiß der Interpreter zu jedem Zeitpunkt der Ausführung, an welcher Stelle im Quelltext sich das Programm gerade befindet. Einerseits kann so bei einem fehlerbedingten Pro-

grammabbruch die unkorrekte Befehlszeile sofort ausgemacht werden, andererseits bieten die meisten Interpreter (so auch Amiga-Basic) eine Trace-Option, bei der während des Programmablaufs die gerade abgearbeitete Befehlszeile angezeigt wird.

Der gemütliche Interpreter

Sie können Amiga-Basic von der Workbench durch Anklicken des zugehörigen Icons starten. Der Interpreter startet aber auch, wenn Sie das Icon einer Amiga-Basic-Datei anklicken oder wenn Sie direkt vom CLI aus den Befehl »run AmigaBasic« eingeben. Nun meldet sich das Programm mit dem Hauptfenster für direkte Befehlseingabe und Programmausgaben. Rechts daneben erscheint das Fenster des eingebauten Editors. Dies ist die Arbeitsoberfläche von Amiga-Basic, die eine der größten Schwächen des Programms darstellt.

Zwar wird uneingeschränkte Amiga-Fenstertechnik geboten, Sie können also alle Arbeitsfenster beliebig verschieben und in der Größe verändern; allerdings geschieht dies mit quälender Langsamkeit. Will man ein mit Text gefülltes List-Fenster von ungefähr einem Viertel der Bildschirmgröße verschieben, sind Wartezeiten bis zu fünf Sekunden keine Seltenheit.

Wurde in dem gerade laufenden Basic-Programm ein Fehler gefunden, geschieht folgendes:

Der Interpreter stoppt das Programm, schaltet das List-Fenster in den Vordergrund, rahmt die fehlerhafte Zeile im Quelltext ein und gibt in einem zusätzlichen kleinen Fenster die Art des Fehlers aus. Dieses Fenster muß der Anwender dann anklicken, um im Quelltext den Fehler verbessern zu können. Anschließend ist der Mauszeiger noch in die Nähe des umrahmten Fehlers zu bewegen und erneut ein Mausklick auszuführen. Bis es dann endlich so weit ist, daß man den geringfügigen Tippfehler mit wenigen Tastendrücken verbessern kann, vergeht also viel Zeit, so daß Geduld und Nerven des Programmierers bei intensiver Arbeit sehr strapaziert werden.

Im Editor selber läßt die Verarbeitungsgeschwindigkeit ebenfalls stark zu wünschen übrig. Wer das Listing zeilenweise durch das Fenster scrollen lassen will, läßt meist den Finger so lange auf der Cursortaste, bis die gesuchte Stelle erreicht ist. Allerdings ist die Tasten-Wiederholung nicht der Scroll-Geschwindigkeit angepaßt, so daß sich nach dem Loslassen der Taste immer noch zahlreiche Tastendrucke im Tastaturpuffer befinden und das Listing lustig weiterscrollt. Passiert dies gerade gegen Ende des Quelltextes, so wird man nach der letzten Zeile für jeden einzelnen übrigen Tastendruck im Puffer mit einem Bildschirmblitz und Piepton bestraft.

Der Editor von Amiga-Basic

Alle Befehle an die Benutzer oberfläche von Amiga-Basic sind über Menüs, aber nur die Hälfte dieser 16 Menüpunkte über Tastatur erreichbar. Leider besitzen gerade oft benötigte Funktionen, wie OPEN oder SAVE, keinen Tastaturcode. Übrigens lassen auch gerade diese beiden Befehle stark zu wünschen übrig. Ein Fileselector, in dem lediglich einer der angezeigten Dateinamen angeklickt wird, fehlt.

Wird man von seinem Gedächtnis einmal im Stich gelassen, so hilft allerdings der Befehl FILES im Command-Fenster; er zeigt eine Übersicht aller Einträge im aktuellen Verzeichnis.

Im Gegensatz zur durchaus verbesserungswürdigen Oberfläche von Amiga-Basic ist Sprachqualität und Sprachumfang für Standard-Anwendungen ausreichend. Amiga-Basic ist eine strukturierte Sprache mit Pascal-Elementen (Schleifen, Entscheidungen, Funktionen und Unterprogrammen mit lokalen Variablen), so daß eine durchgehende Zeilennumerierung zwar noch möglich, aber in keiner Weise nötig ist. Die Zeilennummern, soweit vorhanden, werden einfach als Marken (Label) verwendet, die mit GOTO oder GOSUB anzuspringen sind. Erfreulicherweise unterstützt der Editor die strukturierte Programmierung, indem nach der Eingabe einer eingerückten Zeile der Cursor in der nächsten Zeile automatisch mit der vorigen Einrückung erscheint.

Amiga-Basic unterscheidet auch genau unter den numerischen Variablentypen verschiedener Genauigkeiten (Wörter, Langwörter, Fließkommazahlen einfacher und doppelter Genauigkeit) und bietet entsprechend auch viele Funktionen zur Umwandlung von Werten zwischen den verschiedenen Typen.

Von den Befehlen zur Grafik-Programmierung kann man wohl guten Gewissens sagen, daß sie die Fähigkeiten des Amiga vollständig abdecken. Neben den Standard-Befehlen zum Zeichnen von Punkten, Linien, Rechtecken, Kreisen und Polygonen bekommt die Steuerung der Sprites und Bobs (auch unter dem Namen Shapes bekannt) neue Dimensionen. Zwar ist der Editor für diese grafischen Objekte nicht im Interpreter integriert, sondern selber ein Basic-Programm, dafür kann man nach Herzenslust Priorität, Farbe, Ort, Geschwindigkeit und sogar Beschleunigung bestimmen. Auch Kollisionen zwischen den Objekten werden komfortabel verarbeitet.

Natürlich stehen auch die Befehle WINDOW und SCREEN zum Öffnen von Amiga-Bildschirmen und Fenstern zur Verfügung. Nur leider sind weder Borderless-Fenster (Fenster ohne Rahmen) noch Veränderungen an der Fensterposition und -größe nach dem Öffnen möglich.

Fensterinhalte speichert Amiga-Basic auf Wunsch im ACBM-Format ab. Allerdings befinden sich auf der Diskette zwei Dienstprogramme, die ILBM- in ACBM-Bilder konvertieren (und umgekehrt).

Um die Soundfähigkeiten des Amiga auszunützen, können Sprachausgabe und Töne mit beliebiger Wellenform programmiert werden. Das direkte Abspielen von digitalisierten Sounds aus IFF-Dateien ist übrigens bei keinem der vier Dialekte ohne zusätzlichen Programmieraufwand in Ma-

Die Benchmark-Tests

Um die Geschwindigkeit der vier Basic-Dialekte möglichst objektiv zu vergleichen, werden natürlich Maßstäbe in Form von Ausführungszeiten spezieller Programme benötigt. Diese Testprogramme (Benchmarks) sollten bestimmte Fähigkeiten der Sprache möglichst intensiv nützen.

Für den Test wurden fünf verschiedene »Benchmark«-Testprogramme entworfen, welche die gemeinsamen Fähigkeiten der Dialekte umfassend abdecken. Die Programme wurden jeweils an den Testkandidaten angepaßt und, wenn möglich, auch in den jeweiligen Sprachgegebenheiten optimiert. Die Zeiten mißt man beim Amiga sinnvollerweise nur in ganzen Sekunden, da zahlreiche äußere Einflüsse (Mausbewegung, gleichzeitig bearbeitete Tasks und ähnliches) die Ausführungszeit empfindlich beeinflussen.

Der erste Benchmark-Test berechnet auf einfachstem Wege die Primzahlen zwischen 1 und 2000. Im wesentlich besteht dieses Programm aus zwei geschachtelten WHILE-Schleifen sowie einfachen Rechnungen mit Integer-Variablen.

Im Gegensatz zum ersten bedient sich der zweite Benchmark-Test ausgiebig der Fließkommavariablen und -operationen wie trigonometrischen, Exponential-, Wurzel- und Logarithmus-Funktionen sowie Grundrechenarten. Gerade bei einem Computer wie dem Amiga ist die Geschwindigkeit der Grafikberechnung und Ausgabe in vielen Anwendungen ausschlaggebend.

Der dritte Benchmark-Test prüft die Routinen zum Zeichnen von Einzelpunkten, leeren und ausgefüllten Rechtecken sowie von Ellipsen. Da die Objekte zufällig plaziert werden, spielt auch die Geschwindigkeit der Zufallsfunktion RND eine Rolle.

Die beiden letzten Benchmark-Tests erbringen die Geschwindigkeit für das Schreiben und Lesen in und aus sequentiellen Dateien. Leider stehen Befehle für die Bearbeitung relativer Dateien unter F-Basic und True Basic nicht zur Verfügung, daher wurden nur die einfachen sequentiellen Dateien getestet.

Da jeder Benchmark-Test der jeweiligen Sprache angepaßt wurde, haben wir auf einen Abdruck der Programme verzichtet.

Test	Amiga	True	GFA	F
Benchmark 1	11:05	05:00	04:07	00:34
Benchmark 2	00:49	01:05	00:15	00:04
Benchmark 3	02:40	02:29	00:45	00:51
Benchmark 4	00:44	01:10	00:33	00:29
Benchmark 5	00:35	00:32	00:23	00:21

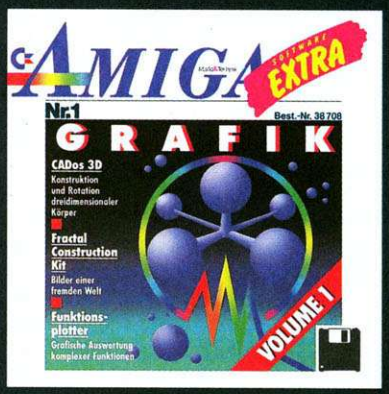
Tabelle 1. Die gemessenen Zeiten für die fünf Benchmark-Tests, die im Kasten oben beschrieben sind.

AMIGA SOFTWARE

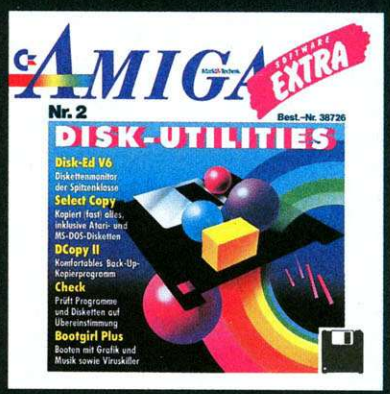
Markt&Technik

SOFTWARE EXTRA

Amiga Extra Nr. 1: Grafik
Drei Programme, die die außergewöhnlichen Fähigkeiten des Amiga 500, 1000 und 2000 nutzen. CADos 3-D: Konstruktion und Rotation dreidimensionaler Körper. Funktionsplotter: Grafische Auswertung komplexer Funktionen. Fractal Construction Kit: Bilder einer fremden Welt.
Bestell-Nr. 38708
DM 49,–* (sFr 44,–*/öS 490,–*)

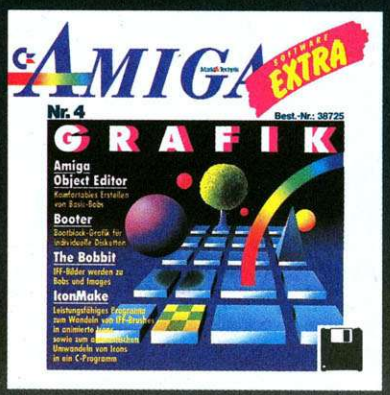


Amiga Extra Nr. 2: Disk Utilities
DiskEd, Select Copy, DCopyII, Check, Bootgirl Plus. Hilfsprogramme, die Ihnen den Umgang mit den Daten auf Ihren Disketten erheblich vereinfachen. Mit einem Super-Diskeditor ist es ein leichtes, versteckte und verlorene Daten aufzuspüren und zu rekonstruieren.
Bestell-Nr. 38726
DM 49,–* (sFr 45,–*/öS 490,–*)



Amiga Extra Nr. 3: Spiele
Bliff: Eine ausgeklügelte Variante des Billards. Quadriga: Ein Spiel für Denker, angelehnt an das berühmte »Vier Gewinnt«. Wikinger I: Ein Strategiespiel, angesiedelt im 10. Jahrhundert. Maximal fünf Spieler taktieren um die Sicherung und die Vergrößerung ihres Heimatlandes.
Bestell-Nr. 38724
DM 49,–* (sFr 44,–*/öS 490,–*)

Amiga Extra Nr. 4: Grafik
Amiga Object Editor: Spielerisch Bobs erzeugen. Animation inbegriffen. The Bobbit: IFF-Bilder in Bobs und Images verwandeln oder mit dem eingebauten Malprogramm erstellen. Iconmake: Generieren von animierten Icons und deren Wandel in C-Programme. Booter: Bootblock-Grafik par excellence.
Bestell-Nr. 38725
DM 49,–* (sFr 44,–*/öS 490,–*)



Amiga Extra Nr. 5: Spiele
Spannende Unterhaltung mit vier Super-Spielen! Breaking out: Actionspiel mit schneller Grafik und tollem Sound. Decoder: Verwandeln Sie Ihren Amiga in eine Morsestation. Megamind: Anregende Unterhaltung für kluge Köpfe. Wikinger II: Spannendes Strategiespiel, angesiedelt im Mittelalter.
Bestell-Nr. 38752
DM 49,–* (sFr 44,–*/öS 490,–*)

* Unverbindliche Preisempfehlung

Markt&Technik-Produkte erhalten Sie im Computerfachgeschäft, in den Fachabteilungen der Warenhäuser, im Versandhandel und in Ihrer Buchhandlung.

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0.
SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656.
ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0;
Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526;
Ueberreuter Media Verlagsges.mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 4815 43-0.

schinensprache oder C zu realisieren.

Außerdem bietet Amiga-Basic an Amiga-spezifischen Optionen noch Maus- und Joystickabfrage, Interrupt-Steue-

Spezialitäten gut genutzt

rung für Objekt-Kollisionen, Mausbetätigung und Menüwahl, sowie die Verwendung von Systembibliotheken.

Amiga-Basic ist erweiterungsfähig durch selbstgeschriebene Assemblerroutinen und die Funktionen der Standard-Libraries. Leider können keine C-Programme eingebunden werden, und der Nutzung von Libraries (die übrigens erst in ein spezielles Format konvertiert werden müssen) sind durch das Fehlen des Datentyps »Struktur« enge Grenzen gesetzt.

Amiga-Basic ist insgesamt der langsamste Dialekt in diesem Vergleich. Daß dies nicht nur in der Konzeption als Interpreter liegen kann, zeigen der oft noch langsamere True Basic-Compiler und der wesentlich schnellere GFA-Basic-Interpreter. Insofern ist Amiga-Basic nur bedingt für professionelle Anwendungen geeignet.

Das Handbuch ist im Vergleich zur Arbeitsgeschwindigkeit hervorragend. In den ersten Kapiteln werden die grundsätzliche Bedienung, die Erstellung von grafischen Objekten, sequentielle und relative Dateien, Datentypen und Operatoren sowie einige Besonderheiten von Amiga-Basic beschrieben. Nach einer kurzen, nach Sprachelementen geordneten Übersicht, sind dann sämtliche Befehle, Anweisungen und Funktionen alphabetisch aufgeführt.

Amiga-Basic hat drei ganz entscheidende Vorteile: Erstens existieren prinzipiell alle nötigen Befehle für die meisten Programmanwendungen, zweitens steht es jedem Amiga-Besitzer kostenlos zur Verfügung, und drittens ist Amiga-Basic damit natürlich eine Art Standard. Zwar liegt Amiga-Basic in bezug auf Bedienungskomfort und Ausführungszeiten fast am Existenzminimum, doch zahlreiche interessante und auch praxisgerechte Listings beweisen immer wieder, daß sich mit dieser Sprache einiges anfangen läßt. Wer also mehr zum Spaß programmiert, für den lohnt sich die mindestens 200 Mark

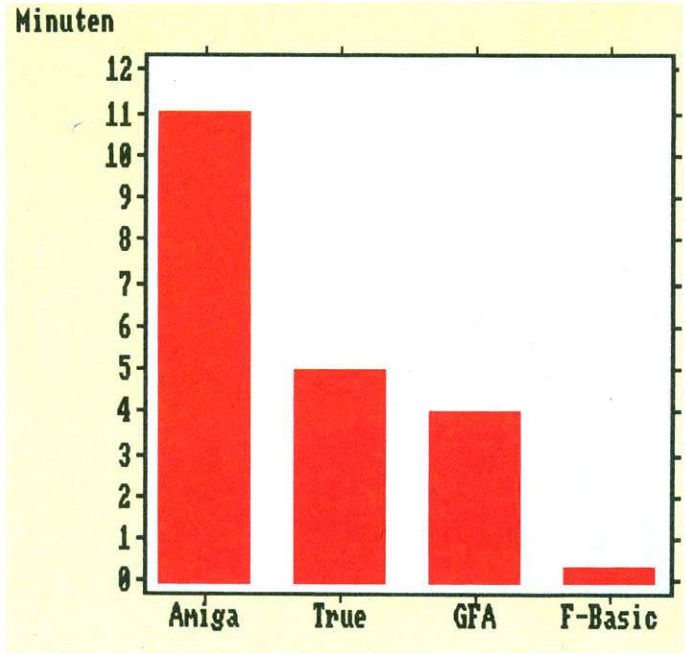


Bild 1. Die vier Kandidaten bei der Berechnung aller Primzahlen von 1 bis 2000

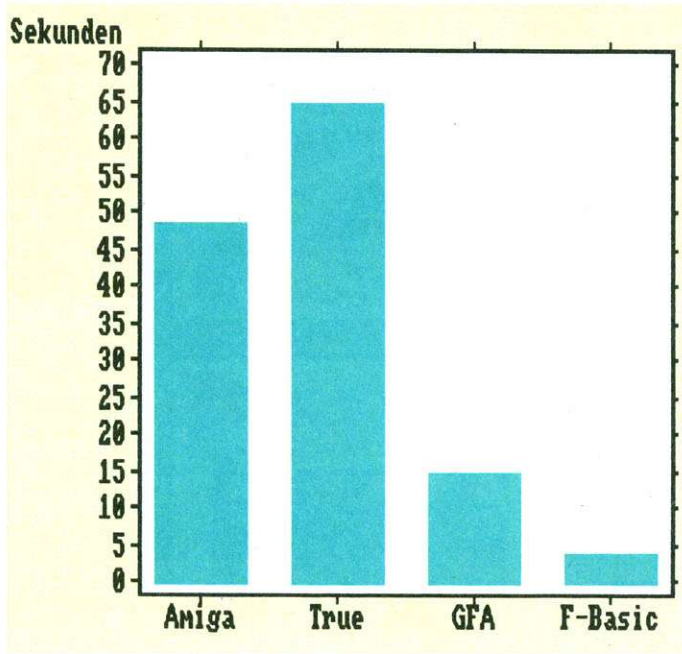


Bild 2. Zweiter Testlauf: Gestoppte Zeiten für Grafikberechnung und Ausgabe

teure Anschaffung eines der Konkurrenzprodukte sicher nicht.

True Basic darf seinen anspruchsvollen Namen (»wahres Basic«) insofern rechtmäßig tragen, da es von John G. Kemeny und Thomas E. Kurtz entwickelt wurde, die vor nunmehr 20 Jahren Basic ins Leben riefen. Daher darf man also von der Qualität dieser Sprache einiges erwarten.

True Basic wurde nicht erst für den Amiga geschrieben, die Versionen für PC und Macintosh haben sich bereits auf dem Markt etabliert. Und hierin liegt auch der entscheidende Vorteil von True Basic: Die bisher existierenden und die vielleicht später noch geschriebenen Umsetzungen des Programms sind zu 100 Prozent kompatibel. Das wurde vor allem dadurch erreicht, daß zum Beispiel numerische Variablentypen automatisch erkannt und entsprechend intern verarbeitet werden, oder daß die Grafikbefehle nur skalierte Koordinaten verwenden.

Wer also ein Programm, daß er auf seinem Amiga geschrieben hat, auch für den PC benutzen will, oder wer bisher auf einem PC unter True Basic programmiert hat, muß nur noch den Quelltext konvertieren, und schon ist das Programm auf dem jeweils anderen Computer lauffähig. Preis für eine so hoch-kompatible Sprache ist allerdings immer, daß die Spezialitäten des Computers nie richtig ausgenützt werden können — doch dazu später.

Kompatibel: True Basic

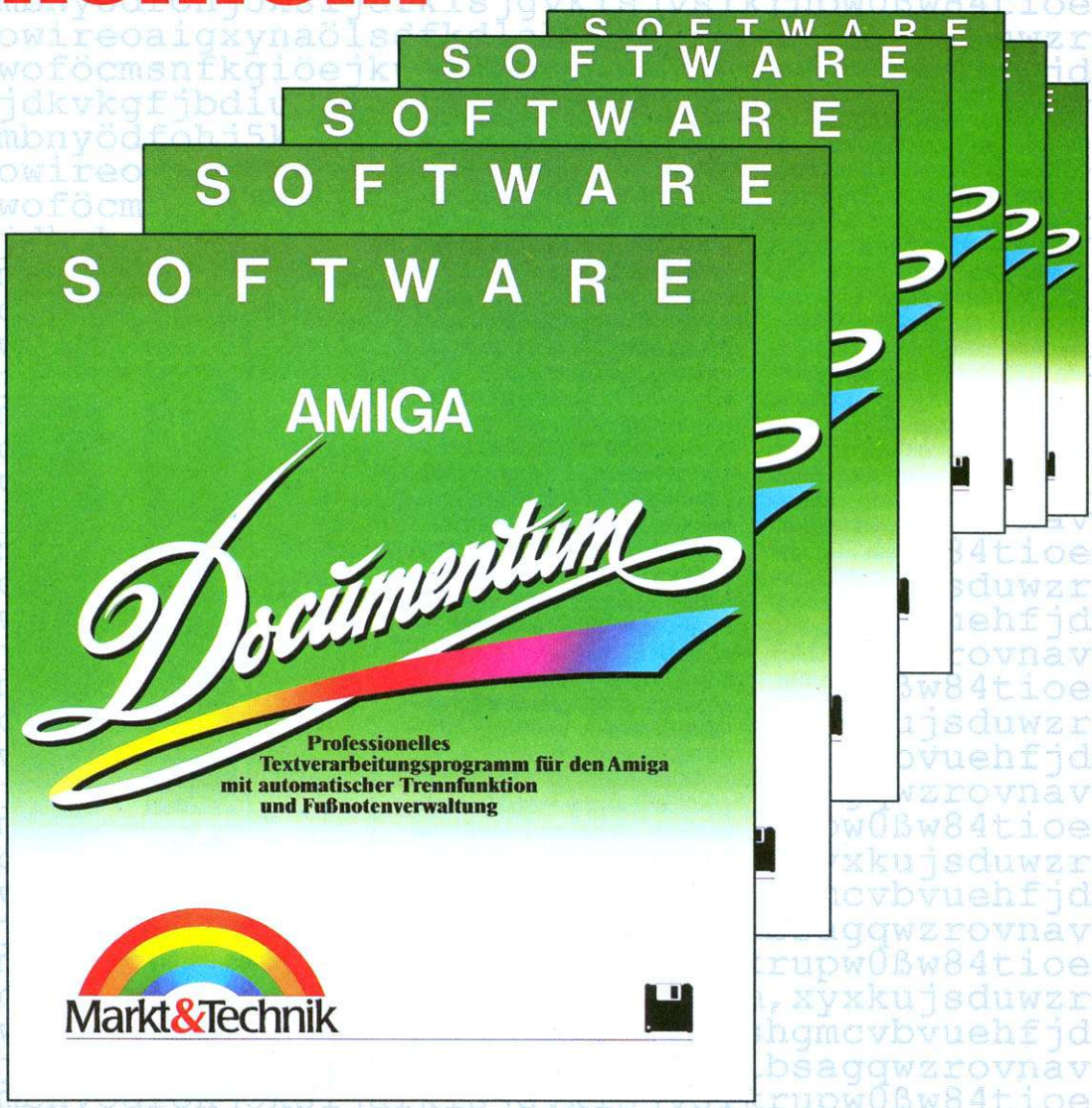
True Basic ist ein in mancher Hinsicht ein untypischer Compiler. Zunächst besitzt er (was sonst bei Compilern nicht üblich ist) eine eigene Benutzeroberfläche mit Editor und Direktmodus. Außerdem wird der Quelltext nicht wirklich in Maschinensprache, sondern in einen sogenannten »Speed Code« übersetzt, der normalerweise immer noch schneller als ein Quelltext-Interpreter ist, aber auch kürzere Interpretier-Zeiten aufweist. Dadurch sind aber auch bei True Basic keine »Stand-Alone«-Programme möglich, der Speed Code muß noch in Maschinencode umgesetzt werden. Dafür weiß aber auch hier der Speed Code-Interpreter, von wo im Quelltext die gerade bearbeitete Befehlssequenz stammt. Entsprechend bietet True Basic auch eine Trace-Option zur leichteren Fehlerbeseitigung.

Die Benutzeroberfläche von True Basic ist der von Amiga-Basic um einiges voraus. So ist die Haupt-Arbeitsfläche das Editor-Fenster, und nicht wie bei Amiga-Basic das Kommando- und Ausgabefenster. Bei Bedarf können allerdings diese beiden Fenster auch bei True Basic eingeblendet werden. Zwar sind die Fenster auch nicht die schnellsten, die der Amiga je gerettet und aufgefrischt hat, doch ist die Geschwindigkeit bei weitem nicht so gering wie bei Amiga-Basic.

Meldungen des Programms, wie Fehler und ähnliches, erscheinen invertiert in einem abgeteilten Feld am unteren Fensterrand. Durch den meist breiten farbigen Balken sind die Meldungen schön auffällig und trotzdem wesentlich angenehmer als die langsamen Fenster von Amiga-Basic.

Die meisten (aber auch nicht alle) Befehle der Oberfläche sind nicht nur über Menüs, sondern auch mit Tastenkombinationen erreichbar. Zudem bietet der Editor einige für Textverarbeitungen typische Funktionen, beispielsweise Suchen, Ersetzen und Einfügen.

Zweiflern zum Trotz: Documentum



Textverarbeitung auf dem Amiga par excellence

Textverarbeitungsprogramme auf dem Amiga wurden bisher von Profis belächelt. Sie waren langsam wie Schreibmaschinen, bunt wie Kindergeburtstage, absturzsicher wie Starfighter oder teuer wie Schweizer Uhren. Bis Documentum kam... Rasend schnell • Automatische Trennhilfe • Fußnotenverwaltung • Editier-/WYSIWYG-Bildschirm • Blockbearbeitung • Kopf-/Fußzeilen • Seitennumerierung • Blockbearbeitung • Alle Amiga-Zeichensätze nutzbar • Tastatur- oder Mausbedienung • Buchdruckoption • Suchen/Ersetzen • NLQ-Druck für alle Drucker • multitaskingfähig • und vieles, vieles mehr...

Für Briefschreiber, Studierende, Autoren, Bürokraten – eigentlich für alle.

Bestell-Nr. 54122

DM 149,–* (sFr 135,–*/öS 1490,–*)

* Unverbindliche Preisempfehlung

220/812

Markt&Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computer-Fachgeschäften oder bei Ihrem Buchhändler.

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 587 1393-0; Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 67 75 26; Ueberreuter Media Verlagsges.mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 48 15 43-0.

Eine ganz ungewohnte, aber sehr praktische Fähigkeit der Oberfläche stellen die sogenannten DO-Programme dar. Dies sind externe Unterprogramme, die während der Bearbeitung des Quelltextes geladen und abgearbeitet werden. Das besondere an den DO-Programmen ist, daß an sie der gesamte Quelltext im Speicher in Form eines String-Arrays sowie ein Argument übergeben wird. Dies läßt eine vielfältige Manipulation des Quelltextes zu, da dem DO-Programm der gesamte Sprachumfang von True Basic zur Verfügung steht. Drei solche Programme werden gleich mitgeliefert und sind direkt aus dem Menü aufrufbar. »DO FORMAT« strukturiert den Quelltext automatisch nach Schleifen, Unterprogrammen und Entscheidungen durch Einrückung. Dadurch muß der Programmierer bei der Eingabe des Quelltextes nicht auch noch an die Strukturierung denken, und hat am Schluß dennoch ein klares, übersichtliches Listing.

Strukturierte Programmierung

»DO NUM« und »DO UNNUM« sind sozusagen Umkehrfunktionen und numerieren beziehungsweise denumerieren die Zeilen eines Quelltextes komplett. In Sachen Strukturierung ist True Basic nämlich eine sehr extreme Sprache: Entweder man numeriert jede Zeile, dann können auch Zeilen mit GOTO und GOSUB angesprungen werden, oder man verzichtet auf die inzwischen überholte Nummerierung — und hat dafür auch keinen einzigen Sprungbefehl zur Verfügung, und muß allein mit Schleifen und bedingter Ausführung zurechtkommen. Dadurch erzieht True Basic den Benutzer natürlich automatisch zum guten Programmierstil, Anfänger aber werden sicher ihre Probleme haben.

Natürlich versorgt True Basic den Programmierer auch mit einer mächtigen Schleifen-, Entscheidungs- und Unterprogrammtechnik. Während unter Amiga-Basic bedingte Schleifen mit den beiden Befehlen WHILE und WEND in genau einer Kombination abgedeckt sind, kann bei True Basic die Schleife unbedingt, mit Bedingung am Anfang oder Schluß oder sogar mit beidem ausgeführt werden. Zusätzlich ist das Verlassen der Schleife unmittelbar durch »EXIT DO« möglich.

Auch bietet True Basic in Sachen Entscheidungen wesentlich mehr Komfort durch »SELECT CASE«. Statt die einzelnen Zustände einer Variable mit mehreren »IF...THEN«-Befehlen abzufragen, werden nach diesem Befehl den angegebenen Variablen-Inhalten bestimmte Anweisungssequenzen zugeordnet.

Interne und externe Unterprogramme und Funktionen unterscheiden sich vor allem durch die Globalität beziehungsweise Lokalität ihrer Variablen. Zwar sind lokale Variablen in Unterprogrammen nichts neues, aber unter True Basic dürfen die Variablen der verschiedenen Programmabschnitte durchaus gleiche Namen tragen, ohne daß Werte durcheinander geraten. Außerdem kann man oft gebrauchte externe Unterprogramme und Funktionen in Bibliotheken speichern, die dann vor der Ausführung des Hauptprogramms automatisch nachgeladen werden und zur Verfügung stehen.

Angenehm ist die Arbeit auch mit Array-Variablen. So werden mit einem Befehl ganze Arrays aus DATA-Zeilen, Dateien oder mit INPUT eingelesen, mit beliebigen Werten initialisiert oder auch komplett ausgegeben. Praktisch für mathematische Anwendungen (auch für mehrdimensionale Grafik) sind die Befehle und Funktionen zur Matrizen-Arithmetik und Skalarrechnung.

Besonders interessant ist gerade beim kompatiblen True Basic auf dem Amiga natürlich die Ausnutzung der Sound- und Grafikeigenschaften. Zu-

Vernachlässigte Amiga-Vorteile

nächst einmal sind die Befehle zur Sound-Programmierung enttäuschend. Zwar kann man ganze Melodien mit dem Befehl PLAY und einem String mit entsprechenden Steuerzeichen abspielen lassen, allerdings nur in einem Einheitsklang — ein PC bietet nun mal keine frei definierbaren Hüllkurven, seine Abstammung kann der Compiler hier nicht verleugnen.

Die größte Enttäuschung erlebt man aber, will man sich an die Grafikprogrammierung wagen. Wo bei den anderen Sprachen erst einmal die Definition eines eigenen Bildschirms und Fensters nötig ist, sucht man bei True Basic vergebens nach solchen Befehlen. Beim Start eines Programms wird automatisch ein rahmenloses Fenster geöffnet, in dem dann alle Ein- und Ausgaben stattfinden. Änderungen oder zusätzliche Fenster sind also nicht möglich, und da True Basic aus Amerika stammt, ist sogar das eine Fenster nur 200 Punkte hoch. Was True Basic an Fenster-Befehlen zu bieten hat, erinnert stark an das Basic 7.0 des C 128: Ein- und Ausgabe werden lediglich auf bestimmte Ausschnitte des Bildschirms beschränkt, allerdings ohne Rettung des Inhalts.

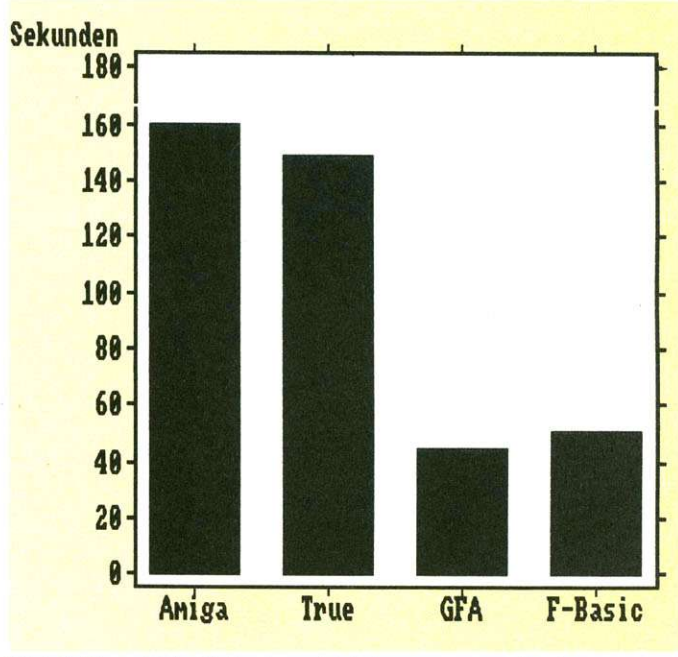


Bild 3. Das dritte Kriterium: Zeichnen von grafischen Objekten und die Geschwindigkeit der Zufalls-Funktion (Random-Generator)

Ebenso vermißt man auch Sprites und Bobs. Diese animierten grafischen Objekte sind doch hin und wieder recht praktisch, allerdings ist für semiprofessionelle Programme dieser Umstand noch am ehesten zu verkraften. Dafür hält True Basic neben den Standard-Befehlen noch etwas ganz besonderes bereit:

Die »PICTURES« sind interne Unterprogramme, in denen beliebige Zeichnungen (von einfachen Strichmännchen bis zu komplizierten Grafiken) programmiert werden. Diese PICTURES können dann mit verschiedenen Parametern aufgerufen werden, welche vom Programm automatisch auf die Koordinaten aller Zeichenbefehle umgerechnet werden: »SHIFT« setzt das Bild an eine beliebige Stelle, »SCALE« streckt oder staucht das Bild, »ROTATE« dreht es um den Koordinatenpunkt 0/0 und »SHEAR« kippt alle vertikalen Linien um einen beliebigen Winkel, wobei die horizontalen Linien horizontal bleiben.

Eine äußerst wichtige Besonderheit des Amiga fehlt True Basic: Sie können zwar Assembler- und C-Programme schreiben und einbinden (das Verfahren dazu wird im Handbuch erklärt), jedoch der Aufruf der Amiga-Bibliotheken beziehungsweise der darin enthaltenen Funktionen ist direkt nicht vorgesehen. Dies mag nun auf den ersten Blick nicht so wild erscheinen, aber es setzt auf jeden Fall Kenntnisse in einer der beiden genannten anderen Sprachen voraus und macht insgesamt die Programmerstellung wesentlich komplizierter und zeitraubender.

Als Beispiel für Ergänzungen wird eine Library mitgeliefert, die unter anderem einfache File-Selectors, Farbscrolling und den Aufruf von DOS-Befehlen unterstützt.

Bei der Messung der Ausführungsgeschwindigkeit von True Basic-Programmen erlebt man eine böse Überraschung. Trotz Compilation sind die Programme durchschnittlich nicht wesentlich schneller als die von Amiga-Basic, teilweise sogar langsamer. Lediglich bei reiner Integer- und Schleifenprogrammierung ist ein deutlicher Geschwindigkeitsvorteil erkennbar, aber in der Praxis dürften solche Programme

eher selten sein. Wesentlich wichtiger sind die Fließkomma-Operationen, und da ist True Basic ganze 35 Prozent langsamer als der Interpreter Amiga-Basic — eine fast beschämende Leistung.

True Basic wird mit zwei Handbüchern in englischer Sprache ausgeliefert, die sich gegenseitig ergänzen. Das »Reference Manual« ist das universelle Handbuch, das unabhängig vom verwendeten Computer sämtliche Befehle und Funktionen nach Sachgebieten geordnet vorstellt und mit vielen Programmbeispielen erläutert. Der Computerspezifische »User's Guide« enthält nicht alle Standardbefehle, dafür aber die jeweils berücksichtigten Besonderheiten des Computers, zum Beispiel die Grafikmodi, Dateinamen und Spezial-Libraries. Insgesamt sind die englischen Handbücher sehr umfangreich aber übersichtlich.

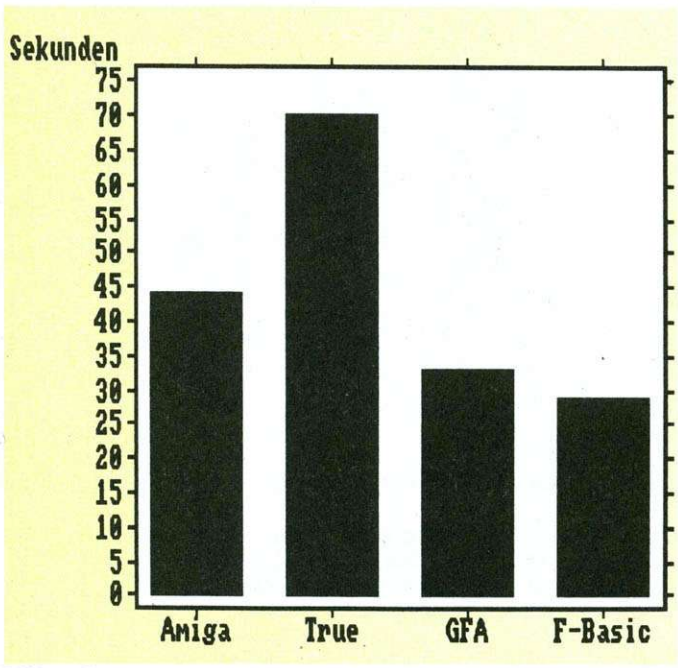


Bild 4. Schreiben in eine sequentielle Datei

Eigentlich läßt sich True Basic gegenüber dem ebenfalls 200 Mark teuren Konkurrenten GFA-Basic nur durch seine Kompatibilität zur PC-Version rechtfertigen. Wer unbedingt die unter True Basic auf dem PC geschriebenen Programme weiterverwenden will, für den ist True Basic natürlich eine feine Sache, doch ansonsten kann man True Basic vor allem wegen der niedrigen Geschwindigkeit und dem in bezug auf Amiga-Spezialitäten winzigen Befehlsumfang nicht unbedingt empfehlen.

Wie True Basic ist auch **GFA-Basic** eine Programmumsetzung: Auf dem Atari ST hat sich dieser Dialekt schon lange zum Standard entwickelt. Ob das auf dem Amiga auch gelingt, bleibt abzuwarten. Jedenfalls hat dieser recht gute Aussichten. Mit einer als geradezu ideal zu bezeichnenden Benutzeroberfläche und einem satten Sprachumfang von über 340 Befehlen hat dieser Dialekt einiges zu bieten.

GFA-Basic: eine Art Luxus

GFA-Basic ist ein Interpreter mit integriertem Editor, und dieser Editor ist einer der ganz großen Vorteile von GFA-Basic. Das Erscheinungsbild ist auf den ersten Blick etwas ungewohnt. Die Ursache dafür liegt in der Bedienung. Es können nämlich nur einige wenige Befehle über ein Menü aufgerufen werden, die meisten (und wichtigsten Kommandos) liegen auf den Funktionstasten. Und da nun mal die Funktionen von 10 doppelt belegten Funktionstasten recht schwer zu merken sind, ist deren Belegung ständig am oberen Fensterrand abgebildet.

Die übrigen Editorbefehle, die weder auf den Funktionstasten noch im Menü Platz fanden (und das sind immer noch recht viele), ruft man über Control-Kombinationen auf. Das garantiert zwar am Anfang häufiges »Handbuchblättern«, aber Übung macht den Meister. Dafür stehen bei GFA-Basic Befehle zur Verfügung, die oft nur bei sehr guten Textverarbeitungen zu finden sind. Suchen und Ersetzen sind vorhanden, ebenso komfortable Block-Operationen wie Kopieren, Verschieben, Speichern, Ausdrucken und ähnliches. Man kann beliebig (und übrigens auch recht schnell) im Listing herumspringen, auch können Zeilen ihrer Nummer nach angesprungen werden. Nervenschonend wirkt auch die »Undo«-Funktion, wenn versehentlich eine korrekte Zeile gelöscht wurde.

»LLIST« zur Ausgabe des Listings auf den angeschlossenen Drucker hat es in sich. In das Listing eingefügte Pseudo-Codes stellen das Seitenformat, Kopf- und Fußnoten, Zeilennumerierung und beliebige Steuerzeichen ein, sogar Datum und Uhrzeit können eingefügt werden. Die Uhrzeit wird übrigens wie die Zeilennummer ständig am oberen Fensterrand angezeigt.

Bei der Eingabe eines Listings überprüft der Editor dann automatisch jede Zeile auf korrekte Syntax. Hat man sich erst einmal daran gewöhnt, keine unvollständigen Zeilen mit dem Cursor verlassen zu dürfen, erspart das bei Tippfehlern viele Testläufe und damit auch Zeit. Zusätzlich wird jede Zeile auch noch formatiert: Der Editor löscht überflüssige Leerzeichen beziehungsweise Leerzeilen, und rückt die Zeile entsprechend den Schleifen- und Entscheidungsbefehlen ein. Außerdem kann der Benutzer sogar einstellen, ob Befehle, beziehungsweise Variablennamen, automatisch groß, klein oder klein mit großem Anfangsbuchstaben dargestellt werden sollen.

Ein Befehlsumfang von über 340 Befehlen mag einige vielleicht abschrecken, doch sind ja nicht alle Befehle von Anfang an nötig. Viele Befehle dienen nur der Programmoptimierung. Insofern sind diese zahlreichen Befehle und Funktionen ein großer Vorteil, zumal sich darunter einige ausgesprochene Leckerbissen befinden.

Bezüglich der Variablentypen, beziehungsweise deren Überführung ineinander, ist GFA-Basic ähnlich exakt wie Amiga-Basic. Zusätzlich stehen auch noch die Typen »Byte« (Werte von 0 bis 255) und »Boolean« (Werte logisch wahr oder falsch) zur Verfügung. Zur Kennung der verschiedenen Typen verwendet man normalerweise Postfixe, jedoch kann man auch zu Beginn des Programms beliebige Namensanfänge bestimmten Typen zuordnen. So sind dann zum Beispiel alle Variablen, deren Namen mit »By« beginnen, vom Typ Byte. Pointer-Befehle ermöglichen den Zugriff auf Variablen über die Adressen. Entsprechend existieren auch vielfältige PEEK- und POKE-Befehle, die je nach Variablentyp eines odere mehrere Bytes lesen oder schreiben. Auch werden Integer-Werte in binäres, oktales und hexadezimales Format konvertiert.

Die Befehle »QSORT« und »SSORT« sortieren Arrays beliebigen Typs in alphanumerische oder frei definierbare Reihenfolge nach dem Shellsort- oder Quicksort-Verfahren. Ebenfalls wichtig im Umgang mit Datenfeldern sind die Befehle zum Entfernen beziehungsweise Einfügen von Daten in Arrays.

Bei den numerischen Funktionen wurde besonders Wert auf Befehle zum Runden und Ganzzahligmachen von Fließkommavariablen gelegt. Auch der Zufallszahlengenerator bietet die ungewohnte Option, auch Ganzzahlen zwischen Null und einem beliebigen Wert zu erzeugen, wahlweise mit ungleich verteilter Wahrscheinlichkeit. Der Sinn beispielsweise der Funktionen »ADD« und »SUB« als Ersatz zu den üblichen numerischen Operatoren mag nicht so recht einleuchten. Lediglich das äußere Erscheinungsbild der Listings läßt sich dadurch den Sprachen C und Assembler etwas angleichen. Diesem Trend entsprechend stehen auch Assemblerbefehle für Bit-Operationen zur Verfügung (Bits löschen, setzen, testen, invertieren und rotieren).

Bei den vielen nützlichen Befehlen von GFA-Basic verwundert es, daß DATA-Zeilen mit Schleifen und nicht (wie bei den Konkurrenten True Basic und F-Basic) mit Hilfe eines einzigen Befehls eingelesen werden.

Im Umgang mit Diskettendateien sind einige unübliche Befehle und Funktionen geboten: »DFREE« ermittelt den freien Platz auf der Diskette, »DIR$« das aktuelle Verzeichnis, welches mit »CHDIR« gewechselt wird. »DIR« und »FILES« gibt das Directory in verschiedenen Formaten auf einem beliebigen Ausgabekanal (Drucker, Bildschirm, Datei) aus. »EXIST« ermittelt, ob eine bestimmte Datei im Verzeichnis vorhanden ist. Komplex aber einfach zu handhaben ist der

Befehl »FILESELECT«, der eine Fileselector-Box erstellt und steuert — eine Amiga-Spezialität, welche die Amiga-Basic-Entwickler noch nicht einmal in die programmeigene Benutzeroberfläche einbauten. Sequentielle, relative und Binär-Dateien (letztere zum Ablegen von Speicherbereichen) werden von GFA-Basic komfortabel unterstützt.

Warum nur GFA-Systemtechnik auf die Idee kam, einen Hardcopy-Befehl in den Basic-Dialekt zu integrieren, bleibe dahingestellt — auf jeden Fall aber hebt auch dieses Extra die Qualität von GFA-Basic. In Sachen Schleifen und Entscheidungen hat GFA-Basic genauso viel zu bieten wie True Basic, also alles, was das Herz des Struktur-begeisterten Programmierers begehrt. Die Unterprogramme und Funktionen entsprechen mit Lokalvariablen und Parameter-Übergabe dem Standard, zusätzlich wurden auch nicht die von den älteren Dialekten her bekannten einzeiligen Funktionen vergessen. Trotzdem steht GFA-Basic in diesem Kriterium True Basic deutlich nach.

Auch alle üblichen Grafikbefehle für Punkte, Linien, Rechtecke, Kreise und beliebige Polygone sind vorhanden, und wieder gibt es ein ungewohntes Extra. »DEFLINE« definiert ähnlich wie in CAD-Programmen das Aussehen der Linien für die Standard-Grafikbefehle nach Dicke, Muster und Anfangs- beziehungsweise Endsymbolen der Linien. Der »DRAW«-Befehl unterstützt optional die sogenannte »Turtle-Grafik«.

Einige ungewöhnliche Extras

Die grafischen Objekte und Bobs sind ähnlich gut zu handhaben wie in Amiga-Basic. Allerdings ist die Sprite-Unterstützung minimal, Animation ist nicht zu realisieren.

Schließlich unterstützt GFA-Basic auch noch Maus- und Joystick-Kontrolle, Screens und Fenster, Menüs sowie Ereignisverarbeitung und sogar einfache Requester.

Zudem hat der Benutzer freien Zugriff auf sämtliche Amiga-Libraries, der Umgang wird sogar vom Interpreter erleichtert, zum Beispiel durch die Bereitstellung der Basisadressen in Systemvariablen. Dieser Umstand und die ausführliche Erklärung der zur Zeit vorhandenen Funktionen im Handbuch verdienen einen großen Pluspunkt. Einzige Einschränkung: wie bei Amiga-Basic fehlt der im Betriebssystem oft benötigte Datentyp Struktur. Dieser muß bei Bedarf vom Programmierer über die Variablen-Pointer selbsterstellt werden.

Bei der Messung der Ausführungszeiten erlebt man bei GFA-Basic angenehme Überraschungen. Integer- und Fließkomma-Operationen sowie Grafikausgabe erfolgen durchschnittlich um zwei Drittel schneller als bei Amiga-Basic. Im dritten Benchmarktest zur Grafikausgabe zeigte sich der Interpreter GFA-Basic sogar mehr als zehn Prozent schneller als der Compiler F-Basic, was wohl vor allem an dem durchdachten Zufallszahlen-Generator liegen dürfte. Wo GFA-Basic automatisch ganzzahlige Werte passend zu den Koordinaten erzeugt, braucht es bei F-Basic wie bei den anderen Dialekten erst noch eine Fließkomma-Multiplikation und eine Rundung. Im Umgang mit sequentiellen Dateien ist GFA-Basic immer noch ein Drittel bis ein Viertel schneller als Amiga-Basic.

Ein dickes Lob verdient das Handbuch. Hier sind die Befehle und Funktionen nach Sprachelementen geordnet und in Format und Wirkung hinreichend erklärt. Ein Beispiel für jeweils zwei bis drei verwandte Befehle erleichtert es, den Einstieg in die Praxis zu finden. Insgesamt ist das Handbuch in seiner Ausführlichkeit dem Sprachumfang angemessen und klar strukturiert.

GFA-Basic hat eigentlich alles, was ein gutes Basic braucht: Einen komfortablen Editor, den teilweise als luxuriös zu bezeichnenden Befehlsumfang und eine hohe Ausführungsgeschwindigkeit.

Auch die für fortgeschrittene Programme nötige Unterstützung des Betriebssystems wird geliefert, wobei allerdings das Fehlen von Strukturen den Programmierer voll fordert.

Die selbstgeschriebenen Programme sind außer zu Privatzwecken praktisch nicht verwendbar, da immer der Interpreter vorhanden sein muß — und der darf ja aus urheberrechtlichen Gründen nicht weitergegeben werden. Im Vergleich mit F-Basic (erzeugt Stand-alone-Programme) und Amiga-Basic, das sowieso jeder besitzt, verliert GFA-Basic hier an Boden.

Wer also viel programmiert und auf die Verbreitung seiner Programme keinen Wert legt, für den ist GFA-Basic sehr zu empfehlen. Bei den Besitzern eines Amigas und des Atari ST dürfte auch die weitgehende Kompatibilität der beiden Versionen ein ausschlaggebendes Argument sein.

Anspruchsvoll: F-Basic

F-Basic von der bisher unbekannten Firma Delphi Noetic Systems stand uns nur in einer amerikanischen Vorversion zur Verfügung. Ob dieses Programm in Deutschland auf den Markt kommt, ist bisher unbekannt. Da es sich allerdings um einen hervorragenden Basic-Dialekt handelt, und Interessierte sich eventuell direkt an die Firma wenden können (die Anschrift finden Sie am Textende), stellen wir Ihnen F-Basic hier vor.

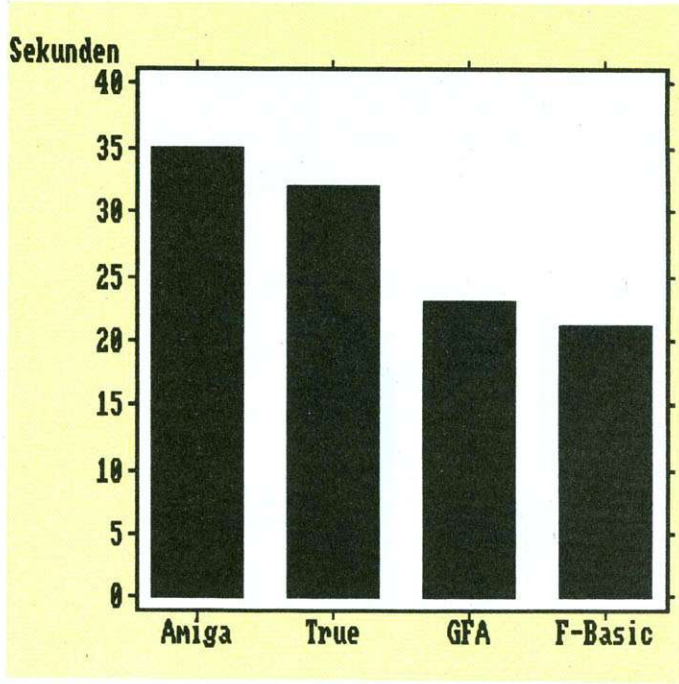


Bild 5. Lesen von Daten aus einer sequentiellen Datei

Bei F-Basic handelt es sich um einen reinrassigen Disketten-Compiler. Das heißt, F-Basic ist ein Compiler und sonst gar nichts: Editor oder Benutzeroberfläche stehen nicht zur Verfügung. Dies hat den Nachteil, daß bei der Programmerstellung ständig Compiler und Editor nachgeladen werden müssen. Dafür kann man aber den besten Editor oder das beste Textverarbeitungsprogramm verwenden, das man besitzt.

F-Basic wird vom CLI aus aufgerufen. Es compiliert den Quelltext aus der angegebenen Quelldatei in die angegebene Zieldatei. Das Endergebnis kann als Stand-alone-Programm bezeichnet werden, obgleich noch eine 37 KByte umfassende Library-Datei beim Start des Programms im gleichen Verzeichnis vorhanden sein muß. Der Compiler selber hat mit den nötigen Zusatz-Libraries eine Gesamtlänge von ungefähr 200 KByte. Er ist somit das längste Programm in diesem Test (Amiga-Basic: 108 KByte, True Basic: 148 KByte, GFA-Basic: 122 KByte).

Damit kann F-Basic für Benutzer ohne Speichererweiterung sehr unkomfortabel werden, denn folgende Operationen sind während der Testphase eines Programms nötig: Editor laden, Quelldatei laden, Speichern der Quelldatei, Compiler laden, Quelldatei laden, Speichern der Zieldatei und schließlich auch noch das Laden der Zieldatei zum Ausprobieren des Ergebnisses. Benutzerfreundlichkeit läßt sich da nur noch bei der Verwendung einer RAM-Disk oder einer Festplatte erkennen.

Wer aber diese Alternative sowie einen guten Editor besitzt (der kostenlose Editor MicroEmacs von der Extras-Diskette leistet da bereits gute Dienste), dem steht mit F-Basic ein ausgereiftes System zur Verfügung, für das schon jetzt interessante Updates angekündigt sind. In der uns vorliegenden Version umfaßt F-Basic ungefähr 240 Befehle

und liegt damit hier zwischen Amiga-Basic (180) und GFA-Basic (340).

Zunächst bietet F-Basic dem Benutzer einige praktische Compiler-Optionen und Pseudo-Befehle. So wird zum Beispiel auf Wunsch nicht in die Zieldatei, sondern in den Speicher compiliert und nach der Compilation das Ergebnis sofort aufgerufen. Eine andere Option ruft, sollte während der Compilation ein Fehler auftreten, automatisch das angegebene Programm (normalerweise ist das der Editor) auf.

Dem allgemeinen Trend folgend hat F-Basic sehr viele Elemente der Sprachen Assembler, C und vor allem Pascal übernommen. Jedes Programm muß mit »PROGRAM« und dem Namen des Programms beginnen. Vor dem ersten Befehl steht die Deklaration der Variablen beziehungsweise Konstanten. Zur besseren Speichernutzung wird jede Textvariable mit einer maximalen Länge deklariert, die natürlich im Programm nicht überschritten werden darf.

Leider unterstützt F-Basic nicht den Variablentyp Boolean, jedoch dürfte dies nicht so stark ins Gewicht fallen. Wichtiger ist dagegen, daß F-Basic bei den Fließkommavariablen vom IEEE-Standard abweicht, was sich in höherer Genauigkeit und Rechengeschwindigkeit auswirkt. Damit sind auch Zahlen bis zu 10 hoch 10000 darzustellen (im IEEE-Format nur bis 10 hoch 38 bei einfacher Genauigkeit) — beeindruckend. Das Problem, vor allem Feldvariablen mit bestimmten Werten aus DATA-Zeilen zu belegen, haben die Entwickler von F-Basic elegant gelöst. In den DATA-Zeilen werden abwechselnd die zu belegenden Variablen und die entsprechenden Werte aufgeführt, so daß aus dem Quelltext sofort hervorgeht, welche Variable nun mit welchen Werten gefüllt wird.

Viele elegante Lösungen

Doch damit nicht genug, F-Basic unterstützt noch zwei weitere Variablentypen. Variablen vom Typ »PTR_TO« enthalten Zeiger auf andere Variablenobjekte im Speicher. Außerdem stehen die für C typischen und damit für die Amiga-Libraries unerläßlichen Strukturen zur Verfügung. Mit diesen beiden Variablentypen und der Möglichkeit, Amiga-Libraries zu öffnen, hat ein F-Basic-Programmierer das gleiche Werkzeug in der Hand wie ein C-Programmierer.

Besonderen Wert legte man beim Sprachumfang anscheinend auf die numerischen und alphanumerischen Funktionen. Hier hat F-Basic teilweise mehr zu bieten als GFA-Basic. Allerdings können zum Beispiel Integer- und Textfelder nur in Standardreihenfolge sortiert werden, der Zufallszahlengenerator für Integerzahlen liefert immer Werte zwischen Null und der größtmöglichen Integerzahl, und die Abfrage der Systemzeit ist mit einigem Aufwand verbunden. Als wertvoller Ausgleich dafür existiert eine Fülle von Möglichkeiten für den Umgang mit Textvariablen, die kaum zu beschreiben ist. Kurz gesagt, es gibt nichts, was es zum Verknüpfen, Durchsuchen, Formatieren oder Umgestalten von Strings nicht gibt. Daraus resultiert eine beträchtliche Verkürzung von Quelltext und Ausführungszeit. Unter vertretbarem Aufwand sind Operationen möglich, die bisher gar nicht realisierbar waren. Davon profitiert jede selbsterstellte Dateiverwaltung oder Textverarbeitung.

Leider fehlen F-Basic die unbedingten und doppelt bedingten Schleifen, ansonsten sind alle Entscheidungs-, Schleifen- und Sprungbefehle vorhanden.

Der Compiler bietet externe Funktionen und Unterprogramme mit lokalen Variablen und Parameter-Deklaration in Pascal-Manier. Um Tipparbeit zu sparen und den Quelltext übersichtlicher zu machen, lassen sich häufig verwendete Deklarationen oder andere Programmteile separat speichern und mit »INCLUDE« oder »APPEND« integrieren. F-Basic bindet dann während der Compilation den Befehl ein — auch dies erinnert stark an die »Header«-Dateien von C.

Die Sound-Qualitäten des Amiga werden von F-Basic hinreichend ausgenutzt, wobei beliebig große Byte-Felder die Wellenformen definieren. Tonhöhenangaben erfolgen leider nur in der expliziten Frequenz, nicht aber nach Notennamen. Zudem fehlt im Handbuch eine Tabelle zur Umrechnung von bestimmten Notenwerten in Frequenzen.

Bei der Unterstützung von Screens und Windows bleiben eigentlich keine Wünsche offen. Ungewohnt ist die Möglichkeit, Windows auch nach dem Öffnen in Größe und Lage zu verändern. Die Standard-Grafikbefehle sind alle vorhanden, Füllmuster kann der Benutzer selbst definieren. Leider fehlen Befehle zur Sprite-Programmierung, lediglich für die langsameren Bobs existieren acht Befehle. Menüs wiederum lassen sich einfach und mit allen Spezialitäten (außer Untermenüs) programmieren. Daß bei den Befehlen für die Bearbeitung von Ereignissen solche zur Fehlererkennung anscheinend vergessen wurden, ist ein Nachteil von F-Basic. Damit ist jeder Fehler während der Programmausführung fatal und führt zum unbedingten Programmabbruch.

Zusätzlich bietet F-Basic ähnlich GFA-Basic einige Befehle zur Speichermanipulation und zum Aufruf von Maschinenroutinen. Darüber hinaus können Programme auch auf die Register beziehungsweise über die Register adressiert auf den Speicher zugreifen. Daß F-Basic auch die Amiga-Libraries öffnen kann, wurde bereits erwähnt. Allerdings gibt es eine Einschränkung: Sowohl bei der Compilation als auch bei der Ausführung des Programmes muß eine zusätzliche Datei namens FastSysLib erreichbar sein. Sie enthält eine Auflistung mit sämtlichen in den Libraries enthaltenen Funktionen und Routinen. Dadurch wird zwar bei einem Update der Amiga-Libraries auch ein Update der FastSysLib nötig, aber diesen Service wird Delphi Noetic Systems hoffentlich anbieten. Profis werden die Änderungen vielleicht sogar selber vornehmen können.

Übrigens können nicht nur DOS-Befehle aus einem F-Basic-Programm heraus aufgerufen werden, auch die Programme selbst sind wie DOS-Befehle mit beliebigen Parametern aufzurufen. Eigenen neuen oder verbesserten DOS-Befehlen steht damit — und mit der entsprechenden Amiga-Library — nichts mehr im Wege.

Von der Ausführungsgeschwindigkeit her stellt F-Basic alle Konkurrenten weit in den Schatten. Besonders frappierend sind die Werte für Integer- und Fließkomma-Operationen. Hier ist F-Basic bis zu 95 Prozent schneller als Amiga-Basic — ein Wert, der sich vor allem in rechenintensiven Programmen durch wesentlich niedrigere Ausführungszeiten äußert. Daß im dritten Benchmarktest zur Grafikausgabe F-Basic etwas langsamer ist als GFA-Basic, dürfte wohl an dem dort besseren Zufallszahlen-Generator liegen. Übrigens ist der Algorithmus bei F-Basic so ausgelegt, daß sich Zufallszahlen erst nach 2 hoch 32 Durchläufen wiederholen.

Die englische Vorversion der Programmanleitung ist nach Sprachelementen geordnet und erklärt fast jeden Befehl mit einem Programmbeispiel. Neben dem umfangreichen Kapitel zu den normalerweise recht ungewohnten String-Operationen enthält das Handbuch auch einen besonderen Abschnitt zur Programmoptimierung sowie ausführliche Befehls-, Fehler- und Stichwortverzeichnisse.

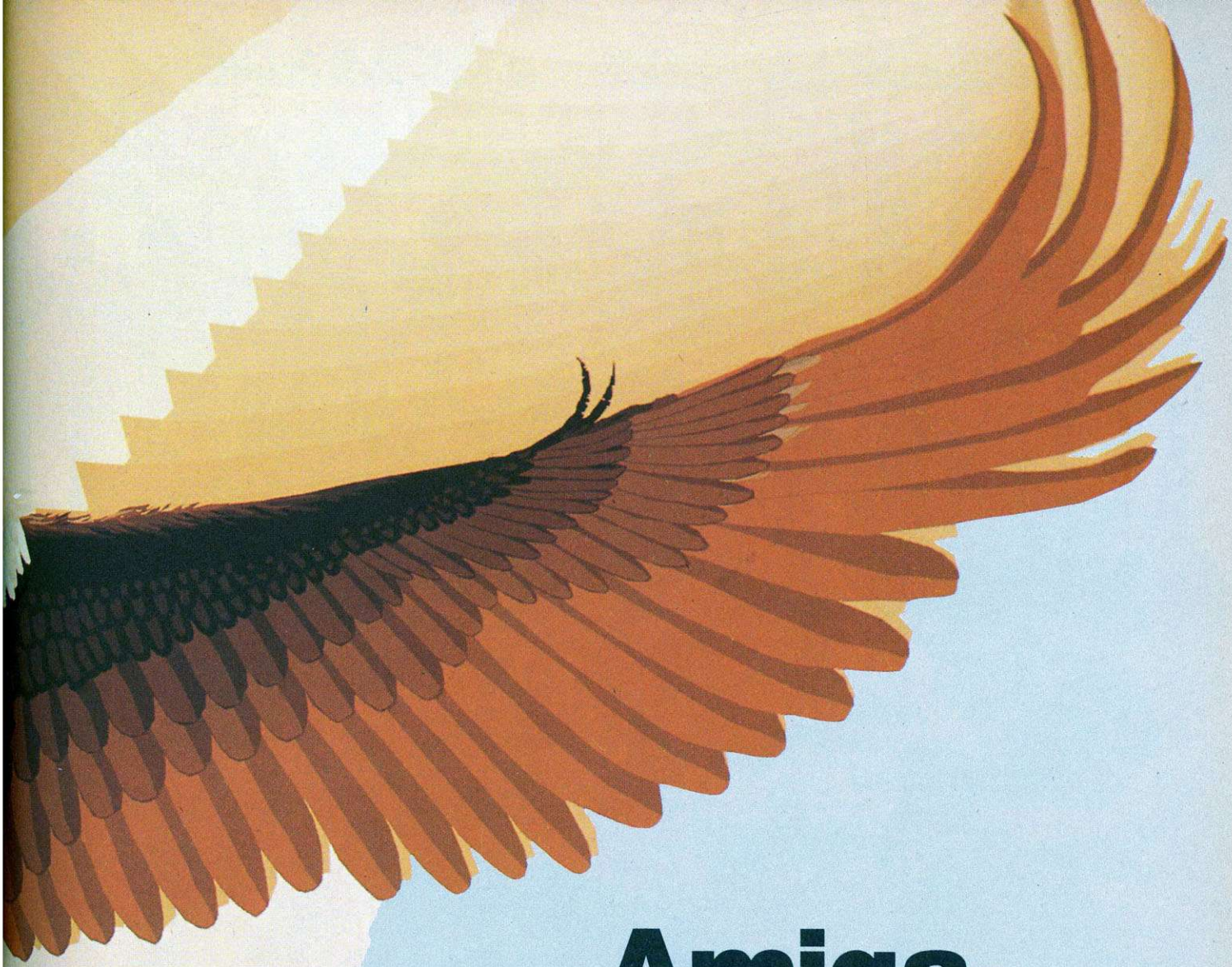
Bereits jetzt sind um wesentliche Elemente verbesserte Versionen von F-Basic angekündigt, die einige Nachteile der vorliegenden Version ausbessern. So werden doppelt genaue Fließkommazahlen, relative Dateien, komplexe Zahlen, Assembler-Quellcode, Animationsroutinen sowie IFF-Dateien unterstützt werden. In welcher Version F-Basic auf dem deutschen Markt erscheint und ob dies überhaupt geschieht, ist derzeit allerdings unbekannt.

Zwar kann sich F-Basic rühmen, als einziger Dialekt im Test Stand-alone-Programme (mit den erwähnten Einschränkungen) zu erstellen. Auch die Geschwindigkeit läßt es als einen echten Konkurrenten zu den C-Compilern erscheinen. Doch gibt es leider noch einen großen Haken: Der Hersteller von F-Basic hält nämlich die Rechte an der für die Ausführung jedes Programms benötigten Library. Das bedeutet, daß die Library (wie bei den anderen Dialekten der Interpreter) nicht weitergegeben werden darf. Damit ist F-Basic ebenfalls nicht für den semi- oder professionellen Einsatz geeignet. Also steht F-Basic deutlich hinter dem umfangreichen und komfortablen GFA-Basic zurück. Es läßt sich lediglich für private Anwendungen empfehlen, bei denen zwar Geschwindigkeit an erster Stelle steht, C aber nicht in Frage kommt.

(Nikolaus Huber/rs)

Die derzeitige Bezugsquelle für F-Basic:
Delphi Noetic Systems, Inc.
P. O. Box 7722
2040 West Main - Suite 211
Rapid City, SD 57709 - USA
Preis: 80 Dollar

BALD
Stuffed by nt



Amiga-Basic im Höhenflug

Nutzen Sie wirklich alles was Amiga-Basic bietet? Wir zeigen, wie Sie Ihre Basic-Programme schneller und effektiver machen können — ohne auf den gewohnten Komfort zu verzichten.

Wer schon einmal einen Blick auf professionelle Software geworfen hat, kennt die Möglichkeiten, die der Amiga bietet: selbstdefinierte Gadgets, Requester, Menüs mit allen Schikanen, um nur einige zu nennen. Amiga-Basic bietet ebenfalls eine Vielzahl von außergewöhnlichen Funktionen, die fast allen bisherigen Basic-Dialekten fehlen. Der Amiga kann jedoch noch einiges mehr. Durch die Verwendung der Betriebssystem-Routinen können Sie seine Fähigkeiten auch in Basic voll ausschöpfen.

Falls Sie Ihr Basic-Handbuch schon einmal genau unter die Lupe genommen haben, wissen Sie, daß mit der »LIBRARY«-Anweisung der Zugriff auf Bibliotheken ermöglicht wird. Leider ist die Beschreibung aber so knapp, daß man ohne zusätzliche Informationen nicht auskommt.

Sie können die Library-Anweisung sowohl für eigene Maschinensprache-Programme verwenden als auch für die zur Verfügung stehenden Betriebssystemroutinen. Mit diesem zweiten Fall werden wir uns im folgenden Kurs beschäftigen.

Systembibliotheken: Was ist das?

Eine Bibliothek ist eine Zusammenfassung von Unterprogrammen, die für den Zu-

Disneyland auf dem Amiga

MOVIESETTER

Die Welt von Disneyland - jetzt auf Ihrem Amiga. Mit **MovieSetter** erstellen Sie Trickfilme - auch wenn Sie kein Profi sind. MovieSetter ermöglicht es, in kürzester Zeit komplexe, minutenlange Animationssequenzen zu erstellen - mit insgesamt nur 1 Mbyte Speicher. Per Mausklick erzeugen Sie aus vorher erstellten Brushes eine fließende Bewegung vor feststehendem oder beweglichem Hintergrund. Plazieren Sie Geräusche innerhalb des Programms und verändern Sie die Tondauer, Tonlage und -stärke; durch Steuern der Tonkanäle können Sie auch Stereoeffekte erzielen.

Zahlreiche vorgefertigte Movie-Clips werden mitgeliefert. Spezielle Animationseffekte erreichen Sie durch Farbdurchlauf oder Playback von bis zu 60 Sequenzen pro Sekunde.
Dieses neue Animationsprogramm ist nicht nur für den Anfänger leicht zu erlernen, es ist auch für den Trickfilmprofi ein vielseitiges Werkzeug.

Bestell-Nr.: 54128, Preis: DM 198,-* (sFr 178,-*/öS 1980,-*)
Deutsche Version in Vorbereitung.
Update DM 49,-* (sFr 49,-*/öS 490,-*)
(lieferbar 1. Quartal 1989).

*Unverbindliche Preisempfehlung

Markt&Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computer-Fachgeschäften oder bei Ihrem Buchhändler.

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0; Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526; Ueberreuter Media Verlagsges.mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.

225-2/811

KURSÜBERSICHT

1. Grundlagen:
Betriebssystem, Startdiskette, Libraries, Bmaps (Seite 29)

2. Fonts:
Schriftarten, Schriftgrößen, Zeichensätze, Arbeiten mit Zeigern und Strukturen (Seite 35)

3. Grafik:
RastPorts, DrawModes, Zeichenbefehle, schnelle Textausgabe (Seite 38)

4. Screens:
Screen-Struktur, Screenbefehle (Seite 40)

5. Windows:
Fenstermanipulation mit Library-Routinen, eigener Maus-Zeiger im Fenster, Intuition-Windows, Borderless- und Backdrop-Windows (Seite 41)

6. Programmbedienung:
Alerts, Requester, Gadgets, Menüs, MessagePorts (Seite 49)

7. Anwendungen:
FileRequester, Hardcopy-Routine (Seite 55)

griff aus verschiedenen Hauptprogrammen geschrieben sind. Das Amiga-Betriebssystem hat eine Vielzahl Routinen, die vom Anwender genutzt werden können. Diese liegen teilweise im ROM, teilweise werden sie auf Diskette mitgeliefert (im »libs«-Ordner Ihrer Workbench finden Sie einige). Normalerweise faßt man der besseren Übersicht wegen Routinen zu einem Thema in derselben Bibliothek zusammen. So liegen alle Routinen für den Zugriff auf die Benutzeroberfläche Intuition in der »Intuition.library«.

Um diese Unterprogramme zu nutzen, muß das Hauptprogramm erfahren, wo das benötigte Unterprogramm zu finden ist — und gegebenenfalls auf welche Weise Werte übergeben werden. Basic verwendet hierfür eine Sprungtabelle, die in Form einer sogenannten ».bmap-Datei« liegen muß.

Eine .bmap-Datei hat folgendes Format:
— Name der Routine, der mit einem Nullbyte CHR$(0) abgeschlossen sein muß.
— Der Abstand von der Einsprungsadresse der Bibliothek (Offset) als 16-Bit-Ganzzahl.
— Die Registerparameter mit einem Nullbyte abgeschlossen.

Registerparameter benötigen Sie nur dann, wenn Ihre selbstgeschriebene Routine zur Parameterübergabe Register verwendet. Andernfalls können Sie den dritten Parameter ignorieren. Für Betriebssystemroutinen existieren vorgefertigte Sprungtabellen, die FD-Dateien im FD1.2-Ordner der Extras-Diskette. In diesen Dateien finden Sie auch alle System-Routinen.

Eigenbau-Startdiskette

Stellen Sie mit Hilfe des »dir«-Befehl im CLI fest, welche ».bmap«-Dateien sich bereits im »libs«-Ordner Ihrer Startdiskette befinden. Sie benötigen für diesen Kurs folgende Dateien:
— diskfont.bmap
— dos.bmap
— graphics.bmap
— exec.bmap
— intuition.bmap

Sollten diese nicht komplett auf Ihrer Startdiskette vorhanden sein, so nehmen Sie die jeweiligen ».FD«-Dateien. Diese können Sie mit dem — ebenfalls auf der Extras-Diskette enthaltenen — Programm »ConvertFD« zu ».bmap«-Dateien konvertieren. Die entstandene ».bmap«-Datei kopieren Sie dann in den »libs«-Ordner Ihrer Basic-Startdiskette.

Zusätzlich sollten Sie zu Beginn des Kurses einige Fonts auf der Startdiskette zur Verfügung haben. Dies sind »garnet« in den Größen 16 und 9 Punkt, »emerald« in 17 und 20 Punkt sowie »topaz 11«. Die Directories Ihrer Startdiskette sollten mindestens die in Tabelle 1 aufgeführten Einträge aufweisen.

Jetzt brauchen wir nur noch eine passende Startupsequence. Hier ist ein Vorschlag mit den minimal benötigten Befehlen, den Sie mit eigenen Ideen ergänzen können:

```
; Startupsequence
zum Basic-Kurs
setmap d
assign libs: df0:libs
run AmigaBasic
```

KURSÜBERSICHT des Basic-Kurses aus dem Sonderheft 1

1. Strukturierte Programmierung:
Programmentwicklung, Top-Down-Design, Struktogramme, optische Gestaltung von Quelltexten, »IF...THEN...ELSE...ENDIF«, »WHILE...WEND« (Seite 85)

2. Unterprogramme
Funktionen, GOSUB-Unterprogramme, echte Unterprogramme, lokale und globale Variablen, Parameterübergabe (Seite 90)

3. Daten und Dateien:
Programmdateien, serielle Dateien, relative Dateien, indexsequentieller Dateizugriff (Seite 92)

4. Grafik
Setzen von Punkten, Screens und Windows, Rechtecke und Ellipsen, Muster, Kopieren und Scrollen von Bildschirmausschnitten (Seite 95)

5. Animation
Bobs und Sprites, Bewegung von Objekten, Kollisionen (Seite 99)

6. Geräusche
Spracherzeugung, Tonleitern, Klangcharakteristik, Wellenformen (Seite 105)

7. Unterbrechungsverarbeitung
Mausereignisse, Menüauswahl, Fehlerbehandlung, Zeitmessung (Seite 106)

8. Externe Funktionen
Einbinden von Maschinenprogrammen, Aufruf von Library-Funktionen, BMAP-Format (Seite 110).

Bevor Sie loslegen ...

... sollten Sie sich schon einige Zeit mit Basic beschäftigt haben. Dieser Kurs setzt einiges an Grundlagen voraus, was bereits in unserem Basic-Kurs für Einsteiger ausführlich behandelt wurde.

Falls Sie also mit diesem Kurs Schwierigkeiten haben, nehmen Sie Sonderheft 1 zur Hand und bearbeiten den Basic-Kurs für Einsteiger. Dieses Heft können Sie jederzeit beim M&T-Verlag, AMIGA-Leserservice, Hans-Pinsel-Str. 2, 8013 Haar, nachbestellen. Eine kurze Übersicht über den Einsteigerkurs finden Sie im nebenstehenden Kasten.

Die Kursdiskette

c (dir)	
Assign	Copy
List	MakeDir
run	SetMap
l (dir)	
Disk-Validator	Port-Handler
Ram-Handler	
devs (dir)	
keymaps (dir)	
d	
printers (dir)	
Epson	(oder ihr Druckertreiber)
parallel.device	printer.device
system-configuration	
s (dir)	
startup-sequence	
fonts (dir)	
garnet (dir)	
16	9
emerald (dir)	
17	20
topaz (dir)	
11	
emerald.font	garnet.font
topaz.font	
libs (dir)	
diskfont.bmap	diskfont.library
dos.bmap	exec.bmap
graphics.bmap	intuition.bmap
AmigaBASIC	

Die Startup-sequence der Kursdiskette:

```
SetMap d
;makedir ram:libs
;copy >NIL: df0:libs/#?.bmap ram:libs
assign libs: df0:libs
;assign libs: ram:libs
run AmigaBasic
```

Tabelle 1. Mindestens diese Einträge sollte Ihre Kursdiskette aufweisen

Falls Sie ein Megabyte oder mehr zur Verfügung haben, können Sie zwischen »setmap« und »assign« noch zwei Zeilen einfügen:

```
makedir ram:libs
copy df0:libs/#?.bmap
ram:libs quiet
```

Ändern Sie jetzt noch die Parameter des Assign-Befehl in

```
assign libs: ram:libs
```

und Sie sparen Diskettenzugriffe in Ihren Programmen, was sich mit kürzeren Laufzeiten angenehm bemerkbar macht. Natürlich können Sie statt der Standard-RAM-Disk auch die resetfeste (»vd0:«) verwenden.

Um von dieser Diskette zu booten, muß jetzt nur noch ein Arbeitsschritt erledigt werden. Sie muß mit dem Befehl »install dfx:« startfähig gemacht werden, wobei Sie für x die Nummer des Laufwerks einsetzen, in dem sich Ihre Diskette befindet. Damit haben Sie alles, was Sie für diesen Kurs benötigen. Um leichter den Überblick zu behalten, sollten Sie noch mit »makedir dfx:kurs« einen eigenen Ordner für den Kurs anlegen, in dem Sie alle Beispielprogramme unterbringen.

Was Sie schon immer über Libraries wissen wollten

In Libraries sind universell verwendbare Unterprogramme enthalten. Gleichgültig, ob Sie in C programmieren, Modula bevorzugen oder vom altgewohnten Basic nicht lassen können, verwenden Sie dieselben Routinen. Eine solche Routine besteht aus einer Ansammlung von Befehlen, denen oft ein oder mehrere Werte (Parameter) übergeben werden müssen. Welche Routinen in diesem Kurs verwendet werden, können Sie aus Tabelle 2 (siehe rechts) entnehmen.

Das Besondere an diesen Unterprogrammen (denn Unterprogramme gibt es schließlich auch in Basic) sind zwei Merkmale:

— Sie liegen in Maschinencode vor und sind deshalb extrem schnell.

— Sie öffnen Ihnen die ganze Leistungsvielfalt des Amiga.

Diese Unterprogramme können — wie in Basic — grundsätzlich zwei Formen annehmen. Sie können als Befehl oder als Funktion programmiert sein. Ein Befehl führt

BIBLIOTHEKSROUTINEN

Routine	Typ(*)	Parameter	Library
ActivateGadget	F	gad&,win&,req&	intuition
ActivateWindow	B	win&	intuition
AddGadget%	F	win&,gad&,pos%	intuition
AddGList%	F	win&,gad&,pos%,n%,req&	intuition
AddPort	B	msgP&	exec
AllocMem&	F	byt&,mode%	exec
AllocRemember&	F	key&,byt&,mode%	intuition
AllocSignal%	F	sigbit%	exec
AskSoftStyle%	F	rastp&	graphics
AutoRequest%	F	**s.text.***	intuition
AvailFonts%	F	buf&,byt&,types%	diskfont
ClearDMRequest	F	win&	intuition
ClearMenuStrip	B	win&	intuition
ClearScreen	B	rastp&	graphics
CloseDevice	B	ioReq&	exec
CloseFont	B	textFont&	graphics
CloseScreen	B	scr&	intuition
CloseWindow	B	win&	intuition
CopyMem	B	src&,dest&,byt&	exec
CopyMemQuick	B	src&,dest&,byt&	exec
DisplayAlert	F	types%,txt&,h%	intuition
DisplayBeep	B	scr&	intuition
DoIO&	F	ioReq&	exec
Draw	B	rastP&,x%,y%	graphics
DrawBorder	B	rastP&,bor&,x%,y%	intuition
DrawEllipse	B	rastP&,mx%,my%,rx%,ry%	graphics
Examine%	F	lock&,buf&	dos
Execute%	F	bef&,ein&,aus&	dos
ExNext%	F	lock&,buf&	dos
FindTask&	F	name&	exec
FreeMem	B	buf&,num&	exec
FreeRemember	B	key&,-1	intuition
FreeSignal	B	sigbit%	exec
FreeSprite	B	num%	graphics
GetMsg&	F	msgP&	exec
GetSprite%	F	ssp&,num%	graphics
InitRequester	B	req&	intuition
Lock&	F	name&,mode%	dos
ModifyProp	B	** s.text. **	intuition
Move	B	rastP&,x%,y%	graphics
MoveScreen	B	scr&,dx%,dy%	intuition
MoveSprite	B	ViewP&,ssp&,x%,y%	graphics
MoveWindow	B	win&,dx%,dy%	intuition
OpenDevice&	F	name&,unit%,ioReq&,flg	exec
OpenDiskFont&	F	textAttr&	diskfont
OpenFont&	F	textAttr&	graphics
OpenScreen&	F	newscr&	intuition
OpenWindow&	F	newwin&	intuition
ParentDir&	F	lock&	dos
PolyDraw	B	rastP&,num%,dat&	graphics
PrintIText	B	rastP&,it&,x%,y%	intuition
RectFill	B	rastP&,l%,o%,r%,u%	graphics
RefreshGadgets	B	gad&,win&,req&	intuition
RemFont%	F	textFont	graphics
RemoveGadget%	F	win&,gad&	intuition
RemoveGList	F	win&,gad&,num%	intuition
RemPort	B	msgP&	exec
ReplyMsg	B	msg&	exec
Request%	F	req&,win&	intuition
ScreenToBack	B	scr&	intuition
ScreenToFront	B	scr&	intuition
SetAPen	B	rastP&,num%	graphics
SetBPen	B	rastP&,num%	graphics
SetDMRequest	F	win&,req&	intuition
SetDrMd	B	rastP&,draw%	graphics
SetFont%	F	rastP&,textFont&	graphics
SetMenuStrip	B	win&,men&	intuition
SetPointer	B	win&,pnt&,h%,b%,x%,y%	intuition
SetRast	B	rastP&,pen%	graphics
SetRGB4	B	viewP&,col%,r%,g%,b%	graphics
SetSoftStyle%	F	rastP&,style%,able%	graphics
SetWindowTitles	B	win&,winT&,scrT&	intuition
ShowTitle	B	scr&,show%	intuition
SizeWindow	B	win&,dx%,dy%	intuition
Text	F	rastP&,txt&,len%	graphics
Unlock	B	lock&	dos
WindowLimits	F	win&,xu%,yu%,xo%,yo%	intuition
WindowToBack	B	win&	intuition
WindowToFront	B	win&	intuition
WritePixel	B	rastP&,x%,y%	graphics

Tabelle 2.

* Typ F = Funktion
B = Befehl

AMIGA SONDERHEFT

PROGRAMM-SERVICE

Direkt bestellen statt abtippen!

Die aktuelle Diskette zum Heft:

Amiga Sonderheft 3: Basic, Spiele

Broker: Erleben Sie die Faszination der Börse hautnah. Diese Simulation für 2 bis 4 Spieler ist einzigartig. Der Autor setzt seine fundierten Kenntnisse in spannendes Spielgeschehen um.

Ping-Pong: Dieses Sportspiel bringt Wettkampfstimmung ins Wohnzimmer. Dreidimensionale Darstellung, realistische Soundeffekte und rasante Ballwechsel führen zu lang anhaltendem Spielspaß.

Anpfiff: Als Manager in der Fußball-Bundesliga führen Sie Ihr Lieblingsteam durch die Saison. Zusätzliche Spiele im UEFA-Cup verhelfen Ihrer Mannschaft zu Ruhm und Ihnen zu vielen Manager-Punkten.

Basic-Routinen: Die Basic-Kurse im Sonderheft 3 bieten zahlreiche, hilfreiche Routinen. Alle dort vorgestellten Programme finden Sie auch auf dieser Programmservice-Diskette.

Weiterhin befinden sich auf der Diskette alle Programme, die im Inhaltsverzeichnis des Amiga-Sonderhefts 3 mit einem Diskettensymbol gekennzeichnet sind.

3½"-Diskette für Amiga
Bestell-Nr. 45903

DM 29,90* *(sFr 24,90*/öS 299,–*)*
* Unverbindliche Preisempfehlung

Bewegte Grafik per Farbdurchlauf – Diashow für Anspruchsvolle

Es ist vollbracht! Die besten Cycling-Grafiken können nun alle grafikbegeisterten Leser hautnah erleben. Die schönsten Bilder des **»Color-Cycle«-Wettbewerbs,** das im Amiga-Magazin (Ausgabe 3/88 Seite 142) gestartet wurde, präsentieren wir Ihnen auf zwei randvoll bespielten Disketten.

Die Bilder können entweder mit dem enthaltenen Diashow-Programm angesehen oder mit jedem gängigen IFF-Malprogramm (zum Beispiel Deluxe Paint II) geladen werden.

Lassen Sie sich die faszinierenden Computer-Bilder nicht entgehen.

Zwei Disketten für Amiga.
Bestell-Nr. 49901

DM 29,90* *(sFr 24,90*/öS 299,–*)*
* Unverbindliche Preisempfehlung

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Weitere Angebote auf der Rückseite!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0; Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 677526, Ueberreuter Media Verlagsges. mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0

AMIGA PROGRAMMSERVICE

Sie suchen hilfreiche Utilities und professionelle Anwendungen für Ihren Computer? Sie wünschen sich gute Software zu vernünftigen Preisen? Hier finden Sie beides!
Unser stetig wachsendes Sortiment enthält interessante Listing-Software für alle gängigen Computertypen. Jeden Monat erweitert sich unser aktuelles Angebot um eine weitere interessante Programmsammlung für jeweils einen Computertyp.
Bei Fragen zu Bestellung und Versand der Programmservice-Disketten wählen Sie bitte:
Telefon (089) 4613-232.

Bestellungen bitte *nur gegen Vorauskasse* an: Markt&Technik Verlag AG, Unternehmensbereich Buchverlag, Hans-Pinsel-Straße 2, D-8013 Haar, Telefon (089) 4613-0.
Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656.
Österreich: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 5871393-0, Microcomput-ique, E. Schiller, Fasangasse 24, A-1030 Wien, Telefon (0222) 785661; Bücherzentrum Meidling, Schönbrunner Straße 261, A-1120 Wien, Telefon (0222) 833196. Ueberreuter Media, Verlagsges. mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 481543-0.
Bestellungen aus anderen Ländern bitte nur schriftlich an: Markt&Technik Verlag AG, Abt. Buchvertrieb, Hans-Pinsel-Straße 2, D-8013 Haar. Nur gegen Bezahlung der Rechnung im voraus.

Bitte verwenden Sie für Ihre Bestellung und Überweisung die beigeheftete Postgiro-Zahlkarte, oder senden Sie uns einen Verrechnungs-Scheck mit Ihrer Bestellung. Sie erleichtern uns die Auftragsabwicklung, und dafür berechnen wir Ihnen keine Versandkosten.

Amiga Sonderheft 2: Grafik, Anwendung

Object-Editor: Animierte Figuren, beispielsweise für eigene Spiele, entwickeln Sie mit diesem Editor auf komfortable Weise. Sogar mit Deluxe Paint erstellte Pinsel lassen sich einlesen.
Haushaltsbuch: Mit diesem hervorragenden Anwendungsprogramm verwalten Sie alle Einnahmen und Ausgaben auf übersichtliche Weise. Eine Monats- oder Jahresstatistik zeigt, in welchen Bereichen Sie zukünftig sparen können. Jetzt haben Sie Ihre Finanzen im Griff.
Keyboard-Master: Lernen Sie im Zehn-Finger-System zu tippen. Mit diesem didaktisch ausgereiften Programm ist dies kein Problem. Für Programmierer sind sogar Spezial-Lektionen mit wichtigen Sonderzeichen vorhanden.
FastLoadCopy: Dieses Tool bringt den DIR-Befehl auf Trab. Nach der »Operation« wird das Inhaltsverzeichnis einer Diskette im D-Zug-Tempo eingelesen. Zusätzlich kopiert das Programm Disketten und versieht diese mit dem schnellen Directory. Weiterhin befinden sich auf der Diskette alle Programme, die im Inhaltsverzeichnis des Amiga-Sonderhefts 2 mit einem Disketten-symbol gekennzeichnet sind.
3½"-Diskette für Amiga

Bestell-Nr. 45802 **DM 29,90*** sFr 24,90*/öS 299,–*

Amiga Sonderheft 1: Bauen Sie Ihr eigenes Tonstudio

Digisoft Plus: Die Software zu unserem Selbstbau-Digitizer, die den Amiga zum digitalen Sound-Studio macht. Das Amiga-Tonstudio arbeitet mit vier getrennten Tonspuren und besitzt neben der Digitalisierung von Klängen viele weitere Features. So können einzelne Spuren oder Teile daraus beliebig gemischt, geschnitten oder verbunden werden. Die grafische Ausgabe der digitalisierten Sounds macht die Bearbeitung zum Kinderspiel.
Suremosch: Ein Strategiespiel der neuen Dimension: Begeben Sie sich in eine ferne, fremde Welt und befreien Sie die Bevölkerung von bösen Wesen, die den Zeitfluß verlangsamen.
Dcopy: ist ein schnelles Kopierprogramm der Extraklasse. Neben hoher Geschwindigkeit bietet es die Möglichkeit, bis zu drei Kopien in einem Arbeitsgang anzufertigen.
Biorhythmus: Neben einer sehr schönen grafischen Darstellung der Kurven von Körper, Geist und Seele erlaubt das komfortable Programm auch die Ausgabe einer Jahresstatistik. Sie erfahren alles über Ihre »guten« und »kritischen« Tage.
3½"-Diskette für Amiga

Bestell-Nr. 45801 **DM 29,90*** sFr 24,90*/öS 299,–*

Die Wiederbelebung für die C64-Peripherie

Viele Amiga-Besitzer haben noch einen C64 mit Peripheriegeräten zu Hause stehen. Mit ein bißchen Hard- und Software können Sie diese zu neuem Leben erwecken und Ihre Daten so weiterbenutzen. Dabei ist die Bedienung wirklich einfach.
Der fertig aufgebaute IEC-Handler erlaubt es, alle C64-Geräte wie die Floppy 1541 oder 1571, Commodore-MPS-Drucker und natürlich auch den C64 (zur Datenübertragung) am Amiga zu betreiben.
Das Gesamtpaket besteht aus der fertig aufgebauten Platine mit Verbindungskabel, der Treibersoftware auf 3½"-Diskette sowie einer entsprechenden Dokumentation.

Bestell-Nr. 39101 **DM 79,–*** sFr 71,–*/öS 790,–*

Bestellungen bitte nur gegen Vorauskasse bei:
Markt&Technik Verlag AG
– Buchverlag –
IEC-Handler
Hans-Pinsel-Straße 2, 8013 Haar bei München

Amiga 3/88 Bildschirmfüllende Boot-Bilder mit allen Extras

BootGirl: Fantastische Bilder sofort nach dem Reset. Bis zu 32 Farben mit Color-Cycling. Die Bilder können auch bildschirmfüllend ohne Rand sein. Ein absolutes Muß für jeden Amiga-Besitzer.
CassCover: Selbstgedruckte Kassettenhüllen geben Ihnen den richtigen Überblick. Einfache Bedienung macht das Eingeben und Ausdrucken zur wahren Freude.
Command: Das Programm ermöglicht die Steuerung des Aztec-C-Compilers mit der Maus. Keine langen Eingaben per Tastatur, sondern ein einziger Mausklick startet nun die Übersetzung.
VideoText: Ein unentbehrliches Werkzeug für alle Video-Fans, die ihren eigenen Vorspann mit dem Amiga generieren wollen. Laufbänder, verschiedene Schriften und IFF-Bilder sind nur einige Stichpunkte, die das Programm so interessant machen.
3½"-Diskette für Amiga

Bestell-Nr. 48803 **DM 29,90*** sFr 24,90*/öS 299,–*

Amiga 12/87 Super-Kopierprogramm mit viel Komfort

DCopy: Unser Programm des Monats, ein Kopierprogramm, das alles bietet, was man sich nur wünschen kann. Einige Fähigkeiten: Bis zu vier Laufwerke werden verwendet, Mehrfachkopien, abschaltbares Verify und vieles mehr.
SpeedHc: Eine sehr schnelle Hardcopyroutine für Schwarzweißausdrucke mit höchster Qualität. Leicht an andere Drucker anzupassen.
Sternenhimmel: Ein unentbehrliches Werkzeug für alle Himmelsbeobachter. Das Programm zeigt alle Sterne und Planeten von jedem beliebigen Punkt der nördlichen Hemisphäre.
Checkie42: Der Checksummer für alle Programmiersprachen von Assembler über Basic bis zu C. Ab dieser Ausgabe finden Sie bei jedem Listing die Prüfziffern.
Joy: Ein sehr kurzes und schnelles C-Programm zur Abfrage des Joysticks. Es ist leicht in eigene Programme einzubinden.
Amiga-Shell: Ein C-Programm, das Komfort ins CLI bringt. Editieren der Befehlszeile, Funktionstastenbelegung und Aliasnamen sind nur einige Fähigkeiten dieses fantastischen Programms.
3½"-Diskette für Amiga

Bestell-Nr. 48705 **DM 29,90*** sFr 24,90*/öS 299,–*

Amiga 10/87 Super-Malprogramme für alle Amiga-Computer

Rainbow-Drawer: Dieses Programm des Monats bietet leistungsfähige Befehle und Funktionen, wie sie von professionellen Programmen bekannt sind: bis zu 32 Farben, alle Auflösungen, viele Befehle zum Zeichnen sowie FILL mit Mustern, BOW und anderem.
Turtle: Mit dieser Befehlserweiterung verfügen Sie über die Grafikbefehle, die bei Logo bekannt und beliebt sind.
Fractals: Dreidimensionale, realistische Gebirge mit Schattierung erzeugt dieses Programm.
Clouds: Genauso wirklichkeitsnah wie die Gebirge, aber noch erstaunlicher, sind die Wolken, die Sie mit Clouds generieren.
Apfelmännchen: Hiermit erzeugen Sie schöne Grafiken aus der beliebten Mandelbrot-Ebene.
Kudiplo: Ein gutes, unverzichtbares Werkzeug für die Kurvendiskussion stellt »Kudiplo« dar.
Senso: Testen Sie mit dieser Computer-Adaption des bekannten Spiels Ihr Gedächtnis!
Division: Bis zu 32000 Nachkommastellen können durch dieses Programm berechnet werden.
Alert: Alarme, zum Beispiel die bekannten Guru-Meditations, können Sie nun selbst programmieren. Das Programm ist in erster Linie für C-Programmierer aufschlußreich.
Border: Lassen Sie den Fensterrahmen des CLI-Fensters einfach verschwinden!
SCD: Mit diesem Utility können Sie den Pfadnamen in der Titelleiste des Fensters anzeigen.
3½"-Diskette für Amiga

Bestell-Nr. 48703 **DM 29,90*** sFr 24,90*/öS 299,–*

* Unverbindliche Preisempfehlung

Übrigens: Mit den Gutscheinen aus dem »Super-Software-Scheckheft« für DM 149,– können Sie sechs Software-Disketten Ihrer Wahl aus dem Programm-Service-Angebot der Zeitschriften

PC Magazin	Happy-Computer-Sonderheft	Computer persönlich
PC Magazin Plus	Amiga-Magazin	64'er-Magazin
Happy-Computer	Amiga-Sonderheft	64'er-Sonderheft

bestellen – egal, ob diese DM 29,90 oder DM 34,90 kosten. Das Scheckheft können Sie per Verrechnungsscheck oder mit der eingehefteten Zahlkarte direkt beim Verlag bestellen.
Kennwort: Software-Scheckheft, Bestell-Nr. 39100.

eine oder mehrere Aktionen für das Hauptprogramm durch und übergibt die Kontrolle dann wieder an das aufrufende Programm. Eine Funktion dagegen gibt außerdem noch einen Wert zurück. Dieser muß vom aufrufenden Programm mittels einer Wertzuweisung ausgelesen werden (beispielsweise»mask%=AskSoftStyle %(WINDOW(8))« in Listing 1).

Dagegen werden Befehle wie gewohnt mit »CALL« aufgerufen. Der letzte Unterschied bezieht sich auf die Deklaration. Befehle sind für Basic bekannt, sobald die Library geöffnet ist, müssen also nicht deklariert werden. Dagegen müssen Sie für jede Funktion eine Deklaration an den Anfang des Programms stellen:

```
DECLARE FUNCTION
FunktionsName LIBRARY
```

Manche Funktionen geben nur eine Fehlermeldung zurück. Diese können auch als Befehle benutzt werden. Sie dürfen dann allerdings **nicht** deklariert sein. Haben Sie eine Funktion einmal deklariert, so müssen Sie auch jedesmal den Rückgabewert auslesen.

Nachdem Sie alle aus einer Bibliothek benötigten Funktionen deklariert haben, übergeben Sie mit »LIBRARY "name.library"« noch den Namen der Bibliothek, in der alle Routinen zu finden sind. AmigaBasic sucht dann in der entsprechenden ».bmap«-Datei nach dem Offset der deklarierten Funktionen.

```
 1 B90 'Dieses Programm zeigt die Verwendung
 2 xe  'verschiedener Schrifttypen
 3 pv  'von BASIC aus
 4 kG  DECLARE FUNCTION AskSoftStyle% LIBRARY
 5 ff  DECLARE FUNCTION SetSoftStyle% LIBRARY
 6 06  LIBRARY "graphics.library"
 7 mc  DIM style$(3),neu%(3)
 8 Ib  FOR i=0 TO 3
 9 I52   READ style$(i),neu%(i)
10 FKO NEXT
11 GM  DATA Normal,0,Unterstrichen,1,Fett,2,Kursiv,4
12 qv  mask%=AskSoftStyle%(WINDOW(8))
13 Ng  FOR i=0 TO 3
14 3K2   nmask%=SetSoftStyle%(WINDOW(8),neu%(i),mask%)
15 G1    PRINT style$(i)
16 cF    PRINT
17 MRO NEXT
18 wI  nmask%=SetSoftStyle%(WINDOW(8),0,mask%)
(C) 1988 M&T
```

Listing 1. Hier wird der Gebrauch verschiedener Schrifttypen demonstriert

DiskFonts

emerald 20

Fonts

Sicher wollten Sie schon einmal einen Schriftzug unterstrichen darstellen und haben dafür Sonderzeichen wie das Gleichheitszeichen (»=«) oder den Underscore (»_«) verwendet. Aber kennen Sie die Schriftarten, die der Amiga eingebaut hat? Jeder Text kann unterstrichen, kursiv, fett oder normal dargestellt werden. Sie brauchen dafür lediglich eine Bibliothek, die »graphics.library«. Die Funktionen zum Umschalten der Schriftstile heißen »AskSoftStyle« und »SetSoftStyle«.

Bevor Sie eine dieser Funktionen anwenden können, deklarieren Sie beide und öffnen die »graphics.library«. Verwenden Sie dafür die Zeilen

```
DECLARE FUNCTION
 AskSoftStyle& LIBRARY
DECLARE FUNCTION
 SetSoftStyle& LIBRARY
LIBRARY "graphics.
 library"
```

SetSoftStyle erneuert die Werte der sogenannten »Textattribute«. Diese legen fest, ob die aktuelle Schriftausgabe normal, unterstrichen, fettgedruckt, kursiv oder breitgedruckt erfolgt. Dazu wird folgende Tabelle verwendet:

Bit	Wert	Bedeutung
0	1	unterstrichen
1	2	fett
2	4	kursiv
3	8	breit

Ein Wert von 0 (kein Bit gesetzt) bedeutet normale Schrift, 3 (Bit 0 und Bit 1 gesetzt) heißt fettgedruckt und unterstrichen. Auf diese Art lassen sich alle Attribute beliebig kombinieren.

Wenn Sie die derzeitige Einstellung verändern wollen, verwenden Sie folgende Zeile:

```
neueMaske=SetSoftStyle
(RastPort,neu,maske)
```

Dabei ist »RastPort« die Adresse der RastPort-Struktur des Ausgabefensters. Eine RastPort-Struktur wird von Intuition angelegt, um die Grafikdaten eines Windows oder Screens zu verwalten. Sie erhalten einen Zeiger auf diese Struktur mit der Basic-Funktion »WINDOW(8)«. Mit »neu« übergeben Sie den Wert der von Ihnen gewünschten Schriftart. »maske« legt fest, welche Bits verändert werden dürfen. SetSoftStyle stellt eine logische Und-Verknüpfung zwischen »neu« und »maske« her und setzt den dabei entstehenden Wert als neue Schriftart.

Wenn Sie als Maske eine Null übergeben, läßt SetSoftStyle keine Veränderungen zu. Bei »-1« wird jede beliebige Veränderung akzeptiert. Beim ersten Zugriff auf die Textattribute empiehlt es sich, mit

```
maske=AskSoftStyle
(RastPort)
```

die im aktuellen RastPort festgelegten Einstellungen zu holen.

AskSoftStyle holt den Wert für »maske« direkt aus dem RastPort. Sie können diesen Wert nur mit SetSoftStyle verändern (indem Sie einen neuen Wert für »mask« angeben).

In Listing 1 sehen Sie ein kurzes Beispielprogramm. Zuerst werden die beiden erwähnten Funktionen als Integerfunktionen deklariert und die "graphics.library" geöffnet. Dann werden zwei Felder »style$« und »neu%« dimensioniert, in denen die Namen der Schriftarten beziehungsweise die entsprechenden Parameter eingelesen werden. Dann werden mit der Funktion AskSoftStyle die momentanen Einstellungen geholt. In der folgenden Schleife werden die Stilnamen im zugehörigen Stil ausgegeben und schließlich wieder Normalschrift eingestellt.

Wenn Sie genau beobachtet haben, dann ist Ihnen vielleicht aufgefallen, daß ein Wert ungenutzt bleibt. Der Rückgabewert von SetSoftStyle, »nmask%« enthält den aktualisierten Wert von »maske«. Da dieser aber nicht verändert wird, muß er auch nicht ausgewertet werden.

Schriften — die Erste

Sicher haben Sie schon einmal in Preferences die Möglichkeit gesehen, zwischen 60 und 80 Zeichen pro Zeile umzuschalten. Der Sinn dieser Einstellung liegt darin, die Schrift bei Betrieb an einem Fernsehgerät lesbar zu machen. Umgeschaltet wird dabei lediglich zwischen den beiden internen (ROM-residenten) Zeichensätzen des Amiga, von denen einer eine Höhe von acht Bildpunkten aufweist, der andere eine von neun.

Mit AmigaBasic haben Sie im Gegensatz zu Notepad und den meisten Textverarbeitungsprogrammen keine Möglichkeit zum Wechseln des Zeichensatzes. Aber auch hier helfen uns die Betriebssystem-Routinen weiter.

Listing 2 ist eine kleine Erweiterung unseres ersten Programms. Die verschiedenen Schriftarten werden jetzt in ein eigenes Fenster ausgegeben. Außerdem wird vor der Ausgabe der Zeichensatz verändert (Zur Unterscheidung: In Listing 1 wurde nicht der Zeichensatz, sondern dessen Darstellungsform geändert). Diese Aufgabe übernimmt der Befehl »sFont 9«. Dieser verzweigt in ein Unterprogramm, das den aktuellen Zeichensatz auf die in y% übergebene Größe umschaltet.

Sehen wir uns diese Routine etwas genauer an: Die eigentliche Arbeit wird von der Funktion SetFont erledigt. Diese setzt im RastPort (»WINDOW(8)«) einen Zeichensatz. Hierfür muß der Zeiger auf die Textfont-Struktur des neuen Zeichensatzes übergeben werden (»font&«). Eine Textfont-Struktur ist eine Liste aller wichtigen Parameter des Zeichensatzes. Sie haben von Basic aus keinen direkten Zugriff auf diese Struktur, daher wird der Zeiger mit »OpenFont« ermittelt.

OpenFont benötigt einen Parameter, einen Zeiger auf eine

```
 1 B90 'Dieses Programm zeigt die Verwendung
 2 40  'der residenten Zeichensätze
 3 kT  '"topaz9" und "topaz8"
 4 kG  DECLARE FUNCTION AskSoftStyle% LIBRARY
 5 ff  DECLARE FUNCTION SetSoftStyle% LIBRARY
 6 DZ  DECLARE FUNCTION OpenFont& LIBRARY
 7 ZL  DECLARE FUNCTION SetFont LIBRARY
 8 28  LIBRARY "graphics.library"
 9 PF  WINDOW 3,"Topaz9",(100,30)-(250,93),0
10 pf  DIM style$(3),neu%(3)
11 Wa  DIM SHARED attr%(3)
12 Mf  FOR i=0 TO 3
13 M92   READ style$(i),neu%(i)
14 J00 NEXT
15 KQ  DATA Normal,0,Unterstrichen,1,Fett,2,Kursiv,4
16 Vy  sFont 9
17 v0  mask%=AskSoftStyle%(WINDOW(8))
18 S1  FOR i=0 TO 3
19 8P2   nmask%=SetSoftStyle%(WINDOW(8),neu%(i),mask%)
20 DP    PRINT style$(i);
21 kG    IF i<3 THEN PRINT:PRINT
22 RW0 NEXT
23 1N  nmask%=SetSoftStyle%(WINDOW(8),0,mask%)
24 np  WHILE INKEY$="":SLEEP:WEND
25 b3  sFont 8
26 h0  WINDOW CLOSE 3
27 B6  END
28 w3  SUB sFont (y%) STATIC
29 LO2   IF font&<>0 THEN CALL CloseFont(font&)
30 z8    top$="topaz.font"+CHR$(0)
31 ii    POKEL(VARPTR(attr%(0))),SADD(top$)
32 19    attr%(2)=y%
33 5M    font&=OpenFont&(VARPTR(attr%(0)))
34 Kn    e=SetFont(WINDOW(8),font&)
35 bd0 END SUB
(C) 1988 M&T
```

Listing 2. Mit »topaz8« und »topaz9« wird die Schriftgröße verändert

Liste. Diese Liste enthält drei Elemente: Das erste Element ist ein String, der den Namen des neuen Zeichensatzes enthält. Genaugenommen ist dies ein Zeiger auf das erste Zeichen des Strings; daher kommt die Konstruktion mit »POKEL (VARPTR(attr%(0))), SADD(top$)«. Das Ende des Strings wird von der Betriebssystemroutine an einem Nullbyte erkannt. Zu diesem Zweck wird der String »top$« um ein »CHR$(0)« verlängert.

Das zweite Element der Liste ist die Höhe in Punkten (hier mit y% übergeben).

Das dritte ist der Schriftstil, der in dieser Unterroutine nicht verwendet wird, weil er im Hauptprogramm gesetzt wird.

Die Liste selbst legen wir in einem Feld (»attr%()«) ab. OpenFont erhält über die Funktion »VARPTR« die Anfangsadresse dieses Feldes.

Wenn Sie einen Zeichensatz nicht mehr benötigen, sollten Sie Ihn mit dem »CloseFont«-Befehl entfernen. Es gehört zum »guten Ton« bei der Programmierung auf Betriebssystemebene, nicht mehr benötigte Speicherbereiche freizugeben. Damit sparen Sie nicht nur Speicher, Sie vermeiden auch fehlerträchtige Doppelzugriffe. Durch das Multitasking-Konzept des Amiga ist es gut möglich, daß Sie zwar Bescheid wissen, wieviel Speicher und sonstige Ressourcen (dazu zählen beispielsweise auch Devices) Ihr Programm belegt. Dennoch können die nebenbei laufenden Programme soviel Speicher anfordern, daß für den letzten Task nicht mehr genügend übrig bleibt.

Der Amiga kennt nur zwei ROM-residente Zeichensätze, »topaz 8« und »topaz 9«. Übergeben Sie daher keine anderen Schriftgrößen, diese würden vom System nicht gefunden.

Zeiger und Strukturen

Basic-Programmierer glauben manchmal, Geschichten aus böhmischen Dörfern zu hören, wenn von Zeigern und Strukturen die Rede ist. Beides sind Datentypen, die in modernen Programmiersprachen wie Modula oder C sehr verbreitet sind. Nachdem das Betriebssystem des Amiga in C geschrieben ist, folgen viele Betriebssystemroutinen den Aufrufkonventionen dieser Sprache.

Dies ist allerdings noch lange kein Grund, als Basic-Programmierer die Flinte ins Korn zu werfen und sich einen teuren C-Compiler zu kaufen. Man kann auch in Basic fast alles »nachbauen«, was in C Verwendung findet.

Zeiger sind Adressen von Variablen. Dabei kann die Variable ein beliebiges Format haben, der Zeiger »zeigt« immer auf die erste Adresse der Variablen. In Basic stehen zwei Funktionen zur Verfügung, »VARPTR« und »SADD«. Mit SADD kann die Adresse eines Strings ermittelt werden, während VARPTR für alle anderen Variablen angewendet wird. Das Format dieser beiden Funktionen ist

```
v=VARPTR(Variable)
```

und

```
s=SADD(String)
```

Der Rückgabewert dieser Funktionen ist die Adresse der jeweiligen Parameter im Speicher, also nichts anderes als ein Zeiger auf eine Variable beziehungsweise auf einen String.

Strukturen sind Zusammenfassungen aus verschiedenen Datentypen zu einem »Thema«. Beispielsweise sind in der Window-Struktur alle Informationen zum aktuellen Fenster abgelegt. Diese Informationen haben aber unterschiedliches Format. So werden die Lage und die Maße mit Zwei-Byte-Zahlen dargestellt, sogenannte »Flags« beanspruchen nur ein Bit, um einen Zustand zu beschreiben, und Adressen sind Langwörter (also Vier-Byte-Zahlen). Um in Basic auf die richtige Speicherstelle zuzugreifen, brauchen wir neben der Adresse der Struktur den Offset — das ist der Abstand von der Startadresse. Diesen Abstand ermitteln wir über den Aufbau einer Struktur.

Der Offset wird zu der Adresse der Struktur addiert, um die Speicherstelle zu ermitteln, an die der jeweilige Wert mittels eines »POKE«-Befehls gebracht werden muß.

Wie viele Schriften braucht der Mensch?

In unserem letzten Beispiel hatten wir von zwei ROM-residenten Zeichensätzen gesprochen. Für einen Computer wie den Amiga wäre das in der Tat etwas wenig. Der Amiga ist so konzipiert, daß er grundsätzlich beliebige Schriften darstellen kann. Um jedoch das System nicht unnötig zu belasten (unter normalen Umständen wird man immer nur einen Zeichensatz gleichzeitig anwenden), existieren alle anderen Schriften nur auf Diskette.

Sehen Sie sich auf Ihrer Original-Workbench den »Fonts«-Ordner an. Sie finden neben einigen Files mit der Endung ».font« auch die gleichnamigen Directories und in diesen jeweils einige Files, die nur durch eine Zahl gekennzeichnet sind. Diese Zahlen bezeichnen die Höhe der Zeichensätze. Die meisten sind in unterschiedlichen Höhen verfügbar.

Die Verwaltung von nicht speicherresidenten Zeichensätzen in einem Multitasking-System, in dem auch noch jeder Task mehrere Windows betreiben kann, ist eine reichlich komplizierte Angelegenheit. Schließlich können verschiedene Tasks auch auf denselben Zeichensatz zugreifen.

Bis zum letzten Task

Hier könnte die gegenseitige Beeinflussung verschiedener Tasks leicht zu Katastrophen führen, hätten die Entwickler nicht sehr umsichtig gearbeitet. Denn der »RemFont«-Befehl streicht einen Zeichensatz aus der System-Font-Liste. Wenn aber der Zeichensatz noch von einem anderen Task benutzt wird, wird nur vermerkt, daß der Zeichensatz entfernt werden sollte. Wenn der letzte Task, der auf diesen Zeichensatz zugreift, den Close Font-Befehl aufruft, erkennt das System dies und entfernt jetzt den Zeichensatz.

Listing 3 zeigt anhand der »AvailFonts«-Funktion, wie man die verfügbaren Zeichensätze ermittelt. Aus diesem Programm soll hier nur die AvailFonts-Funktion erklärt werden; »AllocMem« und »FreeMem« werden später ausführlicher behandelt.

Die AvailFonts-Funktion hat die Syntax

```
e%=AvailFonts(buf&, len%
, types)
```

»buf&« ist ein Zeiger auf einen freien Speicherplatz, einen sogenannten Puffer, für die System-Font-Liste, »len%« ist die Länge des Puffers und »types« gibt an, ob der Speicher, die Diskette oder beides durchsucht werden sollen (1 = AFF__DISK; 2 = AFF__ME-

```
 1 wY0 'Dieses Programm zeigt, wie man fest-
 2 08  'stellt, welche Zeichensätze verfuegbar
 3 ht  'sind und listet diese auf
 4 g5  OPTION BASE 1
 5 6h  DECLARE FUNCTION AvailFonts% LIBRARY
 6 VM  DECLARE FUNCTION AllocMem& LIBRARY
 7 ow  LIBRARY "diskfont.library"
 8 28  LIBRARY "graphics.library"
 9 9m  LIBRARY "exec.library"
10 cG  CheckFonts:
11 3p  buf&=AllocMem&(2000,65537&)
12 a1  b&=buf&
13 Yj  e%=AvailFonts%(b&,2000,3)
14 8k  anz%=PEEKW(b&):b&=b&+2
15 2F  PRINT USING"# # # Zeichensätze gefunden";anz%
16 4X  DIM type%(anz%),nam$(anz%),ySize%(anz%),Style%(anz%),Flags%(
       anz%)
17 du  FOR i%=1 TO anz%
18 FK2   type%(i%)=PEEKW(b&)
19 sm    na&=PEEKL(b&+2)
20 QQ    rn:
21 QL    n%=PEEK(na&)
22 ln    IF n%<>0 THEN
23 LW4     nam$(i%)=nam$(i%)+CHR$(n%)
24 nE      na&=na&+1
25 V4      GOTO rn
26 MF2   END IF
27 ro    ySize%(i%)=PEEKW(b&+6)
28 NN    Style%(i%)=PEEK(b&+8)
29 5J    Flags%(i%)=PEEK(b&+9)
30 tu    b&=b&+10
31 af0 NEXT
32 sV  PRINT
33 bu  PRINT "Type        Name                ySize   Style  Flags"
34 zF  PRINT STRING$(55,"_")
35 vY  PRINT
36 wD  FOR i%=1 TO anz%
37 Mq2   IF type%(i%)=1 THEN PRINT "AF_MEMORY : ";:ELSE:PRINT "AF_D
         ISK    : ";
38 KK    PRINT USING"\                \";nam$(i%);
39 1Q    PRINT USING" # # #  ]";ySize%(i%),Style%(i%),Flags%(i%)
40 jo0 NEXT
41 46  WHILE INKEY$="":SLEEP:WEND
42 Zk  CALL FreeMem(buf&,2000)
(C) 1988 M&T
```

Listing 3. Mit »AvailFonts« stellen Sie fest, welche Zeichensätze verfügbar sind

MORY; 3 = beides). Ist nur »AFF_DISK« gesetzt, so wird nur auf der Diskette gesucht, bei »AFF_MEMORY« nur im ROM. Wenn genügend Speicher vorhanden ist, wird in e% eine Null für fehlerlos abgelegt, ansonsten ist e% die Anzahl an Bytes, die zusätzlich zu »len%« benötigt werden, um die System-Font-Liste komplett aufzubauen.

Was legt nun die AvailFonts-Funktion im Puffer (buf&) ab? Zuerst kommt hier die »AvailFontsHeader«-Struktur, in der als einziger Eintrag die Anzahl der folgenden Einträge steht (dieser Wert ist 16 Bit breit, kann also mit »PEEKW« ausgelesen werden). Dann folgt genau diese Anzahl an AvailFonts-Strukturen. In jeder dieser Strukturen steht der Typ des Zeichensatzes (»AFF_DISK« oder »AFF_MEMORY« in Wortbreite) und eine »TextAttr«-Struktur, die man benötigt, um den Zeichensatz zu öffnen. Natürlich kann man die TextAttr-Struktur selbst aufbauen, wie wir in Listing 1 gesehen haben. Der Vorteil von AvailFonts liegt darin, daß festgestellt werden kann, ob die benötigten Datenstrukturen vorhanden sind, bevor darauf zugegriffen wird. Wenn Sie dieses Vorgehen bei Systemzugriffen konsequent anwenden, können Sie so manchem Guru den Weg versperren. Um mit »OpenDiskFont« einen Zeichensatz zu laden, müssen im »Fonts«-Ordner eine Datei mit der Endung ».font« vorhanden sein und im zugehörigen Unterverzeichnis eine Definitions-Datei für die richtige Größe (zum Beispiel »20« im emerald-Ordner und »emerald.font« im Fonts-Ordner für »emerald 20«).

Beliebige Schriften mit PRINT

Nachdem das Programm aus Listing 3 gelaufen ist, sind im Ausgabefenster alle Zeichensätze aufgelistet, die uns zur Verfügung stehen. Nun wollen wir uns daran machen, diese zu verwenden. Listing 4 gibt ein einfaches Beipiel dafür. Da wir wissen, daß »emerald 20« vollständig vorhanden ist, können wir die TextAttr-Struktur direkt aufbauen; sollten Sie sich nicht sicher sein, dann sehen Sie mit Hilfe des Programms aus Listing 3 noch einmal nach und kopieren Sie — falls notwendig — die entsprechenden Dateien auf Ihre Systemdiskette. Da es sich um einen externen Zeichensatz handelt, wird er mit der »OpenDiskFont«-Funktion geöffnet und ins RAM geladen. Am Ende des Programms wird mit »CloseFont« und »RemFont« der neue Zeichensatz wieder entfernt und »topaz 8« zum aktuellen Zeichensatz erklärt.

Die beiden Funktionen »OpenDiskFont« und »AvailFonts« finden Sie in der »diskfont.library«. Diese ist nicht ROM-resident, muß also nachgeladen werden. Dazu muß auf Ihrer Boot-Diskette im »Libs«-Ordner außer dem Eintrag für die ».bmap« auch die Library stehen.

Bevor Sie zum nächsten Kapitel weitergehen, sollten Sie das bisher Besprochene kurz überdenken; Zeiger und Strukturen sind eine wichtige Grundlage, um erfolgreich mit den Libraries zu jonglieren, ohne dabei ständig den Guru herbeizuzaubern. Versuchen Sie am besten, ein erstes eigenes Programm zu schreiben, das alle behandelten Funktionen anwendet: Lassen Sie Ihr Programm feststellen, welche Schriftarten in welcher Höhe verfügbar sind und geben Sie diese Informationen in der zugehörigen Schriftart auf dem Bildschirm aus.

```
 1 T50 DECLARE FUNCTION OpenDiskFont& LIBRARY
 2 9V  DECLARE FUNCTION OpenFont& LIBRARY
 3 J0  DECLARE FUNCTION SetFont% LIBRARY
 4 a6  DECLARE FUNCTION RemFont% LIBRARY
 5 mu  LIBRARY "diskfont.library"
 6 06  LIBRARY "graphics.library"
 7 f5  DIM TextAttr%(3)
 8 3M  font$="emerald.font"+CHR$(0)
 9 uG  POKEL (VARPTR(TextAttr%(0))),SADD(font$)
10 xJ  TextAttr%(2)=20
11 ia  font&=OpenDiskFont&(VARPTR(TextAttr%(0)))
12 56  e%=SetFont%(WINDOW(8),font&)
13 lr  CLS
14 cw  PRINT "Emerald 20"
15 1F  CALL CloseFont(font&)
16 is  e%=RemFont%(font&)
17 C0  font$="topaz.font"+CHR$(0)
18 3P  POKEL (VARPTR(TextAttr%(0))),SADD(font$)
19 2W  TextAttr%(2)=8
20 6a  font&=OpenFont&(VARPTR(TextAttr%(0)))
21 EF  e%=SetFont%(WINDOW(8),font&)
(C) 1988 M&T
```

Listing 4. Mit »OpenDiskFont laden Sie Zeichensätze von Diskette

```
 1 B90 'Dieses Programm zeigt die Verwendung
 2 An  'der verschiedenen Zeichenmodi von
 3 LL  'BASIC aus
 4 y4  LIBRARY "graphics.library"
 5 mg  DIM mode%(3),mode$(3),pat%(1)
 6 GZ  FOR i=0 TO 3
 7 FH2   READ mode$(i),mode%(i)
 8 DI0 NEXT
 9 X7  DATA JAM1,0,JAM2,1,COMPLEMENT,2,INVERSEVID,4
10 M2  pat%(0)=&HFF0:pat%(1)=&HF0
11 1J  DATA &H8181,&H4242,&H2424,&H1818
12 33  DATA &H1818,&H2424,&H4242,&H8181
13 p9  WINDOW 3,,(100,30)-(200,95),0
14 z6  PATTERN &HFFFF,pat%
15 Pi  FOR i=0 TO 3
16 HD2   x1=5:x2=x1+8*LEN(mode$(i))+5
17 u0    y1=16*i+6:y2=y1+11
18 Pq    LINE (x1,y1)-(x2,y2),2,b
19 vn    PAINT (x1+2,y1+2),2
20 wY    CALL SetDrMd(WINDOW(8),mode%(i))
21 NP    LOCATE 2*(i+1),2
22 ex    PRINT mode$(i)
23 5F    CALL SetDrMd(WINDOW(8),1)
24 TY0 NEXT
25 oq  WHILE INKEY$="":SLEEP:WEND
26 h0  WINDOW CLOSE 3
(C) 1988 M&T
```

Listing 5. Die verschiedenen »DrawModes« können auch von Basic genutzt werden

Grafik

Solange Sie nur in Basic programmieren, wird zwischen Text und Grafik unterschieden. Text wird mit dem »PRINT«-Befehl ausgegeben, Grafik mit »LINE« oder »CIRCLE«. Ein Nebeneffekt dieser Unterscheidung ist, daß Sie beides nur schwer mischen können. Wenn Sie mit »PRINT« über eine Grafik schreiben, ziehen Sie eine regelrechte »Schleifspur« hinter sich her. Der »PRINT«-Befehl gibt Texte normalerweise im Modus »JAM2« aus (siehe unten). Der komplette Bereich, in den geschrieben wird, ist dabei in der Hintergrundfarbe hinterlegt. Dadurch ist von einer eventuell vorher gezeichneten Grafik nicht mehr viel übrig. Der Amiga hat einige eingebaute Grafikroutinen, die von Basic nur unvollständig genutzt werden. Diese erkennen prinzipiell keinen Unterschied zwischen Grafik und Text. Darüber hinaus bieten die Systemroutinen eine enorme Flexibilität bei der Textausgabe.

Dieser Befehl hat zwei Parameter: den schon bekannten »RastPort«-Zeiger (»WINDOW(8)«) sowie den gewünschten Zeichenmodus (»DrawMode«) in einer Integervariablen, von der wieder nur die untersten vier Bit verwendet werden. Diese haben die aus Tabelle 3 ersichtliche Bedeutung.

In unserem Beispielprogramm (Listing 5) wird die Grafik durch eine »PATTERN«-Anweisung hergestellt. Diese veranlaßt den »LINE«-Befehl, keine einheitliche Fläche zu zeichnen. Statt dessen wird alle 16 Bildpunkte die Farbe umgeschaltet. Dadurch entsteht ein Muster, das durch die einzelnen DrawModes auf unterschiedliche Weise überschrieben wird. Vergleichen Sie den Effekt dieser DrawModes mit der Tabelle, um die genaue Bedeutung der Flags zu verstehen. Natürlich können Sie einzelne DrawModes miteinander kombinieren. Dazu addieren Sie deren Werte und legen das Ergebnis im Übergabeparameter für »SetDrMd« ab.

Wenn Sie einen veränderten DrawMode verwenden, sollten Sie daran denken, auf JAM2 zurückzuschalten, bevor die Kontrolle der Ausgabe an Basic zurückgegeben wird. Sonst kommt es eventuell zu Schwierigkeiten bei der Eingabe im Direkt-Modus.

Die DrawModes

Bit	Wert	Name	Bedeutung
0	1	JAM1	Zeichnen in Vordergrundfarbe, der Hintergrund bleibt unbeeinträchtigt
1	2	JAM2	Füllt den Zeichenbereich in der Hintergrundfarbe auf (wie vom »PRINT«-Befehl gewohnt
2	4	COMPLEMENT	komplementiert die vorhandene Farbe
3	8	INVERSID	stellt die Zeichen invertiert dar

Tabelle 3. Mit diesen Parametern setzen Sie die »DrawModes«

So kommt der Text auf den Bildschirm: die DrawModes

Bei der Textausgabe können Sie selbst wählen, auf welche Art Ihr Text erscheinen soll. Neben den im letzten Kapitel besprochenen Manipulationen des »PRINT«-Befehls können Sie noch einiges verändern. Beachten Sie in Listing 5 die Möglichkeiten zur Textdarstellung, die der Amiga in der »Graphics.library« vorrätig hält. Die zentrale Rolle spielt dabei ein Befehl: »SetDrMd«.

DrawModes im Grafikeinsatz

Listing 6 verdeutlicht, daß gerade bei der Grafikprogrammierung der Einsatz der eben besprochenen Zeichenmodi sinnvoll ist. Die erste »FOR«-Schleife zeichnet ein Muster aus zehn konzentrischen Kreisen, das in der »WHILE«-Schleife bis zum Abbruch per Tastendruck mit einem Quadrat überschrieben wird. Durch das vorher durchgeführte »SetDrMd« stellt das Betriebssystem bei jedem zu setzenden Punkt fest, welche Farbe vorher auf dem Bildschirm stand. Statt einen Punkt in der vorher festgelegten Farbe zu setzen, wird jetzt die existierende Farbe einfach invertiert.

Nur so ist es möglich, das schnelle Blinken von Listing 6 zu erreichen. Versuchen Sie nur für einen Moment, sich vorzustellen, welchen Aufwand Sie von Basic aus betreiben müßten, wenn Sie diese Abfrage jedesmal selbst durchführen wollten. Mit dem veränderten Zeichenmodus dagegen muß das Programm sogar noch mit einer Warteschleife gebremst werden. Probieren Sie ruhig, was geschieht, wenn Sie die »FOR«-Schleife innerhalb der »WHILE«-Schleife entfernen.

Tempo im Text

Gehören Sie auch zu den geduldigen Zeitgenossen, die sich noch nie an der Geschwindigkeit des »PRINT«-Befehls gestört haben? Wenn nicht, wird Ihnen Listing 7 eine interessante Anregung geben.

Als Alternative zum »PRINT«-Befehl stellt uns das Betriebssystem eine leistungsfähige Funktion zur Verfügung: Sie hat den bezeichnenden Namen Text und befindet sich ebenfalls in der »Graphics.library«. »Text« liefert normalerweise einen Fehlerwert zurück, der bei erfolgreichem Schreibvorgang den Wert Null erhält. Wenn man die Fehlerabfrage ignoriert, kann »Text« auch als Befehl eingesetzt werden. In diesem Fall darf dieser aber nicht als Funktion deklariert worden sein.

Die Syntax von »Text« ist

```
fehler=Text(Rast-
Port,txt,len)
```

wobei »txt« ein Zeiger auf das erste auszugebende Zeichen ist. Dieser wird mit »SADD (string))« ermittelt. »len« ist die Anzahl auszugebender Zeichen. »RastPort« kennen wir bereits.

Listing 7 ist eine Art Benchmark-Programm, das die Geschwindigkeit von »PRINT« mit den entsprechenden Systemroutinen vergleicht. Dabei stellt der Text-Befehl das Standard-Basic eindrucksvoll in den Schatten. Für die gleiche Aufgabe braucht »Text« nur etwa ein Zehntel der Zeit von »PRINT«.

Natürlich muß man fairerweise zugeben, daß das Rennen unter etwas ungleichen Voraussetzungen lief. »PRINT« übernimmt nämlich neben der reinen Textausgabe noch einige Kontrollfunktionen für den Programmierer. So wird beispielsweise geprüft, ob der Text in der übergebenen Länge überhaupt in das aktuelle Fenster paßt. Im anderen Fall schreibt »PRINT« in der nächsten Zeile weiter. »Text« dagegen spart sich derartige Mühen und schreibt einfach ab der angegebenen Stelle den String in seiner vollen Länge. Wenn Sie also nicht sicher sind, ob Ihr Fenster den gewünschten String aufnehmen kann, sollten Sie entweder »PRINT« verwenden oder die Abfrage selbst durchführen. Die »Graphics.library« hat auch dafür eine Funktion: »TextLength«.

Für alle Anwendungen, bei denen sicher ist, daß der Platz ausreicht, kann »Text« uneinge-

```
 1 B90 'Dieses Programm zeigt die Verwendung
 2 KR  'der Zeichenmodus COMPLEMENT von
 3 LL  'BASIC aus
 4 y4  LIBRARY "graphics.library"
 5 cV  WINDOW 3,,(100,30)-(250,105),0
 6 Fn  FOR i=1 TO 10
 7 Cf2   CIRCLE (75,38),i*7
 8 kI    CIRCLE (75,38),i*7+1
 9 EJ0 NEXT
10 CI  FOR i=0 TO 3000:NEXT
11 y9  CALL SetDrMd(WINDOW(8),2)
12 wW  WHILE INKEY$=""
13 9B2   LINE (35,18)-(115,58),,bf
14 AE    FOR i=0 TO 1000:NEXT
15 gU0 WEND
16 XE  WINDOW CLOSE 3
(C) 1988 M&T
```

Listing 6. Ein Beispiel für den Einsatz der »DrawModes« für die Grafikausgabe

```
 1 v10 LIBRARY "graphics.library"
 2 Oz  WINDOW 3,,(0,0)-(500,180),0
 3 b5  rp&=WINDOW(8)
 4 O8  txt$ ="Hier wird unter Basic gedruckt.....
       ."
 5 B4  txt2$="Und jetzt mit dem Text-Befehl aus der graphics.librar
       y"
 6 8Y  t1=TIMER
 7 j0  FOR i%=1 TO 22
 8 Fm2   PRINT PTAB(i%)txt$
 9 EJO NEXT
10 Fg  t2=TIMER
11 ac  WHILE INKEY$="":SLEEP:WEND
12 kq  CLS
13 Ln  t3=TIMER
14 nc  l%=LEN(txt2$)
15 5D  a&=SADD(txt2$)
16 s9  FOR i%=1 TO 22
17 FH2   CALL Move(rp&,i%,8*i%)
18 lD    CALL Text(rp&,a&,l%)
19 OTO NEXT
20 Vy  t4=TIMER
21 km  WHILE INKEY$="":SLEEP:WEND
22 dK  WINDOW CLOSE 3
23 DG  PRINT USING"Unter Basic wurden #.## Sekunden benötigt;";t
       2-t1
24 4I  PRINT USING"der Text-Befehl brauchte #.## Sekunden,";t4-t
       3
25 b4  PRINT USING"war also ca. #.# mal so schnell !";(t2-t1)/(t4
       -t3)
(C) 1988 M&T
```

Listing 7. Der Text-Befehl sorgt für Tempo

schränkt empfohlen werden. Zumal »Text« noch weitere Vorteile bietet: Mit dem Befehl »Move(RastPort,x,y)« kann der Cursor nicht nur zeilenweise (wie mit »LOCATE«), sondern punktgenau gesetzt werden.

Um eine Zeile zu löschen, gibt es noch den — hier nicht verwendeten — Befehl »ClearEOL(RastPort)«, der ab Cursorposition bis zum Ende der Zeile löscht (genau genommen: mit Hintergrundfarbe auffüllt).

Eigene Schriftarten mit dem Textbefehl

Mit den beiden eben vorgestellten Befehlen Text und Move lassen sich aber auch sehr eindrucksvolle Schrifteffekte erzielen.

Das Programm aus Listing 8 zeigt zum Beispiel eine Outline-Schrift. Das ist Schrift in der Hintergrundfarbe, von einer anderen Farbe umrandet. Sehen Sie es sich einmal an. Sicher werden Sie viele Ideen haben, was sich noch alles damit machen läßt.

Schnelle Grafik auch mit Basic

In Listing 9 werden einige der zahlreichen im Betriebssystem implementierten Befehle vorgestellt, die die fantastischen Grafikfähigkeiten des Amiga nutzen. Hier wird nichts geboten, was nicht auch in Basic zu realisieren wäre. Der Vorteil liegt jedoch in der einfacheren Handhabung. Sie sind zudem schneller als die Basic-Routinen und außerdem notwendig für das Zeichnen in vollständig selbst erstellten Fenstern (diese werden etwas später behandelt).

Der erste neue Befehl, dem wir in Listing 9 begegnen, ist der »SetRast«-Befehl mit der Syntax

```
SetRast(RastPort,Color)
```

Er färbt einen Rastport (eines Screens oder eines Windows) mit der angegebenen Farbnummer. Das gleiche in Basic würde beispielsweise lauten

```
COLOR 1,2:CLS:COLOR 1,0
```

Diese Variante wäre allerdings erheblich langsamer.

Weitere Befehle sind:

SetAPen(RastPort,color)

Er dient zum Setzen der Zeichenfarbe. Analog gibt es »SetBPen« für die Hintergrundfarbe.

RectFill(RastPort,xl,yo,xr,yu)
bewirkt exakt das gleiche wie der Basic-Befehl »LINE (xl,yo)(xr,yu),,bf«.

DrawEllipse(RastPort,xm,ym,xr,yr)
dient zum Zeichnen von Kreisen und Ellipsen. Um das Bildverhältnis (»aspect« beim Basic-»CIRCLE«-Befehl) muß man sich mit Hilfe der Radius-Parameter, »xr« und »yr«, selbst kümmern.

Draw(RastPort,x,y)
bewirkt das gleiche wie »LINE -(x,y)« in Basic. Will man eine Linie von einer anderen als der momentanen Cursorposition aus zeichnen, muß man den

```
 1 v10 LIBRARY "graphics.library"
 2 Zc  txt$ ="Outline"
 3 Q7  WINDOW 3,,(100,30)-(190,40),0
 4 c6  rp&=WINDOW(8)
 5 aU  CALL SetDrMd(rp&,0)
 6 xH  COLOR 3,0
 7 4M  FOR i%=20 TO 22
 8 2E2   FOR j%=10 TO 12
 9 9o4     CALL Move(rp&,i%,j%)
10 z6      CALL Text(rp&,SADD(txt$),LEN(txt$))
11 GL2   NEXT
12 HMO NEXT
13 vC  COLOR 0,0
14 aG  CALL Move(rp&,21,11)
15 4B  CALL Text(rp&,SADD(txt$),LEN(txt$))
16 1J  COLOR 1,0
17 ql  CALL SetDrMd(rp&,1)
18 hj  WHILE INKEY$="":SLEEP:WEND
19 rx  CLS
20 bI  WINDOW CLOSE 3
(C) 1988 M&T
```

Listing 8. Mit »Text« und »Move« bauen Sie Ihren eigenen Zeichensatz

```
 1 v10 LIBRARY "graphics.library"
 2 Dr  DIM SHARED rp&
 3 G8  DIM PolyDat%(7),PolyDat2%(7)
 4 UX  FOR i%=0 TO 7
 5 4c2   READ PolyDat1%(i%)
 6 BGO NEXT
 7 MZ  DATA 180,80,180,90,70,90,70,80
 8 Yb  FOR i%=0 TO 7
 9 C12   READ PolyDat2%(i%)
10 FKO NEXT
11 G5  DATA 320,80,360,90,320,100,280,90
12 EF  SCREEN 2,640,200,2,2
13 m7  WINDOW 3,,(100,30)-(500,150),0,2
14 mG  rp&=WINDOW(8)
15 ke  CALL SetDrMd(rp&,0)
16 NV  CALL SetRast(rp&,2)
17 QB  sLine 90,2,220
18 xm  sLine 90,12,220
19 eF  sText "Zeichenbefehle",90,220,10,1
20 kO  CALL SetAPen(rp&,3)
21 i2  CALL RectFill(rp&,25,25,90,35)
22 aB  CALL SetAPen(rp&,0)
23 nk  CALL DrawEllipse(rp&,190,35,40,15)
24 gI  CALL SetAPen(rp&,1)
25 k2  CALL Move(rp&,300,20)
26 QM  CALL Draw(rp&,370,25)
27 9W  CALL Draw(rp&,260,60)
28 Mx  CALL Draw(rp&,330,8)
29 e5  CALL Move(rp&,70,80)
30 vm  CALL PolyDraw(rp&,4,VARPTR(PolyDat1%(0)))
31 ZA  CALL Move(rp&,280,90)
32 3q  CALL PolyDraw(rp&,4,VARPTR(PolyDat2%(0)))
33 pR  CALL SetAPen(rp&,1)
34 Bv  sText "RectFill",10,95,45,1
35 ew  sText "DrawEllipse",100,180,60,1
36 BU  sText "Draw",300,60,50,2
37 o3  sText "PolyDraw",0,400,110,1
38 13  WHILE INKEY$="":SLEEP:WEND
39 Ua  SCREEN CLOSE 2
40 OJ  END
41 Wn  SUB sLine (x%,y%,l%) STATIC
42 622   CALL Move(rp&,x%,y%)
43 AC    CALL Draw(rp&,x%+l%,y%)
44 km0 END SUB
45 Ur  SUB sText (t$,xu%,w%,y%,mode%) STATIC
46 SS2   IF mode%=0 THEN x%=xu%
47 uT    IF mode%=1 THEN x%=xu%+(w%/2-4*LEN(t$))
48 je    IF mode%=2 THEN x%=xu%+w%-8*LEN(t$)
49 D9    CALL Move(rp&,x%,y%)
50 7R    CALL Text(rp&,SADD(t$),LEN(t$))
51 rt0 END SUB
(C) 1988 M&T
```

Listing 9. Schnelle Grafik-Ausgabe ermöglicht die »graphics.library«

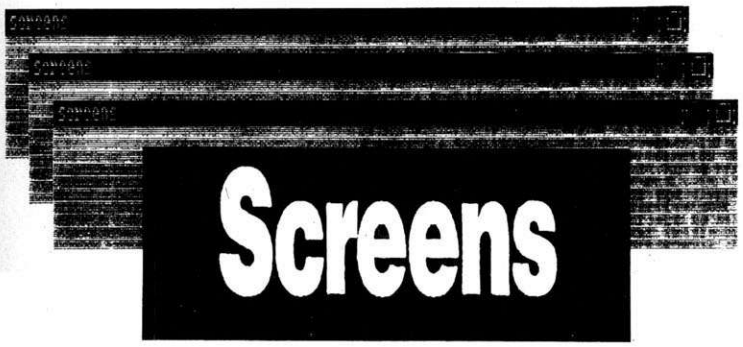
Cursor vorher mit »Move« an den Startpunkt bringen.

PolyDraw(RastPort,anz, coord)

zeichnet Linien nach einer beliebigen Anzahl Koordinaten. Es ist der einzige vorgestellte Befehl, der kein direktes Analog in Basic hat. Der Ausgangspunkt ist wieder die momentane Cursorposition. In »anz« steht die Anzahl der zu zeichnenden Linien und in »coord« der Zeiger auf ein Integer-Array, das die Koordinaten der Endpunkte für die einzelnen Linien enthält. Man kann damit mit einem einzigen Befehl beliebige Vielecke zeichnen, auch wenn deren Seiten nicht parallel zu den Window-Rändern sind.

Wenn Sie sich die ».FD«-Dateien ansehen, werden Sie noch eine Vielzahl weiterer Grafikbefehle finden. Der hier zur Verfügung stehende Raum reicht aber bei weitem nicht aus, um sie alle zu erklären und Ihre Verwendung an Beispielen zu zeigen; damit könnte man so manches Buch füllen.

Nachdem Sie nun anhand einiger einfacher Beispiele in die enormen Möglichkeiten der Betriebssystemroutinen hineingeschnuppert haben, können wir uns an etwas kompliziertere Dinge heranwagen.



Screen heißt übersetzt Bildschirm. Wenn Sie nun lesen, der Amiga könne mehrere Screens verwalten, so heißt dies nicht, daß das Ausnutzen dieser Fähigkeiten Ihr Budget belastet (etwa weil Sie sich dafür einen weiteren Monitor zulegen müßten). Screens sind vielmehr virtuelle (»gedachte«) Bildschirme auf einem Monitor, die unterschiedliche Fähigkeiten aufweisen können.

Screens sind die elementarsten Bedienungselemente des Amiga. Sie stellen die wesentlichen Einstellungen wie Auflösung oder Farben zur Verfügung, die normalerweise von allen in diesem Screen geöffneten Fenstern übernommen werden. In der Titelleiste des Screens sind eventuell vorhandene Menüs verankert.

Nachdem in Basic alles eingebaut ist, was Sie benötigen, um einen Screen zu öffnen, muß man hierfür nicht auf die Systemroutinen zurückgreifen. Für alle, die's trotzdem nicht lassen können: Die Funktionen heißen »OpenScreen« und »CloseScreen«. Sie müssen eine »NewScreen«-Struktur beziehungsweise einen »ScreenPointer« übergeben

Interessanter sind jedoch die Routinen, die bereits geöffnete Screens manipulieren. So ist es beispielsweise möglich, Screens programmgesteuert zu verschieben, einen neuen Titel einzusetzen, oder Screens in den Vorder- oder Hintergrund zu schalten.

Listing 10 zeigt eine Auswahl dieser Funktionen:

Zusätzlich zum Workbench-Screen (dieser hat in Basic standardmäßig die Nummer 1) werden »SCREEN 2« und »SCREEN 3« zwei neue Screens definiert. Zur besseren Unterscheidung stellen wir die Hintergrundfarbe auf einen anderen Wert (»PALETTE 0,1,0,0« und »PALETTE 0,.5,.5,1«).

Für alle Operationen im Zusammenhang mit Screens benötigen wir einen Zeiger auf diesen Screen. Dieser wird normalerweise von der Betriebssystemfunktion »Open Screen« zurückgeliefert. Nachdem wir uns aber entschieden hatten, ohne diese Funktion auszukommen, müssen wir einen kleinen Trick anwenden: Wir öffnen in jedem Screen ein Fenster mit dem Namen des Screens. Zu jedem Fenster gehört eine Window-Struktur. Die Basic-Funktion »WINDOW(7)« liefert einen Zeiger auf diese Struktur. Am Offset 46 steht der Zeiger auf den Screen, in dem das Fenster geöffnet wurde. Mit einer kleinen Verrenkung

```
s2&=PEEKL(WINDOW(7)+46)
```

legen wir diesen Zeiger in der Variablen »s2&« ab.

Die »MoveScreen«-Befehle in den beiden »FOR..NEXT«-Schleifen bewegen die beiden neuen Screens um 80 beziehungsweise 160 Bildpunkte nach unten. Der erste Parameter ist der Zeiger auf den Screen, der zweite gibt die Verschiebung in horizontaler Richtung in Bildpunkten an, der dritte in vertikaler Richtung. In der derzeitigen Betriebssystemversion wird nur die vertikale Verschiebung unterstützt. Daher hat der zweite Parameter keine Funktion. Wenn Sie allerdings erreichen wollen, daß Ihre Programme auch mit zukünftigen Versionen zusammenarbeiten, ohne dabei unvorhergesehene Ergebnisse zu produzieren, sollten Sie diesen Wert auf Null setzen.

Die nächste Funktion »DisplayBeep« sendet ein kurzes Bildschirmblinken. Sie benötigt den Zeiger auf den Screen, der blinken soll. Jeweils nach einer kurzen Warteschleife werden die Bildschirme nach hinten und dann wieder nach vorne durchgeschaltet (mit der Funktion »ScreenToFront« beziehungsweise »ScreenToBack«). Auch hier wird nur der Zeiger auf die Screen-Struktur als Parameter übergeben.

Am Ende werden die beiden Screens mit dem Basic-Befehl »SCREEN CLOSE« geschlossen.

```
 1 CP0 LIBRARY "intuition.library"
 2 EE  SCREEN 2,320,256,1,1
 3 SG  WINDOW 2,"Screen 2",(0,0)-(308,240),0,2
 4 fj  PALETTE 0,1,0,0
 5 ze  s2&=PEEKL(WINDOW(7)+46)
 6 MN  SCREEN 3,320,256,1,1
 7 ja  WINDOW 3,"Screen 3",(0,0)-(308,240),0,3
 8 fw  PALETTE 0,.5,.5,1
 9 6m  s3&=PEEKL(WINDOW(7)+46)
10 Wn  FOR i%=0 TO 160
11 Nf2   CALL MoveScreen(s3&,0,1)
12 HM0 NEXT
13 zJ  FOR i%=0 TO 80
14 Md2   CALL MoveScreen(s2&,0,1)
15 KP0 NEXT
16 iD  CALL DisplayBeep(s2&)
17 E1  FOR i%=0 TO 5000:NEXT
18 SM  CALL ScreenToBack(s3&)
19 8a  FOR i%=0 TO 3000:NEXT
20 OM  CALL ScreenToBack(s2&)
21 Im  FOR i%=0 TO 5000:NEXT
22 tB  CALL ScreenToFront(s2&)
23 Ce  FOR i%=0 TO 3000:NEXT
24 xG  CALL ScreenToFront(s3&)
25 GY  FOR i%=0 TO 15000:NEXT
26 HN  SCREEN CLOSE 2
27 NU  SCREEN CLOSE 3
28 6K  LIBRARY CLOSE
(C) 1988 M&T
```

Listing 10. Dieses Programm demonstriert den Gebrauch der Befehle zur Screen-Manipulation

Hinweis

An dieser Stelle ein wichtiger Hinweis: Wie weiter vorne bereits erwähnt, sollten Sie verwendete Speicherbereiche und sonstige Ressourcen in jedem Fall am Ende Ihres Programms wieder freigeben. Dabei sollten Sie jedoch **immer** mit Basic-Befehlen schließen, was mit Basic-Befehlen geöffnet wurde. Das gleiche gilt für Betriebssystembefehle. Denn Basic übernimmt einen Großteil der Ressourcen-Verwaltung in eigener Regie. Wenn Sie nun ein mit »WINDOW ...« geöffnetes Fenster mit der Betriebssystemfunktion »CloseWindow« schließen, so können Sie nicht kontrollieren, welche Ressourcen noch belegt bleiben.

Um dies zu vermeiden, halten Sie sich an folgende Regel: Zu jedem OpenFont gehört ein CloseFont, zu jedem Open Screen ein CloseScreen; ebenso muß nach jeder »WINDOW«-Anweisung ein »WINDOW CLOSE« stehen, nach jedem »LIBRARY« ein »LIBRARY CLOSE«.

Windows

Windows stellen die typische Benutzerschnittstelle unter Intuition dar. Jedes Fenster bildet eine virtuelle Konsole. Das bedeutet, solange das Fenster aktiv (angeklickt) ist, erhält es alle Informationen, die über den sogenannten »Input-Stream« ankommen. Ausgaben sind dagegen immer sichtbar, unabhängig davon, ob das Fenster gerade aktiv ist. Jedes Eingabeelement (wie Gadgets, Requester etc.) ist fest an ein Fenster gebunden.

Jedes Fenster ist also so etwas wie eine eigene Arbeitsstation für den jeweiligen Task. Damit in dem nun zwangsläufig entstehenden Durcheinander noch einigermaßen Ordnung gehalten wird, haben die Entwickler des Amiga so eine Art »Hausmeister« installiert. Sein Name ist »Intuition« und dieser Hausmeister behält alle Informationen zu dem, was gerade auf der grafischen Benutzeroberfläche abläuft, im Gedächtnis. Dieses Gedächtnis ist eine Datei namens »IntuitionBase«.

Soviel als Hintergrundinformation, denn langsam werden wir den sicheren Boden von Amiga-Basic mehr und mehr verlassen.

```
 1 6h0 'Einfache Demo fr den
 2 Cp  'MoveWindow-Befehl
 3 ER  LIBRARY "intuition.library"
 4 Rt  POKEW wptr+26,fl% XOR (2**10+2**7)
 5 EF  WINDOW 3,"MoveWindow",(40,20)-(200,60),0
 6 FE  wptr&=WINDOW(7)
 7 AJ  PRINT "von links oben..."
 8 GO  FOR i=0 TO 5000:NEXT
 9 hn  CLS
10 E5  CALL MoveWindow(wptr&,420,120)
11 7B  FOR i=0 TO 1000:NEXT
12 Uo  PRINT "nach rechts unten !"
13 LT  FOR i=0 TO 5000:NEXT
14 VC  WINDOW CLOSE 3
15 zu  END
(C) 1988 M&T
```

Listing 11. Fenster lassen sich auch programmgesteuert bewegen

```
 1 330 'Demo fuer SizeWindow
 2 DQ  LIBRARY "intuition.library"
 3 bk  WINDOW 3,"SizeWindow",(40,20)-(300,36),0
 4 DC  wptr&=WINDOW(7)
 5 79  txt 1
 6 5A  FOR i=0 TO 2000:NEXT
 7 85  CALL SizeWindow(wptr&,0,8)
 8 48  FOR i=0 TO 1000:NEXT
 9 HF  txt 2
10 9E  FOR i=0 TO 2000:NEXT
11 C9  CALL SizeWindow(wptr&,0,8)
12 8C  FOR i=0 TO 1000:NEXT
13 RL  txt 3
14 DI  FOR i=0 TO 2000:NEXT
15 KK  CALL SizeWindow(wptr&,100,0)
16 11  FOR i=0 TO 10000:NEXT
17 YF  WINDOW CLOSE 3
18 2x  END
19 FX  SUB txt (i%) STATIC
20 sy2   CLS
21 Nm    RESTORE nix
22 ZS    FOR j%=1 TO i%
23 oW4     READ a$:PRINT a$
24 TY2   NEXT
25 Ym    EXIT SUB
26 FW    nix:
27 Yk    DATA "Auch die Fenstergröße kann"
28 bT    DATA "vom Programm  sehr einfach"
29 FH    DATA "verändert werden !"
30 WYO END SUB
(C) 1988 M&T
```

Listing 12. Ebenso können Sie vom Programm aus die Größe verändern

Einfache Manipulationen

Zunächst wollen wir uns mit einigen einfachen Manipulationen von Windows begnügen. Dies hat einen großen Vorteil: Wir können vorerst noch mit den Fenstern arbeiten, die von Basic zur Verfügung gestellt werden.

Listing 11 zeigt, wie man ein Window programmgesteuert bewegen kann. Zum »MoveWindow«-Befehl sei bemerkt, daß hier keine Plausibilitätskontrolle stattfindet. Wenn Sie also das Fenster an einen Bereich außerhalb des Screens bewegen wollen, wird dies von Intuition prompt erledigt: Die Folge ist eine Audienz beim Guru. Denn sobald ein Fenster über die Kante des aktuellen Screens ragt, bringt das die Weltsicht dieses Lenkers im Hintergrund gewaltig durcheinander. Sie können für dx und dy auch negative Parameter angeben, Ihr Fenster wird dann nach oben beziehungsweise links bewegt.

Besonders interessant ist die Veränderung von Window- und Screentitel. Diese Möglichkeit wird von professionellen Programmen zu vielfältigen Zwecken genutzt. Typische Anwendungsbereiche sind die Anzeige des aktuellen Directory in einer Shell oder die Anzeige von Fehlermeldungen in einem Editor. Sowohl Screen- als auch Windowtitel können mit dem Befehl »SetWindowTitles« verändert werden. Dieser benötigt drei Parameter: Einen Zeiger auf ein Fenster (auch wenn Sie nur den Screen-Titel verändern wollen) und zwei Zeiger auf Strings. Von diesen wird der erste für den Screen und der zweite für das Fenster verwendet. Eine besondere Bedeutung haben dabei die Werte 0 und -1 für diese Zeiger. Bei -1 wird der alte Titel beibehalten, bei 0 wird dieser ersatzlos gelöscht.

Auch die Größe eines Fensters kann man programmgesteuert verändern, wie Listing 12 zeigt. Auch der »SizeWindow«-Befehl überprüft die Parameter nicht; sie sollten also nicht versuchen, das Fenster größer als den Screen zu machen oder eine negative Höhe oder Breite zu bewirken. Dies könnte zwar Ihren Amiga nicht gefährden, bringt Ihnen aber unnötige Guru-Meldungen. Als Parameter werden die selben verwendet wie für den »MoveWindow«-Befehl.

Häufig kann es vorkommen, daß Sie dem Benutzer zwar die Veränderung der Fenstergröße erlauben wollen, die von Basic vorgegebenen Minimal- und Maximalwerte für Ihr Programm aber nicht genügen (denken sie nur an die »Text«-Routine). In diesem Falle können Sie mit der Funktion »WindowLimits« die Grenzwerte für die Fenstergröße neu festlegen. Hierbei müssen Sie aber verschiedene Besonderheiten beachten. Wenn Sie einen der Parameter (minH, maxH, minB, maxB) so angeben, daß er im Widerspruch zu der aktuellen Fenstergröße steht (beispielsweise minH größer als die aktuelle Fensterhöhe), wird dieser Parameter nicht verändert. Die anderen angegebenen Grenzwerte werden aber gesetzt, sofern sie gültig sind. Die Funktion liefert dann den Fehlerwert 0 zurück. Wenn alle Parameter gesetzt wurden, erhalten Sie den Wert 1. Sollten Sie einen der Parameter nicht verändern wollen, setzen Sie ihn im Funktionsaufruf auf 0.

Listing 13 zeigt ein Beispiel für die Anwendung der »WindowLimits«-Funktion. Auf eine Behandlung des rückgemeldeten Fehlerwertes wurde verzichtet, da das Programm so aufgebaut ist, daß kein Fehler auftreten kann. Üblicherweise sollten Sie aber eine Reaktion auf auftretende Fehler vorsehen.

Vielleicht das eindrucksvollste Beispiel ist Listing 14. Es zeigt Ihnen, wie Sie programmgesteuert verhindern können, daß der Benutzer den Aufbau einer Grafik oder einer sonstigen Ausgabe in allen Phasen miterlebt. Durch die Befehle »WindowToBack« und »WindowToFront« können Sie ein Fenster beliebig in den Hintergrund verbannen und, wenn alles fertig ist, wieder hervorzaubern. Dieses Fenster sollte aber den Basic-Typ-Parameter 16 (oder höher) haben.

Jedem Fenster seinen Mauszeiger

Wußten Sie eigentlich, daß jedes Window seinen eigenen Mauszeiger haben kann? Man kann sogar den Mauszeiger eines Fensters abhängig von der Position der Maus verändern (dies wird beispielsweise in Programmen wie »GoAmiga! Datei« oder »Excellence« aus-

```
 1 cI0 'Demo fuer WindowLimits
 2 0g  DECLARE FUNCTION WindowLimits LIBRARY
 3 ER  LIBRARY "intuition.library"
 4 66  WINDOW 3,"WindowLimits",(40,20)-(300,100),1
 5 ED  wptr&=WINDOW(7)
 6 sN  GOSUB limits
 7 Vw  br%=(x2-x1)/8
 8 A1  WIDTH br%
 9 Tu  PRINT "Verkleinern Sie einmal das"
10 ya  PRINT "Fenster, soweit es geht !"
11 8Z  WHILE (WINDOW(2)>x1) OR (WINDOW(3)>y1)
12 zc2   SLEEP
13 eS0 WEND
14 qD  CALL SizeWindow(wptr&,200,60)
15 KY  FOR i=0 TO 300:NEXT
16 Dg  e=WindowLimits(wptr&,WINDOW(2),WINDOW(3),WINDOW(2)+50,WINDOW
       (3)+30)
17 KC  CLS:PRINT "ok!"
18 eH  PRINT
19 Kx  PRINT "Jetzt versuchen Sie das
20 9H  PRINT "bitte nochmals !"
21 hK  PRINT
22 sQ  PRINT "ende = <RETURN>"
23 3Y  WHILE e$<>CHR$(13)
24 cE2   e$=INKEY$
25 qe0 WEND
26 hO  WINDOW CLOSE 3
27 B6  END
28 MW  limits:
29 10  x1=PEEKW(wptr&+16)
30 Aa  y1=PEEKW(wptr&+18)
31 Lp  x2=PEEKW(wptr&+20)-40
32 Ix  y2=PEEKW(wptr&+22)-20
33 tV  RETURN
(C) 1988 M&T
```

Listing 13. Oft ist eine Begrenzung der Fenstergrößen hilfreich

genutzt). Um einem Fenster einen neuen Mauszeiger zu verleihen, muß man nur die Grafikdaten für den gewünschten Mauszeiger in einer Variablen speichern und dann den Befehl »SetPointer« aufrufen. Die Syntax dieses Befehls ist

```
SetPointer(wp&, pnt&, h%,
b%, xhot%, yhot%)
```

wobei »wp&« der bekannte Zeiger auf die Window-Struktur ist (wp&()=WINDOW(7)), »pnt&« ein Zeiger auf die Grafikdaten. Diese werden mit »READ« eingelesen. Mit »h%« und »b%« geben Sie die Höhe und die Breite (b%<=16) des neuen Mauszeigers an und »xhot%« beziehungsweise »yhot%« bezeichnen den aktiven Punkt (hot spot) des Mauszeigers. »xhot%« und »yhot%« müssen als negative Werte angegeben werden, sonst liegt der aktive Punkt außerhalb der Grafik. Den Aufbau unseres Mauszeigers entnehmen Sie Bild 1.

Die erste und die letzte Zeile der Definition dürfen nur Null-Bytes enthalten. Außerdem müssen die Grafikdaten für den Mauszeiger (wie alle anderen Daten, die für die Spezial-Chips des Amiga bestimmt sind) in den unteren 512k des RAM liegen (Chip-RAM). Es gibt in Basic keine Möglichkeit, dies zu garantieren. Daher wird auf eine Betriebssystemroutine zurückgegriffen. Mit der Funktion

```
block&=AllocMem&
(size&, mode&)
```

können Sie einen zusammenhängenden Speicherbereich reservieren. »size&« ist die Größe des angeforderten Speicherbereiches; »mode&« gibt an, welche Art von Speicher angefordert werden soll. Für »mode&« gibt es folgende Möglichkeiten (siehe Kasten unten links).

Es ist wenig sinnvoll, »MEMF__CHIP« und »MEMF__FAST« zugleich anzugeben.

Bit	Wert	Name	Funktion
0	1	MEMF__PUBLIC	(Speicher kann in beliebigem Bereich liegen)
1	2	MEMF__CHIP	(Speicher wird im Chip-RAM belegt)
2	4	MEMF__FAST	(im Speicher wird im Fast-RAM belegt)
16	65536	MEMF__CLEAR	(Speicher wird bei Übergabe mit Nullen gefüllt)

```
DATA &H0000,&H0000                1                1
DATA &HFFFF,&HFFFF                1000000000000001
DATA &HFFFF,&HFFFF                1000000000000001
DATA &H0000,&H0000                1                1
DATA &H01C0,&H0000                1       \\\      1
DATA &H03C0,&H0000                1      \\\\      1
DATA &H01C0,&H0000                1       \\\      1
DATA &H01C0,&H0000                1       \\\      1
DATA &H01C0,&H0000                1       \\\      1
DATA &H01C0,&H0000                1       \\\      1
DATA &H07F0,&H0000                1     \\\\\\\    1
DATA &H0000,&H0000                1                1
DATA &HFFFF,&HFFFF                1000000000000001
DATA &HFFFF,&HFFFF                1000000000000001
DATA &H0000,&H0000                1                1
```

Bild 1. So bauen Sie mit Data-Zeilen ein Sprite

```
 1 1A0 'Demo fuer WindowToBack/Front
 2 0g  DECLARE FUNCTION WindowLimits LIBRARY
 3 ER  LIBRARY "intuition.library"
 4 iq  txt$="Während in diesem Fenster der Text ausgegeben wird,"
 5 03  txt$=txt$+"wird im Hintergrund eine Grafik gezeichnet !
       "
 6 Nv  txt$=txt$+CHR$(13)+"<TASTE>"
 7 dK  SCREEN 2,640,250,2,2
 8 Cm  WINDOW 5,"",(0,0)-(631,235),0,2
 9 qI  WINDOW 3,"Text",(4,2)-(410,35),16,2
10 o2  w3ptr&=WINDOW(7)
11 Pc  WINDOW 4,"Graphics",(50,50)-(600,230),16,2
12 t8  w4ptr&=WINDOW(7)
13 nW  CALL WindowToBack(w4ptr&)
14 Pf  FOR i=0 TO 500:NEXT
15 jU  ON TIMER(.1) GOSUB text
16 tz  TIMER ON
17 SP  WHILE NOT allesaus%
18 5M2   x1%=20+i%\5+i%*SIN(i%/20)\5
19 uX    x2%=i%
20 yr    y1%=i%\2
21 cy    y2%=40+39*COS(i%/30)
22 vx    LINE (x1%,y1%)-(x2%,y2%)
23 RU    i%=i%+2
24 pd0 WEND
25 aA  TIMER OFF
26 pr  WHILE INKEY$="":SLEEP:WEND
27 fn  CALL WindowToFront(w4ptr&)
28 3H  WINDOW OUTPUT 3
29 17  CLS
30 Ij  PRINT "<TASTE>"
31 uw  WHILE INKEY$="":SLEEP:WEND
32 nU  WINDOW CLOSE 3
33 tb  WINDOW CLOSE 4
34 zi  WINDOW CLOSE 5
35 QW  SCREEN CLOSE 2
36 KF  END
37 cB  text:
38 DR  WINDOW OUTPUT 3
39 lp  j%=j%+1
40 mG  PRINT MID$(txt$,j%,1);
41 sF  IF (j% MOD 51)=0 THEN PRINT
42 Lz  IF j%=LEN(txt$) THEN allesaus%=-1
43 0Y  WINDOW OUTPUT 4
44 4g  RETURN
(C) 1988 M&T
```

Listing 14. Mit »WindowToBack« können Sie Fenster gezielt in den Hintergrund verbannen

»MEMF__PUBLIC« ist überall dort nötig, wo mehrere Tasks auf den gleichen Speicherbereich zugreifen oder wo Interrupts den Speicher verwenden. Hiermit werden normalerweise nur die Speicherbereiche für Ports und Messages und Ähnliches belegt. Mit »MEMF__PUBLIC« belegter Speicher ist nicht verschiebbar.

In Listing 16 sehen Sie ein Beispiel für den »SetPointer«-Befehl. In diesem Programm wird auch gezeigt, wie man ein Fenster programmgesteuert aktivieren kann. Dies wird durch den Befehl

```
ActivateWindow(win&)
```

erledigt, der erst seit Version 1.2 des Betriebssystems zur Verfügung steht.

```
 1 QHO DECLARE FUNCTION AllocMem& LIBRARY
 2 2f  LIBRARY "exec.library"
 3 ER  LIBRARY "intuition.library"
 4 bB  mode&=2**1+2**16
 5 Vk  pnt&(1)=AllocMem&(60,mode&)
 6 Yo  pnt&(2)=AllocMem&(60,mode&)
 7 Ha  FOR i=1 TO 2
 8 xa2   FOR j=0 TO 29
 9 vZ4     READ a%
10 J7      a&=a%+65536&*(a%<0)
11 Fm      POKEW (pnt&(1)+2*j),a&
12 HM2   NEXT
13 INO NEXT
14 dj  wp&(0)=WINDOW(7)
15 OP  WINDOW 4,"Fenster 1",(0,20)-(300,180),0
16 ln  wp&(1)=WINDOW(7)
17 Mj  WINDOW 5,"Fenster 2",(320,20)-(620,180),0
18 tr  wp&(2)=WINDOW(7)
19 9e  CALL ActivateWindow(wp&(0))
20 Bn  CALL SetPointer(wp&(1),pnt&(1),14,16,-7,-7)
21 Jx  CALL SetPointer(wp&(2),pnt&(2),14,16,-7,-7)
22 3A  WINDOW OUTPUT 4:GOSUB prt
23 tr  t=TIMER:WHILE (TIMER-t)<30:SLEEP:WEND
24 dL  CLS:WINDOW OUTPUT 5:GOSUB prt
25 do  cnt%=1:ON TIMER(1) GOSUB auto:TIMER ON
26 qs  t=TIMER:WHILE (TIMER-t)<20:SLEEP:WEND
27 Bg  ON TIMER (0) GOSUB 0
28 In  CALL ActivateWindow(wp&(0))
29 dK  CLS:WINDOW OUTPUT 4:GOSUB prt
30 Ip  ON MOUSE GOSUB maus:MOUSE ON
31 SO  WHILE NOT aus:SLEEP:WEND
32 Rd  ON MOUSE GOSUB 0
33 mL  CloseThings:
34 fP  CALL FreeMem(pnt&(1),60)
35 lW  CALL FreeMem(pnt&(2),60)
36 te  WINDOW CLOSE 4:WINDOW CLOSE 5
37 LG  END
38 O7  '------------------------------
39 h5  prt:
40 Mf  FOR i%=6 TO 10
41 oy2   READ a$:LOCATE i%,8:PRINT a$
42 lq0 NEXT
43 3f  RETURN
44 Eb  auto:
45 HS  CALL ActivateWindow(wp&(cnt%))
46 gv  cnt%=cnt%+1
47 OW  IF cnt%>2 THEN cnt%=0
48 8k  RETURN
49 7R  maus:
50 WB  aus=-1
51 Bn  RETURN
52 EL  '------------------------------
53 W1  pnt1:
54 Es  DATA &H0000,&H0000
55 bX  DATA &HFFFF,&HFFFF
56 cY  DATA &HFFFF,&HFFFF
57 Hv  DATA &H0000,&H0000
58 De  DATA &H01C0,&H0000
59 Or  DATA &H03C0,&H0000
60 Fg  DATA &H01C0,&H0000
61 Gh  DATA &H01C0,&H0000
62 Hi  DATA &H01C0,&H0000
63 Ij  DATA &H01C0,&H0000
64 5Q  DATA &H07F0,&H0000
65 P3  DATA &H0000,&H0000
66 mi  DATA &HFFFF,&HFFFF
67 nj  DATA &HFFFF,&HFFFF
68 S6  DATA &H0000,&H0000
69 rN  pnt2:
70 U8  DATA &H0000,&H0000
71 rn  DATA &HFFFF,&HFFFF
72 so  DATA &HFFFF,&HFFFF
73 XB  DATA &H0000,&H0000
74 pA  DATA &H03E0,&H0000
75 o7  DATA &H0770,&H0000
76 GS  DATA &H0070,&H0000
77 i1  DATA &H01E0,&H0000
78 do  DATA &H0380,&H0000
79 sB  DATA &H0770,&H0000
80 Lg  DATA &H07F0,&H0000
81 fJ  DATA &H0000,&H0000
82 2y  DATA &HFFFF,&HFFFF
83 3z  DATA &HFFFF,&HFFFF
84 iM  DATA &H0000,&H0000
```

```
85 eN  txt1:
86 YI  DATA "Klicken Sie irgendein","Fenster an, um den"
87 O1  DATA "Effekt des Befehles","SetWindowPointer zu","sehen
       !"
88 mW  txt2:
89 fV  DATA "Jetzt wird jede Se-","kunde um ein Fenster"
90 Gb  DATA "weitergeschaltet, um","den ActivateWindow-","Befehl
       zu zeigen !"
91 uf  txt3:
92 VK  DATA "Drücken Sie die linke","Maustaste, um das Pro-","gram
       m zu beenden !","","","",""
(C) 1988 M&T
```

Listing 15. Jedes Fenster kann einen eigenen Maus-Zeiger erhalten — »SetPointer« ist die Lösung

```
 1 Cq0 'Einfache Demo fuer den
 2 Tr  'SetWindowTitles-Befehl
 3 SJ  DECLARE FUNCTION AllocMem& LIBRARY
 4 4h  LIBRARY "exec.library"
 5 GT  LIBRARY "intuition.library"
 6 oD  'Allocate Memory for titles (protects titles from garbage co
       llection)
 7 Ox  st&=AllocMem&(80, 65540&)
 8 9A  wt&=AllocMem&(80, 65540&)
 9 BC  SCREEN 2,640,200,2,2
10 6k  WINDOW 4," ",(20,100)-(620,180),0,2
11 KJ  wptr&=WINDOW(7)
12 5n  PRINT "Der Screen hat noch keinen Titel !"
13 ZC  PRINT
14 Pk  INPUT "Wie soll er heißen";st$
15 PX  st$=st$+CHR$(0)
16 xx  st&=SADD(st$)
17 VV  CALL SetWindowTitles(wptr&,-1,st&)
18 qw  CLS
19 yL  PRINT "Geht doch, oder?"
20 gJ  PRINT
21 3F  INPUT"Jetzt geben wir aber noch dem Fenster einen Titel: ",w
       t$
22 2y  wt$=wt$+CHR$(0)
23 aO  wt&=SADD(wt$)
24 Ck  CALL SetWindowTitles(wptr&,wt&,-1)
25 OT  FOR i=0 TO 2000:NEXT
26 y4  CLS
27 Vf  PRINT "Der Screentitel gefällt mir nicht ..."
28 RW  FOR i=0 TO 2000:NEXT
29 nG  CALL SetWindowTitles(wptr&,-1,0)
30 ii  PRINT "... also lösche ich ihn !"
31 Xd  FOR i=0 TO 3000:NEXT
32 sV  PRINT
33 Wr  PRINT "Und der WindowTitel wird auch geändert !"
34 qp  wt$="Bye, bye, Ronnie!"+CHR$(0)
35 ma  wt&=SADD(wt$)
36 Ow  CALL SetWindowTitles(wptr&,wt&,-1)
37 MM  FOR i=0 TO 10000:NEXT
38 yg  WINDOW CLOSE 4
39 Ua  SCREEN CLOSE 2
40 OJ  END
(C) 1988 M&T
```

Listing 16. Mit »SetWindowTitles« ändern Sie jederzeit den Titel Ihres Fensters

Unser erstes selbstgebautes Fenster

Bevor Sie jetzt Hammer und Meißel bereitlegen: Dieser Kurs spielt sich auch weiterhin auf dem Bildschirm ab. Sie werden aber von jetzt an immer weiter aus dem sicheren Basic-Fundament herauswachsen. So haben Sie auf einige Arten von Fenstern, die auf Betriebssystemebene problemlos genutzt werden können, von Basic aus keinen Zugriff.

Wenn Sie nun endgültig Basic die Herrschaft über Screens und Windows entreißen, sollten Sie ein solides Handwerkszeug verfügbar haben. In Listing 17 finden Sie den wichtigsten Grundstock. Es besteht aus keinem ablauffähigen Programm, sondern aus einer Subroutine, die nichts weiter macht als aus den übergebenen Daten eine »NewWindow«-Struktur zusammenzustellen und diese an die »OpenWindow«-Funktion zu übergeben. Auch wenn Ihnen dieses Vorgehen

```
 1 GmO SUB NewWindow(win&,LeftEdge%,TopEdge%,xWidth%,Height%,Detail
       Pen%,BlockPen%,IDCMP$,Flags$,FirstGadget&,CheckMark&,Title$,x
       Screen&,BitMap&,MinWidth%,MinHeight%,MaxWidth%,MaxHeight%,Typ
       e$) ST
 2 x1  TIC
 3 AE  ' Mit ! gekennzeichnete Strukturelemente
 4 6Z  ' muessen uebergeben werden; alle nicht
 5 kV  ' gekennzeichneten brauchen nicht initialisiert
 6 iv  ' zu werden (0 uebergeben).
 7 9v  DIM NewWin%(23)
 8 by  tit&=0
 9 iF  IF Title$<>"" THEN
10 Hd2   Title$=Title$+CHR$(0)
11 Q7    tit&=AllocRemember&(0, LEN(Title$), 65540&)
12 3M    CALL CopyMem(SADD(Title$), tit&, LEN(Title$))
13 920 END IF
14 ut  ' Umwandeln des Strings fuer IDCMP-Flags
15 Fz  ' in einen LONG-Wert (wie bei Lattice-C)
16 LH  IDCMPFlags&=0
17 72  IF IDCMP$="" GOTO flg:
18 ou  RESTORE IDCMPdat
19 77  i&=0
20 8v  WHILE i&<>-1
21 Jv2   READ a$,i&
22 k2    IF INSTR(IDCMP$,a$)<>0 THEN
23 qn4     IDCMPFlags&=IDCMPFlags&+i&
24 KD2   END IF
25 qe0 WEND
26 w0  ' Umwandeln des Strings fuer Flags
27 RB  ' in einen LONG-Wert (wie bei Lattice-C)
28 4z  flg:
29 xS  Flags&=0
30 jm  IF Flags$="" GOTO flg2:
31 37  RESTORE Flagdat
32 KK  i&=0
33 L8  WHILE i&<>-1
34 W82   READ a$,i&
35 uV    IF INSTR(Flags$,a$)<>0 THEN
36 km4     Flags&=Flags&+i&
37 XQ2   END IF
38 3r0 WEND
39 Pw  ' Umwandeln des Strings fuer Screentyp
40 E5  ' in einen SHORT-Wert (wie bei Lattice-C)
41 9G  flg2:
42 af  Type%=0
43 JW  IF Type$="" THEN Type%=1:GOTO flg3
44 cN  RESTORE Screendat
45 XX  i&=0
46 YL  WHILE i&<>-1
47 jL2   READ a$,i&
48 dq    IF INSTR(Type$,a$)<>0 THEN
49 Cf4     Type%=Type%+i&
50 kd2   END IF
51 G40 WEND
52 PX  flg3:
53 tw  ' Aufbau der NewWindow-Struktur
54 CS  NewWin%(0)=LeftEdge%                    ' !
55 xW  NewWin%(1)=TopEdge%                     ' !
56 iX  NewWin%(2)=xWidth%                      ' !
57 sy  NewWin%(3)=Height%                      ' !
58 MD  POKE  VARPTR(NewWin%(4)),DetailPen%     ' !
59 JA  POKE  VARPTR(NewWin%(4))+1,BlockPen%    ' !
60 ZK  POKEL VARPTR(NewWin%(5)),IDCMPFlags&
61 d2  POKEL VARPTR(NewWin%(7)),Flags&         ' !
62 Dm  POKEL VARPTR(NewWin%(9)),FirstGadget&
63 di  POKEL VARPTR(NewWin%(11)),CheckMark&
64 Dn  POKEL VARPTR(NewWin%(13)),tit&          ' !
65 Cv  POKEL VARPTR(NewWin%(15)),xScreen&      ' !
66 fN  POKEL VARPTR(NewWin%(17)),BitMap&
67 kv  NewWin%(19)=MinWidth%                   ' wenn 0, wird es
       auf xWidth gesetzt
68 XM  NewWin%(20)=MinHeight%                  ' nur, wenn SIZING
       -Gadget verlangt wurde
69 tn  NewWin%(21)=MaxWidth%                   '  "
70 09  NewWin%(22)=MaxHeight%                  '  "
71 bd  NewWin%(23)=Type%                       '  "
72 eK  ' Oeffnen des Fensters und Freigeben des
73 Fv  ' von der NewWindow-Struktur belegten Speichers
74 SI  win&=OpenWindow&(VARPTR(NewWin%(0)))
75 7i  ERASE NewWin%
76 Nb  EXIT SUB
77 TC  ' Daten fuer IDCMP-Flags und Flags
78 i4  IDCMPdat:
79 nq  DATA SIZEVERIFY, 1, NEWSIZE, 2, REFRESHWINDOW, 4, MOUSEBUTTO
       NS, 8
80 WQ  DATA MOUSEMOVE, 16, GADGETDOWN, 32, GADGETUP, 64, REQSET, 12
       8
81 kz  DATA MENUPICK, 256, CLOSEWINDOW, 512, RAWKEY, 1024, REQVERIF
       Y, 2048
82 Wk  DATA REQCLEAR, 4096, MENUVERIFY, 8192, NEWPREFS, 16384, DISK
       INSERTED, 32768
83 RY  DATA DISKREMOVED, 65536, WBENCHMESSAGE, 131072, ACTIVEWINDOW
       , 262144
84 Rj  DATA INACTIVEWINDOW, 524288, DELTAMOVE, 1048576, VANILLAKEY,
        2097152
85 kI  DATA INTUITICKS, 4194304, X, -1
86 d6  Flagdat:
87 Lx  DATA WINDOWSIZING, 1, WINDOWDRAG, 2, WINDOWDEPTH, 4, WINDOWC
       LOSE, 8
88 ma  DATA SIZEBRIGHT, 16, SIZEBOTTOM, 32, SIMPLE_REFRESH, 64, SUP
       ER_BITMAP, 128
89 iJ  DATA BACKDROP, 256, REPORTMOUSE, 512, GIMMEZEROZERO, 1024, B
       ORDERLESS, 2048
90 nF  DATA ACTIVATE, 4096, NOCAREREFRESH, 131072, X, -1
91 B9  Screendat:
92 jo  DATA WBENCHSCREEN, 1, CUSTOMSCREEN, 15, X, -1
93 XZ  END SUB
(C) 1988 M&T
```

Listing 17. Dieses Unterprogramm binden Sie in alle Programme ein, die ein eigenes Fenster öffnen. Dazu muß vom Hauptprogramm die »intuition.library« geöffnet und »AllocRemember« sowie »OpenWindow« als Funktion deklariert werden.

im Moment — verglichen mit Basic — etwas mühsam erscheint, werden Sie den »WINDOW«-Befehl schnell vergessen haben. Denn die Möglichkeiten, die Ihnen vom System geboten werden, sind ungleich vielfältiger.

Die Routine aus Listing 17 können Sie an jedes Programm anhängen, das ein eigenes Fenster öffnen soll. Da sie uns in der nächsten Zeit noch intensiv beschäftigen wird, sollten Sie sehr genau verstanden haben, was dabei geschieht:

Die ungewohnt lange Parameterliste hinter »SUB NewWindow« ist nötig, damit wirklich alle Möglichkeiten ausgenutzt werden können. Folgende Parameter müssen übergeben werden:

win&	Zeiger auf ein Fenster
LeftEdge%	Abstand des linken Fensterrands vom linken Screenrand
TopEdge%	Abstand des oberen Fensterrands vom oberen Screenrand
xWidth%	Breite des Fensters
Height%	Höhe des Fensters
DetailPen%	Farbnummer für Text, Gadgets etc.
BlockPen%	Farbnummer für Hintergrund
IDCMP$	Flags für die Ein- und Ausgabe
Flags$	Flags für die Fenstereigenschaften
FirstGadget&	Zeiger auf eigene Gadgets
Checkmark&	Zeiger auf ein selbstdefiniertes Checkmark
Title$	Fenstertitel
xScreen&	Zeiger auf den zugehörigen Screen
BitMap&	Zeiger auf eine eigene Bitmap
MinWidth%	Mindestbreite des Fensters
MinHeight%	Mindesthöhe des Fensters
MaxWidth%	Maximalbreite des Fensters
MaxHeight%	Maximalhöhe des Fensters
Type$	Flags für den Screentyp

Lassen Sie sich von dieser Menge an Parametern nicht entmutigen. Sie werden sehen, daß Sie einige davon fast nie benötigen werden, aber wenn die Routine universell verwendbar sein soll, muß sie alle Möglichkeiten berücksichtigen.

Die erste Anweisung legt mit

```
DIM NewWin%(23)
```

ein Array an, das die NewWindow-Struktur aufnehmen soll. Danach wird ein Zeiger auf einen String angelegt, der den übergebenen Titel aufnimmt. Der Text muß dabei in einen reservierten Speicherbereich kopiert werden, damit er von der nächsten »garbage collection« nicht beschädigt wird.

IDCMP heißt »Intuition Direct Communications Message Port«. Die zugehörigen Flags filtern den sogenannten »Input-Stream«. Für jedes ge-

setzte Flag wird jeweils eine Sorte Informationen zugelassen, alle anderen werden — solange dieses Fenster aktiv ist — abgelehnt. Die einzelnen Flags finden Sie in den Data-Zeilen am Ende der Unterroutine. Ihre Bedeutung:

SIZEVERIFY	das Sizing-Gadget wurde bewegt
NEWSIZE	die Fenstergröße wurde verändert
REFRESHWINDOW	das Fenster war verdeckt und muß neu gezeichnet werden
MOUSEBUTTONS	ein Mausknopf ist gedrückt
MOUSEMOVE	die Maus wurde bewegt
GADGETDOWN	der linke Mausknopf ist über einem Gadget gedrückt
GADGETUP	der linke Mausknopf wurde über dem Gadget wieder losgelassen
REQSET	ein Requester erscheint
MENUPICK	ein Menü wurde gewählt
CLOSEWINDOW	das Close-Gadget wurde betätigt
RAWKEY	sendet ungefilterten Tastaturcode
REQVERIFY	läßt einen Requester erst zu, wenn das Programm entsprechende Vorbereitungen getroffen hat
REQCLEAR	letzter Requester wurde bedient
MENUVERIFY	läßt die Menüs erst erscheinen wenn das Programm die nötigen Vorbereitungen getroffen hat
NEWPREFS	in Preferences wurde etwas geändert
DISKINSERTED	eine Diskette wurde in ein Laufwerk gelegt
DISKREMOVED	in einem Laufwerk wurde eine Diskette entfernt
WBENCHMESSAGE	»OpenWorkBench« oder »CloseWorkBench« wurde gesendet
ACTIVEWINDOW	Sie werden informiert, wenn das Fenster angeklickt wird
INACTIVEWINDOW	Sie werden informiert, wenn ein anderes Fenster angeklickt wird
DELTAMOVE	die Mauskoordinaten werden relativ zur letzten Meldung angegeben (nur zusammen mit »MOUSEMOVE«)
VANILLAKEY	Tastaturcode als ASCII-Wert entsprechend aktueller Keymap
INTUITICKS	ermöglicht das Einrichten eines einfachen Timers

Sind keine IDCMP-Flags gesetzt, so werden in diesem Fenster keine Systemmeldungen angenommen. Sobald mindestens eines gesetzt ist, werden zwei Ports (»UserPort« und »MessagePort«) eingerichtet, über die jegliche Kommunikation läuft. Sie werden sicher nie alle diese Flags benötigen. Viele sind für sehr spezielle Anwendungen gedacht. So ist es normalerweise völlig unnötig, das Programm über die Systemgadgets zu informieren, weil uns Intuition hier die Arbeit abnimmt. Eine Ausnahme bildet das Close-Gadget. Es muß abgefragt werden, weil das Hauptprogramm noch die Gelegenheit erhalten soll, einige Befehle zum »Aufräumen« abzusetzen. Ebenso könnte es sein, daß Sie eines der Systemgadgets für eine andere Aufgabe »mißbrauchen« wollen. In diesem Fall könnten Sie die Information an Ihr Hauptprogramm umlenken, indem Sie das entsprechende Flag setzen. Die anderen Flags werden Sie nach und nach anhand der Beispielprogramme kennenlernen.

Durch die — auf den ersten Blick etwas ungewöhnliche — Konstruktion mit Data-Zeilen ist es möglich, die IDCMP-Flags im Klartext zu übergeben. Diese werden in der »WHILE«-Schleife umgewandelt. Das gleiche Verfahren wird bei der Variablen »Flags« angewendet. Deren Bedeutungen im einzelnen:

WINDOWSIZING	ein Sizing-Gadget wird eingerichtet
WINDOWDRAG	ein Drag-Gadget wird eingerichtet
WINDOWDEPTH	ein Tiefen-Gadget wird eingerichtet
WINDOWCLOSE	ein Close-Gadget wird eingerichtet
SIZEBRIGHT	Gadget wird an die rechte Seite gelegt
SIZEBBOTTOM	Gadget wird an den unteren Rand gelegt
SIMPLEREFRESH	erhält den Fensterinhalt auch wenn dieser verschoben wird
SUPERBITMAP	erhält den Fensterinhalt auch wenn dies überdeckt oder in der Größe verändert wird
BACKDROP	das Fenster öffnet sich hinter allen anderen Fenstern
REPORTMOUSE	muß gesetzt sein, wenn die Maus über »MOUSEMOVE« abgefragt werden soll
GIMMEZEROZERO	verhindert das Überschreiben des Rahmens und der Gadgets
BORDERLESS	öffnet ein Fenster ohne Rahmen
ACTIVATE	aktiviert das Fenster beim Öffnen
NOCAREREFRESH	das Programm soll keine Information über einen benötigten Refresh erhalten

Die Window-Struktur

&ptr	NextWindow;	Zeiger auf nächstes Fenster
%	LeftEdge, TopEdge;	Linke obere Ecke
%	Width, Height;	Breite und Höhe
%	MouseY, MouseX;	Momentane Mauskoordinaten relativ zu Fenster
%	MinWidth, MinHeight;	minimale Breite und Höhe
%	MaxWidth, MaxHeight;	maximale Breite und Höhe
&	Flags;	siehe unten
&ptr	MenuStrip;	von SetMenuStrip gesetzt
&ptr	Title;	Fenstertitel
&ptr	FirstRequest;	Aktiver Requester
&ptr	DMRequest;	doubleMenu-Requester
%	ReqCount;	Anzahl aktiver Requester
&ptr	WScreen;	Screen dieses Fensters
&ptr	RPort;	RastPort dieses Windows
BYTE	BorderLeft, BorderTop, BorderRight, BorderBottom;	
&ptr	BorderRPort;	
&ptr	FirstGadget;	Zeiger auf erstes Gadget beinhaltet nicht SystemGadgets
&ptr	Parent, Descendant;	Zeiger auf Window-Strukturen
&ptr	Pointer;	Zeiger auf Mauszeigerdaten
BYTE	PtrHeight;	Höhe des Zeigers
BYTE	PtrWidth;	Breite des Zeigers
BYTE	XOffset, YOffset;	HotSpot des Zeigers
&	IDCMPFlags;	
&ptr	UserPort, WindowPort;	Zeiger auf MsgPorts
&ptr	MessageKey;	Zeiger auf IntuiMessage
UBYTE	DetailPen, BlockPen;	Farben
&ptr	CheckMark;	zeigt auf Image
&ptr	ScreenTitle;	Window-eigener Screentitel
%	GZZMouseX;	Mauskoordinaten für GZZ
%	GZZMouseY;	
%	GZZWidth;	Breite und Höhe des inneren
%	GZZHeight;	Fensters bei GZZ
&ptr	ExtData;	
&ptr	UserData;	
&ptr	WLayer;	
&ptr	IFont;	Font für Fenster

Tabelle 4. Diese Struktur wird für jedes Intuition-Fenster angelegt

Die »NewWindow«-Struktur wird per Wertzuweisung initialisiert. Integer-Werte können direkt übernommen werden. Dagegen müssen alle Werte, die mehr oder weniger Speicherplatz benötigen (beispielsweise Byte-Größe wie »DetailPen« oder Zeiger mit 4 Byte) per »POKE« oder »POKEL« übernommen werden.

Nachdem die Struktur initialisiert ist, wird sie an die »OpenWindow«-Funktion übergeben:

```
win&=OpenWindow&
(VARPTR(NewWin%(0)))
```

Der Rückgabewert ist ein Zeiger auf die Window-Struktur. Falls nicht genügend Speicher für das neue Fenster vorhanden ist, wird hier ein Wert von Null zurückgegeben. Alle anderen möglichen Fehler (vor allem im Zusammenhang mit dem Aufbau der New Window-Struktur) werden direkt mit dem Guru quittiert. Nach dieser Zeile wird die NewWindow-Struktur nicht mehr gebraucht, daher kann mit

```
ERASE NewWin%
```

der Speicherplatz freigegeben werden. An dieser Stelle wird die Subroutine verlassen, die darunter stehenden Data-Zeilen wurden bereits weiter oben verwendet.

Den Speicherbereich reservieren wir uns diesmal mit der »intuition.library«-Funktion »AllocRemember« mit der Syntax

```
buf&=AllocRemember&
(key&, size&, flags&)
```

wobei »size« und »flags« die gleiche Bedeutung haben wie bei »AllocMem«. »key« ist der Zeiger auf eine »RememberKey«-Struktur, die Parameter belegter Speicherbereiche zu einer Liste verkettet. Hier geben wir die Adresse einer Longinteger-Variablen (mit »VARPTR(x&)«) an, in der das System immer die Adresse der ersten RememberKey-Struktur hinterlegen wird. Wenn wir hier Null angeben, wird die RememberKey-Liste vom System verwaltet; durch Angabe von Zeigeradressen können wir aber auch mehrere Listen führen.

Ein großer Vorteil von »AllocRemember« ist, daß wir einiges an Verwaltungsaufwand einsparen; dies wird vom System erledigt. Wenn wir mit »AllocMem« belegten Speicher freigeben wollen, müssen wir die Speicherstellen und die Größe der Blocks (wie beim Aufruf festgelegt) kennen. Das heißt, wir müssen sie in einer Variablen »aufbewahren« und uns merken, was freigegeben werden muß. Mit »AllocRemember« dagegen ist am Ende nur ein Aufruf von »FreeRemember« nötig. Die dafür nötigen Informationen werden von Intuition aufbewahrt. Wenn Sie im Parameter »key&« einen von Null verschiedenen Wert übergeben, legt Intuition unter diesem Schlüssel eine eigene Liste für Sie an. Eine Alternative ist die Übergabe von Null. In diesem Fall legt Intuition nur eine Liste für alle »Remember«-Strukturen an. Bei »FreeRemember« sind allerdings auch alle verloren. Ein weiterer Vorteil ist, daß wir in Basic keine Variablen als »SHARED« deklarieren müssen.

Das Kopieren erfolgt mit der »exec.library«-Funktion »CopyMem« mit der Syntax

```
CopyMem
(src&, dest&, size&)
```

Das geht deutlich schneller als mit einer »POKE«-Schleife. Daneben existiert noch ein zweiter Kopierbefehl (»CopyMemQuick«), der noch etwas schneller ist. Allerdings müssen hier alle Parameter (die gleichen wie bei »CopyMem«) »long-aligned« sein, das heißt an einer durch vier teilbaren Adresse liegen.

Noch eine kleine Bemerkung zum Unterprogramm »NewWindow«: Die Parameterliste ist so lang, daß der AC-Basic-Compiler Schwierigkeiten damit hat. Wenn Sie auf das Compilieren des Programms nicht verzichten wollen, müssen Sie einige Parameter einsparen (gute Kandidaten: CheckMark, minimal- und maximal-Werte, Bitmap. Dies sind Parameter, die sehr selten gebraucht werden). Das Unterprogramm öffnet keine der benötigten Ressourcen. Wenn Sie diese Subroutine einbinden wollen, müssen Sie also die »intuition.library« und die »exec.library« geöffnet und die Funktionen »AllocRemember« und »OpenWindow« deklariert haben. Dies sollte keinesfalls vom Unterprogramm aus erledigt werden, um verhängnisvolle Doppel-Deklarationen zu vermeiden.

Listing 18 enthält das Hauptprogramm für alle Window-Demos. Hier soll lediglich die Timer-Unterbrechungsroutine »interr:« erklärt werden. In der Window-Struktur gibt es zwei Komponenten, die die Mauskoordinaten enthalten und von Intuition ständig mit den aktuellen Werten versorgt werden. Diese beiden Werte werden ausgelesen. Wenn sie in einem bestimmten Bereich liegen, wird das Flag »aus« auf -1 gesetzt.

In Listing 19 finden Sie ein einfaches Window-Demo, das lediglich den Aufruf der NewWindow-Routine demonstriert und einen Text in das neu erstellte Fenster schreibt. An den RastPort des Fensters können wir nun natürlich nicht mehr mit WINDOW(8) kommen, da AmigaBasic unser Fenster nicht als Standard-Basic-Window akzeptiert. Also müssen wir diesen Wert aus der Window-Struktur direkt auslesen (mit PEEKL), was genauso einfach geht, wenn man den Offset kennt (50).

Listing 20 ist ebenso einfach gehalten, zeigt aber einen Fenstertyp, den wir in Basic nicht öffnen können: ein »Borderless«-Window — also ein Fenster ohne Rahmen. Ein solches Fenster empfiehlt sich beispielsweise dann, wenn Sie die vollen 80 Textspalten des Bildschirmes nutzen wollen.

Daß es sich tatsächlich um ein Fenster handelt (was auf den ersten Blick nicht offensichtlich ist), erkennen Sie an der im Basic-Fenster ausgegebenen Schrift. Diese verschwindet hinter dem markierten Bereich.

Beenden Sie das Programm dadurch, daß Sie den Mauszeiger über das Feld »END« im Borderless-Fenster bewegen. Sie brauchen dieses »Pseudo-Gadget« nicht anzuklicken, die Koordinaten werden von der

```
 1 t20 ON BREAK GOSUB ignore : BREAK ON
 2 Ji  DECLARE FUNCTION OpenWindow& LIBRARY
 3 Uf  DECLARE FUNCTION AllocRemember& LIBRARY
 4 4h  LIBRARY "exec.library"
 5 GT  LIBRARY "intuition.library"
 6 06  LIBRARY "graphics.library"
 7 jp  WIDTH 75
 8 PV  s&=PEEKL(WINDOW(7)+46)
 9 7T  GOSUB Specials1  'mit MERGE nachladen
10 FS  GOSUB Draw
11 Hc  ON TIMER(.1) GOSUB interr
12 pv  TIMER ON
13 iK  WHILE NOT aus
14 EZ2   GOSUB Specials3
15 gU0 WEND
16 ou  CLS
17 Wb  'GOSUB Specials2 'nur bei Backdrop
18 Cx  CALL CloseWindow(w1&)
19 S2  CALL FreeRemember(0,-1)
20 4z  END
21 3n  '----------------------------
22 y4  ignore:
23 jL  RETURN
24 6q  '----------------------------
25 Mi  interr:
26 3j  my%=PEEKW(w1&+12)
27 5m  mx%=PEEKW(w1&+14)
28 2W  IF (mx%>145)AND(mx%<175)AND(my%>75)AND(my%<86) THEN
29 Bq2   aus=-1
30 QJ0 END IF
31 rT  RETURN
32 Ey  '----------------------------
33 O4  Draw:
34 Lj  rast&=PEEKL(w1&+50)
35 1s  CALL SetBPen(rast&,2)
36 bf  CALL ClearScreen(rast&)
37 mk  CALL SetAPen(rast&,0)
38 dz  CALL RectFill(rast&,115,40,205,61)
39 6s  CALL SetAPen(rast&,3)
40 N1  CALL RectFill(rast&,143,77,177,87)
41 2s  CALL SetAPen(rast&,2)
42 6v  CALL RectFill(rast&,119,42,201,59)
43 Ks  CALL RectFill(rast&,145,78,175,86)
44 zt  CALL SetAPen(rast&,1)
45 BS  CALL SetDrMd(rast&,0)
46 Ln  t$=t$+CHR$(0)
47 w5  CALL Move(rast&,160-4*(LEN(t$)-1),49)
48 44  CALL Text(rast&,SADD(t$),LEN(t$)-1)
49 Yu  t$="Window"+CHR$(0)
50 37  CALL Move(rast&,160-4*(LEN(t$)-1),58)
51 77  CALL Text(rast&,SADD(t$),LEN(t$)-1)
52 R0  t$="END"+CHR$(0)
53 I7  CALL Move(rast&,160-4*(LEN(t$)-1),85)
54 AA  CALL Text(rast&,SADD(t$),LEN(t$)-1)
55 Fr  RETURN
56 cM  '----------------------------
(C) 1988 M&T
```

Listing 18. Dieses Programm verwenden Sie als Steuerprogramm für Listing 19 bis Listing 21. Dazu müssen Sie Listing 17 mit einbinden.

```
 1 dE0 ' WindowDemo specials
 2 UI  Specials1:
 3 b6  NewWindow w1&,100,30,400,150,2,3,"","WINDOWSIZING ] ACTIVATE
       ] WINDOWDRAG",0&,0&,"Test",s&,0&,100,40,500,200,"WBENCHSCREE
       N"
 4 v5  rp&=PEEKL(w1&+50)
 5 1t  t$="Unser selbsterstelltes Fenster"+CHR$(0):t%=LEN(t$)
 6 D8  t&=AllocRemember&(0, t%, 65540&)
 7 dD  CALL CopyMem(SADD(t$), t&, t%):t$="Unser"
 8 5N  CALL Move(rp&,80,30)
 9 Lr  CALL Text(rp&,t&,t%-1)
10 W8  RETURN
11 nd  Specials3:
12 YA  RETURN
(C) 1988 M&T
```

Listing 19. Diese Routine verwenden Sie um ein eigenes Fenster zu öffnen. Das Hauptprogramm muß Listing 17 einbinden.

```
 1 nRO 'Borderless-Window
 2 UI  Specials1:
 3 Po  NewWindow  w1&,160,50,320,100,1,3,"","BORDERLESS",0&,0&,"",s
       &,0&,100,40,640,250,"WBENCHSCREEN"
 4 jr  t$="Borderless"
 5 kU  text$=t$+"-Window    ":i=1
 6 S4  RETURN
 7 jZ  Specials3:
 8 EH2   PRINT MID$(text$,i,1);
 9 93    i=i+1
10 T9    IF i>LEN(text$) THEN i=1
11 X90 RETURN
(C) 1988 M&T
```

Listing 20. Nicht jedes Fenster hat einen Rahmen. Dieses Beispiel zeigt Ihnen, wie Sie ein »Borderless«-Window öffnen.

»interr:«-Routine auch so erkannt.

Einen weiteren von Basic aus nicht zugänglichen Fenstertyp lernen Sie in Listing 21 kennen: das »Backdrop«-Window. Das ist ein Fenster, das hinter allen anderen Fenstern geöffnet wird und vom Anwender auch nicht in den Vordergrund gebracht werden kann (ein eventuell gesetztes Systemgadget »WindowDepth« wird ignoriert). Um es sichtbar zu machen, wird etwas Window-Akrobatik betrieben: Laden Sie Listing 18. Mit »merge« können Sie jetzt Listing 21 und Listing 17 nachladen. Das entstehende Programm speichern Sie unter dem Namen »Backdrop_Window« (bitte genau so eingeben, da dieser Name im Programm abgefragt wird). Starten Sie das Programm und versuchen Sie, irgendeines der anderen Fenster nach hinten zu klicken! Sie werden sehen, das Backdrop-Window bleibt im Hintergrund.

In diesem Programm ist auch noch das Unterprogramm »FindBasWinPars« interessant. Es sucht die Fenster mit den Namen "LIST" und "Backdrop_Window" und liefert die Adressen der beiden zugehörigen Window-Strukturen.

Dazu wird aus der Screen-Struktur die Komponente »FirstWindow« — ein Zeiger auf die Window-Struktur des ersten Windows in diesem Screen — ausgelesen. Aus dieser Struktur wird die Adresse des Fenstertitels ermittelt und dieser Titel mit den gesuchten Namen verglichen. Als erste Komponente der Window-Struktur steht ein Zeiger auf das folgende Fenster, der nach dem Vergleich den bisherigen Zeiger ersetzt (»w&=PEEKL(w&)«). Wird dieser Zeiger Null, ist das letzte Fenster durchsucht worden und das Unterprogramm wird verlassen.

Zum zweiten: Screens

Nachdem Sie anhand der Window-Strukturen schon etwas Erfahrung gesammelt haben, wollen wir noch einmal zum Thema Screens zurück-

```
 1 zo0 'Backdrop-Window
 2 UI  Specials1:
 3 dK  GOSUB FindBasWinPars
 4 FV  NewWindow  w1&,160,50,320,100,1,3,"","BACKDROP",0&,0&,"",s&,
       0&,100,40,640,250,"WBENCHSCREEN"
 5 Sv  t$="Backdrop"
 6 js  dx1%=200-PEEKW(wPrg&+8)
 7 8M  dy1%=60-PEEKW(wPrg&+10)
 8 vL  CALL SizeWindow(wPrg&,dx1%,dy1%)
 9 n8  IF wList&<>0 THEN
10 Vi2   dx2%=200-PEEKW(wList&+8)
11 JI    dy2%=60-PEEKW(wList&+10)
12 as    CALL SizeWindow(wList&,dx2%,dy2%)
13 cs    dx3%=440-PEEKW(wList&+4)
14 Gp    dy3%=-PEEKW(wList&+6)
15 Xn    CALL MoveWindow(wList&,dx3%,dy3%)
16 C50 END IF
17 XE  WINDOW 4,"Normales Fenster",(160,30)-(420,70),6
18 eG  RETURN
19 1l  '----------------------------
20 BM  FindBasWinPars:
21 IA  w&=PEEKL(s&+4)
22 GA  l0:
23 9G  t&=PEEKL(w&+32)
24 60  t$="":i%=0
25 MH  l1:
26 A7  a%=PEEK(t&+i%)
27 YX  IF a%=0 GOTO l2
28 tD  t$=t$+CHR$(a%)
29 UW  i%=i%+1
30 Rv  GOTO l1
31 VR  l2:
32 Dd  IF t$="LIST" THEN
33 hy2   wList&=w&
34 nZ0 ' Fenster=Programmfenster?
35 hA  ELSEIF t$="Backdrop_Window" THEN  't$=Programmname
36 q42   wPrg&=w&
37 XQ0 END IF
38 qx  w&=PEEKL(w&)
39 td  IF w&<>0 GOTO l0 ELSE RETURN
40 QK  '------------------------
41 C1  Specials2:
42 2k  WINDOW CLOSE 4
43 aa  CALL SizeWindow(wPrg&,-dx1%,-dy1%)
44 mT  CALL MoveWindow(wList&,-dx3%,-dy3%)
45 qZ  CALL SizeWindow(wList&,-dx2%,-dy2%)
46 6i  RETURN
47 H1  '------------------------
48 OE  Specials3:
49 9l  RETURN
(C) 1988 M&T
```

Listing 21. Sie können Ihr Fenster auch stets im Hintergrund halten

Die Screen-Struktur

&ptr	NextScreen;	
&ptr	FirstWindow;	erstes Fenster im Scren
%	LeftEdge, TopEdge;	
%	Width, Height;	
%	MouseY, MouseX;	MausPos im Screen
%	Flags;	
&ptr	Title;	
&ptr	DefaultTitle;	Wenn kein Windoweigenner Titel vorhanden ist
Größen der Titelleisten für den Screen und alle darin geöffneten Fenster		
BYTE	BarH, BarVBorder, BarHBorder, MenuVBorder, MenuHBorder;	
BYTE	WBorTop, WBorLeft, WBorRight, WBorBottom;	
&ptr	Font;	default font
struct	ViewPort ViewPort;	dies ist eine eigene Struktur!
struct	RastPort RastPort;	Diese auch!
struct	BitMap BitMap;	noch eine!
struct	Layer_Info LayerInfo;	die letzte
&ptr	FirstGadget;	
UBYTE	DetailPen, BlockPen;	
%	SaveColor0;	für DisplayBeep()
&ptr	BarLayer;	
&ptr	ExtData;	
&ptr	UserData;	

Flags, die Intuition setzt:

Name	Wert	Funktion
WBENCHSCREEN	0x0001	Workbench-Screen oder Kopie davon
CUSTOMSCREEN	0x000F	Screen mit eigenem Design
SHOWTITLE	0x0010	wird von ShowTitle-Funktion gesetzt*/
BEEPING	0x0020	gesetzt bei DisplayBeep
CUSTOMBITMAP	0x0040	Wenn eigene BitMap existiert

Flags, die Sie selbst setzen:

SCREENBEHIND	0x0080	Screen im Hintergrund
SCREENQUIET	0x0100	

Eine wichtige Konstante:

STDSCREENHEIGHT	-1	sollten Sie in Height in der NewScreen-Struktur übergeben

Tabelle 5. Diese Informationen sind in der Screen-Struktur abgelegt

```
 1 Mu0 SUB NewScreen (scr&,LeftEdge%, TopEdge%, xWidth%, Height%, D
       epth%, DetailPen%, BlockPen%, Mode$, Type$, Font&, DefaultTit
       le$, Gadgets&, CustomBitMap&) STATIC
 2 oB  DIM NewScr%(15)
 3 g0  DefaultTitle$=DefaultTitle$+CHR$(0)
 4 8Q  tit&=AllocRemember&(0, LEN(DefaultTitle$), 65540&)
 5 YX  CALL CopyMem(SADD(DefaultTitle$), tit&, LEN(DefaultTitle$))
 6 LW  ' String fuer ViewModes aufbereiten
 7 3j  ' (wie bei Lattice-C)
 8 PI  ViewModes%=0
 9 iS  IF Mode$="" GOTO f1
10 Ys  RESTORE ViewDat
11 zz  i&=0
12 On  WHILE i&<>-1
13 Bn2   READ a$,i&
14 Nb    IF INSTR(Mode$, a$)<>0 THEN
15 Kq4     ViewModes&=ViewModes&+i&
16 C52   END IF
17 iW0 WEND
18 3s  f1:
19 XQ  ' String fuer Screen-Typ aufbereiten
20 Gw  ' (wie bei Lattice-C)
21 FK  Type%=0
22 3s  IF Type$="" THEN Type%=1:GOTO f2
23 dm  RESTORE TypeDat
24 CC  i&=0
25 D0  WHILE i&<>-1
26 O02   READ a$,i&
27 AI    IF INSTR(Type$, a$)<>0 THEN
28 rK4     Type%=Type%+i&
29 PI2   END IF
30 vj0 WEND
31 J9  f2:
32 bG  ' NewScreen - Struktur erzeugen
33 X8  NewScr%(0)=LeftEdge%
34 IC  NewScr%(1)=TopEdge%
35 3D  NewScr%(2)=xWidth%
36 De  NewScr%(3)=Height%
37 n6  NewScr%(4)=Depth%
38 rX  POKE VARPTR(NewScr%(5)), DetailPen%
39 jx  POKE VARPTR(NewScr%(5))+1, BlockPen%
40 1F  POKEW VARPTR(NewScr%(6)),ViewModes&  'der einfachste Weg, da
       s MSB ohne Overflow Error zu uebergeben
41 J3  NewScr%(7)=Type%
42 xZ  POKEL VARPTR(NewScr%(8)),Font&
43 sG  POKEL VARPTR(NewScr%(10)),tit&
44 ST  POKEL VARPTR(NewScr%(12)),Gadgets&
45 lY  POKEL VARPTR(NewScr%(14)),CustomBitMap&
46 N4  ' Screen oeffnen und Speicher freigeben
47 VU  scr&=OpenScreen&(VARPTR(NewScr%(0)))
48 I7  ERASE NewScr%
49 wA  EXIT SUB
50 sw  ViewDat:
51 z3  DATA GENLOCK_VIDEO, 2, LACE, 4, PFBA, 64
52 e0  DATA EXTRA_HALFBRITE, 128, GENLOCK_AUDIO, 256
53 et  DATA DUALPF, 1024, HAM, 2048, VP_HIDE, 8192
54 J1  DATA SPRITES, 16384, HIRES, 32768, X, -1
55 v6  TypeDat:
56 6m  DATA WBENCHSCREEN, 1, CUSTOMSCREEN, 15
57 1B  DATA SHOWTITLE, 16, CUSTOMBITMAP, 64
58 PV  DATA SCREENBEHIND, 128, SCREENQUIET, 256, X, -1
59 z1  END SUB
(C) 1988 M&T
```

Listing 22. Dieses Unterprogramm binden Sie in alle Programme ein, die einen eigenen Screen öffnen

```
 1 1i0 SUB xIText(IntuiText&, FrontPen%, BackPen%, Draw$, LeftEdge%
       , TopEdge%, Font&, Text$, NextText&) STATIC
 2 KF  SHARED IText&
 3 I0  Text$=Text$+CHR$(0)
 4 aq  IText&    =AllocRemember&(0, LEN(Text$), 65540&)
 5 gb  IntuiText&=AllocRemember&(0, 20        , 65540&)
 6 0i  FOR i%=1 TO LEN(Text$)
 7 132   POKE IText&+i%-1, ASC(MID$(Text$,i%,1))
 8 DI0 NEXT
 9 tn  RESTORE Drawdat
10 80  DrawMode%=0:i$=""
11 Wl  WHILE i$<>"X"
12 K82   READ i$,i%
13 IB    IF INSTR(Draw$,i$)<>0 THEN DrawMode%=DrawMode%+i%
14 fT0 WEND
15 3T  POKE  IntuiText&   , FrontPen%
16 U4  POKE  IntuiText&+ 1, BackPen%
17 Av  POKE  IntuiText&+ 2, DrawMode%
18 ju  POKEW IntuiText&+ 4, LeftEdge%
19 Z4  POKEW IntuiText&+ 6, TopEdge%
20 6i  POKEL IntuiText&+ 8, Font&
21 bx  POKEL IntuiText&+12, IText&
22 Th  POKEL IntuiText&+16, NextText&
23 Wk  EXIT SUB
24 9w  Drawdat:
25 oa  DATA JAM1, 0, JAM2, 1, COMPLEMENT, 2, INVERSEVID, 4, X, 0
26 SU  END SUB
(C) 1988 M&T
```

Listing 23. Diese Routine übernimmt den Aufbau einer »IntuiText«-Struktur

```
 1 1T0 ON BREAK GOSUB brk
 2 0D  BREAK ON
 3 PL  DECLARE FUNCTION OpenScreen& LIBRARY
 4 Vg  DECLARE FUNCTION AllocRemember& LIBRARY
 5 Da  LIBRARY"intuition.library"
 6 3K  LIBRARY"graphics.library"
 7 fR  LIBRARY"exec.library"
 8 Ir  GOSUB Specials
 9 Ya  WHILE INKEY$="":SLEEP:WEND
10 CD  brk:
11 bC  CALL CloseScreen(s&)
12 Lv  CALL FreeRemember(0,-1)
13 xs  END
(C) 1988 M&T
```

Listing 24. Dieses Programm verwenden Sie zur Steuerung der Screen-Demos

```
 1 hK0 ' Screen-Demo 1
 2 hK  Specials:
 3 OG  NewScreen s&,0,30,640,256,2,1,3,"HIRES","",0&,"Mein Screen",
       0&,0&
 4 E2  rp&=s&+84
 5 i8  CALL SetRast(rp&, 3)
 6 pg  CALL ShowTitle(s&, 1)
 7 bx  xIText itxt&,1,2,"JAM2",200,80,0&,"...Schrift ohne Fenster",
       0&
 8 WD  FOR i%=0 TO 100 STEP 6
 9 902   CALL PrintIText(rp&, itxt&, i%, i%)
10 FK0 NEXT
11 X9  RETURN
(C) 1988 M&T
```

Listing 25. Man braucht nicht unbedingt ein Fenster für die Textausgabe

```
 1 DN0 'HAMdemo 1
 2 hK  Specials:
 3 QG  NewScreen s&,0,0,320,256,6,0,1,"HAM","CUSTOMSCREEN",0&,"",0&
       ,0&
 4 6u  vp&=s&+44
 5 F3  rp&=s&+84
 6 6S  CALL SetRGB4(vp&,0,0,0,0)
 7 cW  CALL SetDrMd(rp&,0)
 8 jR  FOR r%=0 TO 15
 9 g6  FOR g%=0 TO 15
10 Dx  FOR b%=0 TO 15
11 S82   x%=r%*20
12 Lw    y%=g%+16*b%
13 V8    CALL SetAPen(rp&,r%+32)
14 C0    CALL WritePixel(rp&, x%, y%)
15 xT    x%=x%+1
16 Aj    CALL SetAPen(rp&,g%+48)
17 F3    CALL WritePixel(rp&, x%, y%)
18 0W    x%=x%+1
19 un    CALL SetAPen(rp&, b%+16)
20 X7    CALL Move(rp&, x%, y%)
21 E5    CALL Draw(rp&, x%+17, y%)
22 dW0 NEXT:NEXT:NEXT
23 jL  RETURN
(C) 1988 M&T
```

Listing 26. Auch von Basic aus können Sie 4096 Farben darstellen

Die IntuiText-Struktur

UBYTE	FrontPen, BackPen;	Farben
UBYTE	DrawMode;	JAM1, JAM2 usw.
%	LeftEdge;	Offset links
%	TopEdge;	Offset rechts
&ptr	ITextFont;	NULL, wenn Default gewünscht
&ptr	IText;	Zeiger auf Text (0-terminiert)
&ptr	NextText;	Zur Verkettung

Tabelle 6. Die IntuiText-Struktur wird für die vereinfachte Textausgabe mit Intuition verwendet

kehren. Ähnlich wie bei den Windows stehen Ihnen auch hier noch einige Gestaltungsmöglichkeiten offen, wenn Sie die Organisation selbst in die Hand nehmen.

Listing 22 ist die Routine zum Öffnen eines eigenen Screens. Der Aufbau des Programms ist der gleiche wie der von Listing 17 bekannte. Hier sollen nur kurz die neuen Parameter erklärt werden, soweit sie nicht von Basic her bekannt sind.

Als erstes fällt hier auf, daß man jedem Screen einen Titel geben kann, der immer dann angezeigt wird, wenn kein Fenster-spezifischer Screentitel existiert (dieser wird mit »SetWindowTitles« festgelegt, als Anwendungsbeispiel sehen Sie sich am besten Listing 11 an). Daß diese Möglichkeit in Basic nicht geboten wird, ist sehr schade.

Die Darstellungsmodi (»View Dat:«) gehen auch um einiges über das hinaus, was Basic bietet. So können wir ohne Probleme den »EXTRA_HALF BRITE«- und den »HAM«-Modus verwenden, und wir können die »GENLOCK«- sowie die »Dual-Playfield«-Darstellung aktivieren (wenn Sie diese Möglichkeiten nutzen wollen, sollten Sie den Grafik-Kurs in AMIGA-Magazin, Ausgaben 12/87 bis 3/88 lesen). Es ist auch möglich, durch Setzen des Flags »SCREENBEHIND« den ganzen Bildschirm mit allem, was dazugehört aufzubauen und dann erst mit »ScreenToFront« zur Anzeige bringen.

Listing 24 ist das sehr einfach gehaltene Steuerprogramm für die Screen-Demos. In Listing 23 wird eine weitere Methode der Textdarstellung vorgestellt: »IntuiText«. Hier ist zwar der Aufbau einer Struktur - der »IntuiText«-Struktur - notwendig (siehe Tabelle 6), aber dafür können alle Parameter der Textdarstellung (Farbe, DrawMode, Font, Position) sehr übersichtlich vom Programm vorbereitet werden, um dann durch einen einzigen Befehlsaufruf im Hauptprogramm mit

```
PrintIText
(rp&, IntuiText&, xoff%,
 yoff%)
```

auf den Bildschirm zu kommen. Wie das Beispielprogramm aus Listing 25 zeigt, enthalten »xoff%« und »yoff%« die Startposition der Bildschirmausgabe relativ zu den in der »IntuiText«-Struktur angegebenen Positionen.

Die »IText«-Routine ist völlig offen gehalten und enthält eine vollständige Parameterliste, so daß sie ohne weiteres in anderen Programmen verwendet werden kann. Übrigens macht auch bei dieser Routine der AC-Basic-Compiler Probleme. Wenn Sie diesen also verwenden wollen, sollten Sie einzelne Parameter fest vorgeben, die dann nicht mehr über die Parameterliste übergeben werden müssen.

Noch zwei neue Befehle tauchen in Listing 25 auf: Mit

```
SetRast(rp&, col%)
```

können Sie einen ganzen RastPort (also Fenster oder Screen) durch einen einzigen Befehl mit der angegebenen Farbe füllen. Allerdings sollten Sie damit vorsichtig umgehen. Auch die Titelzeile und bei Fenstern der Rahmen (außer bei Windows, die das »GIMMEZEROZERO«-Flag gesetzt haben) wird übermalt! Deshalb wird mit

```
ShowTitle(s&, mode&)
```

der Screen-Titel wieder zur Anzeige gebracht. »mode&« ist ein Ein-/Aus-Schalter (bei 0 ist kein Titel zu sehen, bei jedem anderen Wert kommt der Screen-Titel in den Vordergrund). Noch eine Verwendung hat der ShowTitle-Befehl bei Backdrop-Windows: Wenn ein Backdrop-Window die Titelleiste des Screens überlagert, kann mit ShowTitle entschieden werden, ob der Screentitel vor oder hinter dem Backdrop-Window ist. Ein »BORDERLESS-BACKDROP«-Window in Maximalgröße mit

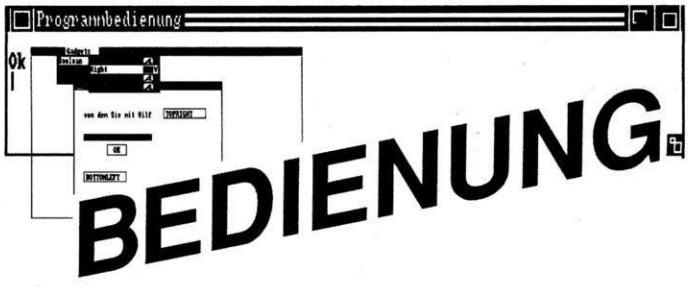
```
ShowTitle(s&, 0)
```

liefert also einen völlig leeren Bildschirm.

Das frappierendste an diesem Programm ist Ihnen aber hoffentlich nicht entgangen: es gibt in diesem Screen kein Fenster!

Das ist für Basic-Programmierer etwas völlig Neues, denn dort können wir nur in Fenster schreiben und zeichnen. Das System der Grafikausgabe orientiert sich aber lediglich an RastPorts, und die sind für Screens und Fenster prinzipiell gleich. Es hindert uns überhaupt nichts daran — ausgenommen der Basic-Interpreter — in Screens ohne Fenster zu schreiben und zu malen, was wir wollen. Beispiele dafür kennen Sie von der Workbench (Disketten-Icons) und vom Texteditor MicroEMACS (auf der Extras_1.2-Diskette), der ebenfalls einen Screen für die Textausgabe verwendet.

Zu guter Letzt finden wir in Listing 26 den Höhepunkt an Farbenvielfalt. Wir aktivieren in diesem Basic-Programm den HAM-Modus und zeigen alle 4096 Farben auf dem Bildschirm (wieder ohne Fenster). Das Programm ist übrigens eine Umsetzung eines C-Programmes aus dem Amiga Programmier-Handbuch von Jörg Koch und Frank Kremser.



Im letzten Teil unseres Kurses dreht sich alles um die Hilfsmittel für die Programmbedienung, die das Betriebssystem zur Verfügung stellt. Wir lernen, wie man Gadgets erstellt und einsetzt, wie man Requester und Alerts programmiert. Schließlich wird auch gezeigt, wie von Basic aus Menüs mit Untermenüs eingerichtet werden können.

Leider werden diese Möglichkeiten von Amiga-Basic nicht im geringsten unterstützt. Vom Interpreter im Stich gelassen, behelfen sich die meisten Basic-Programmierer mit dem Zeichnen irgendwelcher Kästchen sowie umständlicher (und langsamer) Mausabfragen.

Mit der Gestaltung von Menüs ist es beim Amiga-Basic-Interpreter auch nicht weit her. Das Setzen von Checkmarks (Häkchen bei Anwahl des Menüs) ist schon das allerhöchste der Gefühle. Obwohl das Betriebssystem maximal 32 Menütitel mit je maximal 64 Menüpunkten erlaubt, kann man in Basic maximal 10 Menüs mit maximal 19 Menüpunkten verwenden. Das ist noch nicht so schlimm – wer verwendet schon so viele Menüs –, aber schon bei den Untermenüs (im folgenden »Subitems« genannt) hört der Komfort auf: Es gibt sie nicht. Ein ebenso schwerwiegender Nachteil ist, daß man keine Tastaturkürzel angeben kann (Taste in Verbindung mir rechter Amiga-Taste).

All diese Möglichkeiten sind aber im Betriebssystem vorgesehen und sogar teilweise genormt; so sind beispielsweise die eigenen Images für das Checkmark und die Darstellung der rechten Amiga-Taste bereits im Betriebssystem vorhanden.

Wir wollen Sie in die Lage versetzen, auch das letzte Quentchen Leistung aus Ihrem Amiga-Basic herauszuholen und all diese freundlichen Hilfsmittel zu nutzen. Machen Sie sich auf interessante Listings und viel »POKErei« gefaßt. Als Lohn der vielen Tipparbeit bekommen Sie eine ganze Reihe von Unterprogrammen, die Sie unverändert und mit immensem Gewinn in eigenen Programmen einsetzen können.

Im letzten Abschnitt werden zunächst die einzelnen Module vorgestellt und erklärt, bevor

mit dem Demo-Programm eine Anwendung gezeigt wird, in der alle Bedienungselemente verwendet werden. Die Erklärungen hierzu sind nicht mehr ganz so ausführlich wie in den vorhergehenden Abschnitten, weil die notwendigen Grundlagen dort bereits ausreichend besprochen wurden.

Sofern in den Bedienungselementen Text vorkommt (mit Ausnahme des Alerts — siehe unten), muß dieser in Form einer IntuiText-Struktur vorliegen.

Um die Möglichkeiten des Betriebssystems zu nutzen, ist der Aufbau von Strukturen in größerer Anzahl nötig. Den Speicherplatz dafür holen wir uns vom Betriebssystem. So kann uns der Basic-Interpreter nicht ins Handwerk pfuschen und der Speicherplatz wird auch ökonomischer verwaltet.

Alarm!

Nicht als Bedienungselement im engeren Sinne kann man die Alerts bezeichnen. Diese Meldungen sind vielmehr dazu gedacht, den Benutzer von einem sehr schwierigen oder unbehebbaren Problem des Programms zu informieren (prominentes Beispiel: der »Guru«). Wenden Sie Alerts nur an, wenn Sie den Benutzer von einer solchen Situation informieren wollen, nicht um ihn unnötig zu erschrecken!

Das Unterprogramm in Listing 27 erzeugt wahlweise einen »Deadend«- oder einen »Recoverable«-Alert. »Deadend« heißt Sackgasse und in der steckt Ihr Programm nach diesem Alert. Von hier gibt es keine Rückkehr! Dagegen ist der »Recoverable«-Alert für eine Situation geeignet, in der eine Rückkehr möglich ist. In der Regel werden Sie mit der zweiten Variante auskommen und das Programm beenden, nachdem der Benutzer eine Maustaste betätigt hat. Sie müssen zwei Felder vorbereiten, von denen eines alle Textzeilen enthält, die Sie im Alert darstellen wollen, das zweite die y-Koordinaten der Textzeilen. in »Align$« geben sie »LEFT«, »RIGHT« oder »CENTER« an. »Height%« ist die Höhe der roten Box. »type$« erzeugt einen Deadend-Alert, wenn sie »DEADEND« angeben; jeder andere Text in »type$« führt zu Recoverable-Alerts.

Die Aufrufparameter entnehmen Sie bitte wieder den Listings.

Aufforderung zum Handeln: Requester

Requester wenden Sie überall dort an, wo das Programm unbedingt eine Benutzer-Eingabe erfordert; sei es eine Entscheidung (ja/nein) oder die Eingabe von Texten (z.B. Filenamen). Hier ist der Requester das geeignetste Instrument, da er einfach zu programmieren ist, auffällt und nur mit einer Eingabe wieder verlassen werden kann (im Gegensatz zum »INPUT«-Befehl). Die einfachste Form ist hier der »AutoRequester«. Er ermöglicht nur eine binäre Entscheidung (j/n). Als Beispiel sei hier der »no disk present in unit x«-Requester des Betriebssystems genannt. Dieser kann nur mit »Retry« oder »Cancel« verlassen werden. Das Unterprogramm in Listing 28 dient zur Erzeugung eines AutoRequesters und meldet dem aufrufenden Programm, welches Gadget angeklickt wurde (0=rechtes Gadget). Verwendet wird dazu die »AutoRequest«-Funktion der »intuition.library«, die die Zeiger auf drei »IntuiText«-Strukturen benötigt: je eine für linkes und rechtes Gadget und eine für den Text im Requester. Um mehrere Zeilen Text auszugeben, wird ein String-Array übergeben (»title$()«), das in mehrere verkettete »IntuiText«-Strukturen abgelegt wird. Die übrigen Parameter dürften keine Probleme bereiten. Das Programm geht übrigens davon aus, daß Sie nur einen AutoRequester verwenden. Um Speicherplatz zu sparen, wird die einmal aufgebaute Struktur weiterverwendet. Sollten Ihre Erfordernisse anders sein, ändern Sie das Programm an den im Listing angegebenen Stellen.

Etwas komplizierter wird es beim »Custom«-Requester. Dort können (und müssen) Sie alles selbst aufbauen — also auch die Gadgets und die Umrandung. Dafür können sie mit diesen Requestern alles mögliche abfragen; seien es Zahlen, Texte oder Entscheidungen zwischen mehreren Auswahlmöglichkeiten. Wichtig ist nur, daß mindestens eines der verwendeten Gadgets das »ENDGADGET«-Flag (siehe unten) gesetzt hat. Dies führt dazu, daß beim Betätigen dieses Gadgets der Requester verschwindet. Der Effekt bei fehlendem »EndGadget« wäre ein

```
 1 PwO SUB xAlert(anz%,t$(),y%(),Align$,type$,Height%) STATIC
 2 o3  anz%=anz%-1
 3 BN  IF type$="DEADEND"THEN type%=1 ELSE type%=0
 4 hP  al$=""
 5 Nd  FOR i%=0 TO anz%
 6 cl2   IF Align$="LEFT" THEN
 7 Ma4     x%=0
 8 Bz2   ELSEIF Align$="RIGHT" THEN
 9 Vo4     x%=640-8*LEN(t$(i%)
10 zK2   ELSEIF Align$="CENTER" THEN
11 iS4     x%=320-4*LEN(t$(i%))
12 812   END IF
13 01    al$=al$+MKI$(x%)+CHR$(y%(i%))+t$(i%)+CHR$(0)
14 ds    IF i%<anz% THEN al$=al$+CHR$(1) ELSE al$=al$+CHR$(0)
15 KPO NEXT
16 KO  CALL DisplayAlert(type%,SADD(al$),Height%)
17 JL  END SUB
(C) 1988 M&T
```

Listing 27. Erzeugen Sie Ihre eigenen Alarmmeldungen

»Endlos-Requester«, der nicht mehr verlassen werden kann und keine Aktion im Fenster mehr zuläßt.

In Listing 29 finden Sie ein Unterprogramm, das den Aufbau einer Requester-Struktur (Tabelle 7) für Sie übernimmt, deren Adresse in »req&« zurückgegeben wird. Für das Darstellen des Requesters ist das Hauptprogramm zuständig. Einige Parameter möchte ich noch erklären: »RelLeft« und »RelTop« geben an, wo »relativ« zur momentanen Mausposition der Requester erscheinen soll, wenn Sie das »POINTREL«-Flag gesetzt haben. »BackFill« nennt die Farbnummer, die für den Hintergrund verwendet wird. »ReqGadget«, »ReqBorder« und »ReqText« sind Zeiger auf jeweils die erste Struktur verketteter Listen von Gadget-, Border- beziehungsweise IntuiText-Strukturen.

Die IntuiText-Struktur kennen wir; fehlen also noch die Gadget- und die Border-Struktur:

```
 1 wQO SUB xAutoReq(flg%,win&,anz%,title$(),PosText$,NegText$,xWidt
       h%,Height%) STATIC
 2 Dx  IF arq1set%=1 GOTO arq
 3 Lr  arq1set%=1
 4 Qh  FOR i%=1 TO anz%
 5 9D1  xIText tit&,3,0,"JAM1",10,10*i%,0&,title$(i%-1)+CHR$(0),t&
 6 op   t&=tit&
 7 CHO NEXT
 8 f9  xIText postxt&,3,0,"JAM1",7,3,0&,PosText$+CHR$(0),0&
 9 4Y  xIText negtxt&,3,0,"JAM1",7,3,0&,NegText$+CHR$(0),0&
10 Ye  arq:
11 V2  flg%=AutoRequest%(win&,tit&,postxt&,negtxt&,0,0,xWidth%,Heig
       ht%)
12 EG  END SUB
(C) 1988 M&T
```

Listing 28. Diese Routine übernimmt das Erzeugen eines »AutoRequesters«

```
 1 ZFO SUB xReq(req&,LeftEdge%,TopEdge%,xWidth%,Height%,RelLeft%,Re
       lTop%,ReqGadget&,ReqBorder&,ReqText&,flags$,BackFill%) STATIC
 2 nv  IF flags$="POINTREL" THEN flags%=1 ELSE flags%=0
 3 8h  req&=AllocRemember&(0,112,65540&)
 4 0a  CALL InitRequester(req&)
 5 wZ  POKEW req&+ 4,LeftEdge%
 6 wo  POKEW req&+ 6,TopEdge%
 7 XC  POKEW req&+ 8,xWidth%
 8 3G  POKEW req&+10,Height%
 9 1B  POKEW req&+12,RelLeft%
10 P9  POKEW req&+14,RelTop%
11 yV  POKEL req&+16,ReqGadget&
12 aC  POKEL req&+20,ReqBorder&
13 Uj  POKEL req&+24,ReqText&
14 X8  POKEW req&+28,flags%
15 gN  POKE  req&+30,BackFill%
16 IK  END SUB
(C) 1988 M&T
```

Listing 29. Diese Routine verwenden Sie für einen »Customrequester«

Die Requester-Struktur

&ptr	OlderRequest;	zur Verkettung
%	LeftEdge, TopEdge;	
%	Width, Height;	
%	RelLeft, RelTop;	Wenn POINTREL gesetzt
&ptr	ReqGadget;	Zeiger auf Gadgetliste
&ptr	ReqBorder;	Zeiger auf Borderstruktur
&ptr	ReqText;	IntuiText-Struktur
%	Flags;	
UBYTE	BackFill;	Farbe für Hintergrund
&ptr	ReqLayer;	
UBYTE	ReqPad1(32);	
&ptr	ImageBMap;	wenn eigene BITMAP
&ptr	RWindow;	Zeigt auf Window des Req.
UBYTE	ReqPad2(36);	

Mögliche Werte für »Flags«:

POINTREL	0x0001	linke obere Ecke relativ zum Mauszeiger
PREDRAWN	0x0002	Bild wird für den Requester übernommen
NOISYREQ	0x0004	kein Input-Filter

Diese Flags werden nur von Intuition verwendet:

REQOFFWINDOW	0x1000	
REQACTIVE		0x2000
SYSREQUEST	0x4000	

Tabelle 7. Aus diesen Komponenten besteht eine Requester-Struktur

Die Gadget-Struktur

&ptr	NextGadget;	
%	LeftEdge, TopEdge;	
%	Width, Height;	
%	Flags;	
%	Activation;	
%	GadgetType;	
&ptr	GadgetRender;	entweder Border oder Image
&ptr	SelectRender;	andere Border od. Image
&ptr	GadgetText;	Zeiger auf IntuiText
&	MutualExclude;	gesetzt: dieses und jenes nicht zugleich möglich
&ptr	SpecialInfo;	für String u. Propgadgets
%	GadgetID;	Kennummer des Gadgets
&ptr	UserData;	

Diese Flags können Sie setzen:

GADGHCOMP	0x0000	Select-Box wird komplementiert
GADGHBOX	0x0001	Gadget wird eingerahmt
GADGHIMAGE	0x0002	zweites Bild wird gezeichnet
GADGHNONE	0x0003	nichts geschieht
GADGIMAGE	0x0004	setzen, wenn Image statt Border verwendet wird
GRELBOTTOM	0x0008	relative Koordinaten zu unterem Rand (negativ angeben)
GRELRIGHT	0x0010	relative Koordinaten zu rechtem Rand (negativ angeben)
GRELWIDTH	0x0020	
GRELHEIGHT	0x0040	
SELECTED	0x0080	ist bei erster Darstellung aktiv
GADGDISABLED	0x0100	wird später mit On/OffGadget gesetzt

Diese Bits werden interessant, wenn das Gadget aktiviert ist:

RELVERIFY	0x0001	Überprüfen, ob über Gadget losgelassen
GADGIMMEDIATE	0x0002	
ENDGADGET	0x0004	dieses Gadget beendet Requester
FOLLOWMOUSE	0x0008	
RIGHTBORDER	0x0010	Gadget wird in Border eingefügt
LEFTBORDER	0x0020	
TOPBORDER		0x0040
BOTTOMBORDER	0x0080	
TOGGLESELECT	0x0100	Ein/aus-Gadget (Wechselschalter)
STRINGCENTER	0x0200	Darstellung bei StringGadget
STRINGRIGHT	0x0400	
LONGINT	0x0800	
ALTKEYMAP	0x1000	String verwendet eine andere als die Standardkeymap
BOOLEXTEND	0x2000	
Flags, die sich auf den Gadgettyp beziehen:		
SYSGADGET	0x8000	1 = SysGadget, 0 = AppliGadget
SCRGADGET	0x4000	1 = ScreenGadget, 0 = WindowGadget
GZZGADGET	0x2000	1 = Gadget for GZZ borders
REQGADGET	0x1000	1 = this is a Requester Gadget

System-Gadgets:

SIZING	0x0010
WDRAGGING	0x0020
SDRAGGING	0x0030
WUPFRONT	0x0040
SUPFRONT	0x0050
WDOWNBACK	0x0060
SDOWNBACK	0x0070
CLOSE	0x0080

Anwender-Gadgets:

BOOLGADGET	0x0001
GADGET0002	0x0002
PROPGADGET	0x0003
STRGADGET	0x0004

Tabelle 8. Bei Gadgets wird viel mit Flags gearbeitet

Klick mich an!

Listing 30 ist ein Unterprogramm für das Einrichten einer Gadget-Struktur (Tabelle 8). Es liefert in »gad&« die Adresse der Struktur an das aufrufende Programm zurück. Gemäß der Vielfalt an Möglichkeiten hat diese Routine wieder eine sehr lange Parameterliste. Der Aufbau des Programms ist so einfach, daß ich mir hier jede Erklärung sparen kann. Die wichtigsten Flags sind in Tabelle 8 beschrieben. Wieder seien nur einige der Übergabeparameter erklärt: »NextGadget« ist ein Zeiger auf eine Gadget-Struktur und dient zum Verketten mehrerer Gadgets. »Type« ist üblicherweise »BOOLGADGET«, »PROPGADGET« (z.B. für Farbregler) oder »STRGADGET« (zur Texteingabe). Beim »STRGADGET« können Sie noch zusätzlich das »LONGINT-Flag« setzen; es können dann nur Ziffern und Plus- oder Minuszeichen eingegeben werden.

Bitte achten Sie bei allen Flags auf die Großschreibung! Ich habe das in Anlehnung an die C-Konventionen beibehalten, wo Flags grundsätzlich großgeschrieben werden.

»GadgetRender« und »SelectRender« sind üblicherweise Zeiger auf Border-Strukturen; außer wenn das »GADGIMAGE«-Flag gesetzt ist, dann steht dort der Zeiger auf eine Image-Struktur (diese wird im Kurs nicht erklärt). »GadgetText« zeigt auf eine IntuiText-Struktur. Durch Setzen von Bits in »MutualExclude« können sie angeben, welche Gadgetnummern deaktiviert werden sollen, wenn Ihr Gadget angeklickt wird. »SpecialInfo« wird bei String- und Proportional-Gadgets verwendet und zeigt entsprechend auf eine StringInfo- oder eine PropInfo-Struktur (siehe unten). An der »ID« können Sie später erkennen, daß dieses Gadget angeklickt wurde.

Als nächstes kommt in Listing 31 das Erzeugen einer Border-Struktur (Adresse wird in »bor&« geliefert). Von den Parametern muß nur »XY&« erklärt werden. Es ist der Zeiger auf eine Liste von Koordinaten für die Eckpunkte, die wir am besten in geschütztem Speicher anlegen. Beachten Sie, daß für jedes Rechteck fünf Koordinatenpaare benötigt werden; eines für den Startpunkt und je eines für die Bewegung zu den nächsten Ecken - also auch zum Startpunkt zurück. Eine »Border« kann jede beliebige Anzahl an Eckpunkten und beliebige Form haben.

```
 1 b70 SUB xGadget(gad&, NextGadget&, LeftEdge%, TopEdge%, xWidth%,
       Height%, flags$, Acti$, type$, GadgetRender&, SelectRender&,
       GadgetText&, MutualExclude&, SpecialInfo&, id%, UserData&)
       STATIC
 2 Jz  gad&=AllocRemember&(0, 44, 65540&)
 3 Nf  RESTORE GFlagdat
 4 Xq  flags%=0:i$=""
 5 Qf  WHILE i$<>"X"
 6 E22   READ i$,i%
 7 j4    IF INSTR(flags$,i$)<>0 THEN flags%=flags%+i%
 8 ZNO WEND
 9 4S  RESTORE ActiDat
10 TO  Activation%=0:i$=""
11 Wl  WHILE i$<>"X"
12 K82   READ i$,i%
13 6p    IF INSTR(Acti$,i$)<>0 THEN Activation%=Activation%+i%
14 fTO WEND
15 Ve  RESTORE TypeDat
16 8B  GadgetType%=0:i$=""
17 cr  WHILE i$<>"X"
18 QE2   READ i$,i%
19 8c    IF INSTR(type$,i$)<>0 THEN GadgetType%=GadgetType%+i%
20 lZ0 WEND
21 4p  POKEL gad&   ,NextGadget&
22 NY  POKEW gad&+ 4,LeftEdge%
23 Nn  POKEW gad&+ 6,TopEdge%
24 yB  POKEW gad&+ 8,xWidth%
25 UF  POKEW gad&+10,Height%
26 SU  POKEW gad&+12,flags%
27 wV  POKEW gad&+14,Activation%
28 FF  POKEW gad&+16,GadgetType%
29 Dq  POKEL gad&+18,GadgetRender&
30 HB  POKEL gad&+22,SelectRender&
31 8q  POKEL gad&+26,GadgetText&
32 cv  POKEL gad&+30,MutualExclude&
33 uF  POKEL gad&+34,SpecialInfo&
34 R9  POKEW gad&+38,id%
35 IN  POKEL gad&+40,UserData&
36 jx  EXIT SUB
37 ZQ  GFlagdat:
38 lj  DATA GADGHCOMP, 0, GADGHBOX, 1, GADGHIMAGE, 2, GADGHNONE, 3
39 SY  DATA GADGIMAGE, 4, GRELBOTTOM, 8, GRELRIGHT, 16, GRELWIDTH,
       32
40 qj  DATA GRELHEIGHT, 64, SELECTED, 128, GADGDISABLED, 256, X, 0
41 bF  ActiDat:
42 xP  DATA RELVERIFY, 1, GADGIMMEDIATE, 2, ENDGADGET, 4, FOLLOWMOU
       SE, 8
43 lb  DATA RIGHTBORDER, 16, LEFTBORDER, 32, TOPBORDER, 64, BOTTOMB
       ORDER, 128
44 ou  DATA TOGGLESELECT, 256, STRINGCENTER, 512, STRINGRIGHT, 1024
45 7J  DATA LONGINT, 2048, ALTKEYMAP, 4096, BOOLEXTEND, 8192, X, 0
46 mx  TypeDat:
47 2y  DATA BOOLGADGET, 1, GADGET0002, 2, PROPGADGET, 3, STRGADGET,
        4
48 U4  DATA REQGADGET, 4096, GZZGADGET, 8192, SCRGADGET, 16384, X,
       0
49 pr  END SUB
(C) 1988 M&T
```

Listing 30. So definieren Sie Ihre eigenen Gadgets

Für ein String-Gadget benötigen wir noch die »StringInfo«-Struktur. Die dafür nötige Routine steht in Listing 32. Sie liefert drei Werte an das aufrufende Programm zurück: einen Zeiger auf die StringInfo-Struktur, einen Zeiger auf den Textpuffer (»buf&«) und einen Zeiger auf den Undo-Puffer (»UndoBuf&«), wo der Text seit dem letzten Betätigen der Return-Taste gespeichert ist und mit <rechte Amiga-Taste Q> zurückgeholt werden kann. Als weitere Parameter müssen noch übergeben werden:

»BufPos%«: markiert die Stelle wo im Puffer der Cursor stehen soll; »DispPos%«: bezeichnet das erste dargestellte Zeichen (der Text kann länger sein als das Gadget breit); »MaxChars%«: wie lange soll der Text maximal sein dürfen, und schließlich »LongInt&«: bei Longint-Gadgets steht dort jederzeit abrufbar die eingegebene Zahl im 4-Byte-Format — auch eine eventuelle Vorgabe kann dort beim Aufbau der Struktur abgelegt werden.

Proportional-Gadgets verlangen in »SpecialInfo« eine »PropInfo«-Struktur. Listing 33 hilft beim Aufbau dieser Struktur; ein Zeiger auf die Struktur wird in »PInfo&« zurückgemeldet. In »flags$« geben Sie an, ob sie den vorbereiteten »Knopf« verwenden wollen, in welche Richtung das Gadget beweglich sein soll und ob eine

Menu-Struktur

&ptr	NextMenu;	
%	LeftEdge, TopEdge;	
%	Width, Height;	
%	Flags;	
&ptr	MenuName;	Zeiger auf IntuiText
&ptr	FirstItem;	Zeiger auf MenuItem
%	JazzX, JazzY, BeatX, BeatY;	nur intern verwendet
Dieses Flag kann von Intuition oder Ihrem Programm gesetzt werden:		
MENU ENABLED	0x0001	zeigt an, ob dieses Menü wählbar ist

Tabelle 9. Menüs werden mit verbundenen Listen verwaltet

Border gezeichnet werden soll.

Die »xxxPot«-Werte geben an, welche Stellung der Regler beim Öffnen des Gadgets haben soll. Maximum ist hier hexadezimal &HFFFF (entspricht dezimal 65536&). Wenn Sie beispielsweise 21845 angeben, erscheint der Knopf des Reglers genau 1/3 des Weges vom Start weg. Intuition setzt diese Werte bei Veränderungen neu; sie können ständig abgefragt und so die Bewegung des Reglers ausgewertet werden.

Die »xxxBody«-Werte geben die Größe des Knopfes und damit die Schrittweite des Reglers an, wenn außerhalb des Knopfes im Gadget geklickt wird. Sinnvollerweise geben Sie hier den Wert so (als Bruchteil von 65536&) an, daß die Relation »angezeigte Daten zu vorhandene Daten« gleich »xxxPot zu 65536&« ist.

Übrigens gibt es in den Flags noch ein Bit (Bit 8 = 256), das von Intuition gesetzt wird, wenn der Knopf gerade niedergedrückt ist. Es ermöglicht Ihnen, nach Anklicken des Gadgets die Reglerbewegung zu verarbeiten, während der Knopf noch gedrückt ist.

Der letzte Gang: Menüs

Als letztes kommen die Menüs an die Reihe. Hier benötigen wir zwei Strukturen: Die »Menu«-Struktur für den Menütitel und die »MenuItem«-Struktur für die Menüpunkte (die auch für die »Subitems« verwendet wird). Beide sind als verkettete Liste geführt (siehe Tabelle 9).

Zuerst zu den Menütiteln: in Listing 34 sehen Sie, wie die »Menu«-Struktur aufgebaut ist. Das Programm liefert in »men&« einen Zeiger auf die Struktur. Als erste Komponente finden wir einen Zeiger auf eine weitere »Menu«-Struktur (zur Verkettung). »LeftEdge« und »xWidth« geben den Abstand vom linken Rand und die Breite des Menütitels an. Für »TopEdge« und »Height« gibt es in der derzeitigen Betriebssystemversion keine Verwendung. Möglicherweise wird in späteren Versionen die Möglichkeit offen gehalten, das Me-

```
 1 eXO SUB xBorder (bor&, LeftEdge%, TopEdge%, FrontPen%, BackPen%,
       Draw$, Count%, XY&, NextBor&) STATIC
 2 cp  bor&=AllocRemember&(0&, 16, m&)
 3 uW  RESTORE DMdat
 4 2I  DrawMode%=0:i$=""
 5 Qf  WHILE i$<>"X"
 6 E22   READ i$,i%
 7 C5    IF INSTR(Draw$,i$)<>0 THEN DrawMode%=DrawMode%+i%
 8 ZNO WEND
 9 3o  'Einrichten der Border-structure
10 gT  POKEW bor&   ,LeftEdge%        'relativ zur linken
11 dj  POKEW bor&+ 2,TopEdge%         'oberen Ecke des RastPorts
12 nh  POKE  bor&+ 4,FrontPen%
13 jL  POKE  bor&+ 5,BackPen%
14 3F  POKE  bor&+ 6,DrawMode%
15 qX  POKE  bor&+ 7,Count%           'Anzahl der Eckpunkte
16 UG  POKEL bor&+ 8,XY&              'Zeiger auf Koordinaten-Array
17 Hg  POKEL bor&+12,NextBor&         'Zeiger auf naechste Border
18 Rf  EXIT SUB
19 Fh  DMdat:
20 jV  DATA JAM1, 0, JAM2, 1, COMPLEMENT, 2, INVERSEVID, 4, X, 0
21 NP  END SUB
(C) 1988 M&T
```

Listing 31. Das Anlegen einer »Border«-Struktur vereinfacht »xBorder«

```
 1 qNO SUB xStrInfo(SInfo&, Buf&, UndoBuf&, BufPos%, MaxChars%, Dis
       pPos%, LongInt&) STATIC
 2 TI  Buf&=AllocRemember&(0, MaxChars%+1, 65540&)
 3 OD  UndoBuf&=AllocRemember&(0, MaxChars%+1, 65540&)
 4 H9  SInfo&=AllocRemember&(0, 36, 65540&)
 5 g7  POKEL SInfo&   ,Buf&
 6 vq  POKEL SInfo&+ 4,UndoBuf&
 7 16  POKEW SInfo&+ 8,BufPos%
 8 rp  POKEW SInfo&+10,MaxChars%
 9 I3  POKEW SInfo&+12,DispPos%
10 HO  POKEL SInfo&+28,LongInt&
11 DF  END SUB
(C) 1988 M&T
```

Listing 32. Mit »xStringInfo« erzeugen Sie eine »StringInfo«-Struktur

```
 1 AoO SUB xPropInfo(PInfo&, Flags$, HorizPot%, VertPot%, HorizBody
       %, VertBody%) STATIC
 2 OL  RESTORE PFlgDat
 3 UH  Flags%=0:i$=""
 4 Pe  WHILE i$<>"X"
 5 D12   READ i$,i%
 6 ON    IF INSTR(Flags$,i$)<>0 THEN Flags%=Flags%+i%
 7 YMO WEND
 8 WM  PInfo&=AllocRemember&(0, 22, 65537&)
 9 5p  POKEW PInfo&   ,Flags%
10 4A  POKEW PInfo&+ 2,HorizPot%
11 zG  POKEW PInfo&+ 4,VertPot%
12 op  POKEW PInfo&+ 6,HorizBody%
13 Bk  POKEW PInfo&+ 8,VertBody%
14 Nb  EXIT SUB
15 aq  PFlgDat:
16 9C  DATA AUTOKNOB, 1, FREEHORIZ, 2, FREEVERT, 4
17 qM  DATA PROPBORDERLESS, 8, X, 0
18 KM  END SUB
(C) 1988 M&T
```

Listing 33. »xPropInfo« unterstützt Sie bei Proportional-Gadgets

nü nicht nur in die Titelleiste, sondern an eine beliebige Stelle auf dem Bildschirm zu positionieren. Bei den Flags gibt es nur die Möglichkeit »MENUENABLED« zu setzen; das heißt das Menü ist bei gesetztem Flag aktiv. In »MenuName« wird der Zeiger auf den »IntuiText« für den Titel eingetragen. In »FirstItem« steht ein Zeiger auf den ersten Menüpunkt.

Bei der MenuItem-Struktur (Listing 35; Zeiger ist in Menl enthalten) ist es etwas komplizierter. Die Strukturkomponenten »nextItem« bis »Height« entsprechen der Menu-Struktur. Die Flags sind in Tabelle 9 beschrieben; MutualExclude kennen Sie von der Gadget-Struktur. »ItemFill« und »SelectFill« geben an, was dargestellt werden soll. Wenn Sie das »ITEMTEXT«-Flag gesetzt haben (der Normalfall), steht dort der Zeiger auf eine IntuiText-Struktur; sonst der Zeiger auf eine »Image«-Struktur (z.B. zur Farbauswahl bei Malprogrammen). In »Command« geben Sie den ASCII-Wert der Taste an, die in Verbindung mit der rechten Amiga-Taste den Menüpunkt aktiviert — das funktioniert jedoch nur, wenn Sie dazu auch das »COMMSEQ«-Flag gesetzt haben. In »SubItem« steht schließlich der Zeiger auf die erste »MenuItem«-Struktur einer eventuellen SubItem-Liste. Bei einem Subitem selbst wird dieser Zeiger ignoriert. In »NextSelect« schließlich wird die Nummer des nächsten zu aktivierenden Menüs angegeben, wenn der Anwender mehrere gleichzeitig durch sogenanntes »Dragselecting« anwählt.

Vorerst dürften Sie von neuen Strukturen genug haben. Die wichtigste Frage, die sich nun stellt ist: Wie wende ich das alles an?

Briefkästen mit Signalsystem

Das Multitaskingsystem des Amiga erfordert eine ziemlich komplexe Handhabung des Nachrichtenaustausches zwischen verschiedenen Tasks. Die Hauptrolle dabei spielen die sogenannten MessagePorts, die jedem Task zugeordnet werden können (bis zu 16 je Task). Diese sind für die Übermittlung von Nachrichten zuständig, sie haben in etwa die Funktion von Briefkästen. Nun reicht es für den Amiga aber nicht aus, wenn — wie bei der Post — einmal am Tag geleert wird. Deshalb wird jedesmal, wenn eine Nachricht eintrifft, zusätzlich ein Wecksignal an den betroffenen Task geschickt. Intuition bietet darüber hinaus über die jedem Window zugeordnete »IntuiMessagePort«-Struktur noch weitere Informationen an, wie beispielsweise den exakten Zeitpunkt des Eintreffens, Mauskoordinaten zu dieser Zeit und so weiter (siehe Tabelle 10). Für uns wichtig sind die Parameter »Class« (korrespondiert direkt zu den IDCMP-Flags in der Window-Struktur), »Code« (wo je nach Class spezielle Informationen abgelegt sind — beispielsweise bei Menüanwahl die Menü/Item/Subitem-Nummern) und »IAddress«, wo je nach Class ein Zeigerwert eingetragen wird (z.B. bei »Class= GADGETUP« ein Zeiger auf die Gadget-Struktur).

Mehr zur Theorie können wir hier aus Platzgründen nicht bieten. Wer sich in dieses interessante Thema tiefer einarbeiten möchte, sollte das sehr gute »Amiga Programmer's Handbook Vol.I« von Eugene P. Mortimore (Sybex Verlag 1987) zur Hand nehmen, wo eine ausführliche Abhandlung über dieses Thema zu finden ist.

Die in Listing 36 gezeigte Routine ist ein Anwendungsbeispiel, das für die meisten Anwendungen genügen dürfte. Sie fragt den »UserPort« des angegebenen Windows ab und liefert das Flag »GotIt%«, das -1 ist, wenn eine Nachricht empfangen wurde, sowie die Parameter bei empfangener Nachricht. Dies geschieht über die »Exec«-Funktion »Get Msg«. Wenn eine Nachricht an-

Die MenuItem-Struktur

&ptr	NextItem;	
%	LeftEdge, TopEdge;	
%	Width, Height;	
%	Flags;	
&	MutualExclude;	
&ptr	ItemFill;	Zeiger auf Image, IntuiText oder Null
&ptr	SelectFill;	wie ItemFill, für Selected
BYTE	Command;	wenn COMMSEQ gesetzt: Tastenschlüssel
&ptr	SubItem;	Untermenü
%	NextSelect;	

Flags, die Sie setzen können:		
CHECKIT	0x0001	
ITEMTEXT	0x0002	
COMMSEQ	0x0004	
MENUTOGGLE	0x0008	
ITEMENABLED	0x0010	
Folgende Flags für das Kennzeichnen beim Anwählen verwenden:		
HIGHIMAGE	0x0000	Sie haben eine eigene Image-Struktur, die Sie verwenden wollen
HIGHCOMP	0x0040	Select-Box wird komplementiert
HIGHBOX	0x0080	ein Rahmen wird gezeichnet
HIGHNONE	0x00C0	keine Kennzeichnung

Tabelle 10. Menüitems können auch SubItems haben

IntuiMessage-Struktur

struct	Message ExecMessage;	Unterstruktur
&	Class;	
%	Code;	
%	Qualifier;	
&ptr	IAddress;	
%	MouseX, MouseY;	
&	Seconds, Micros;	
&ptr	IDCMPWindow;	Zeiger auf zugehöriges Fenster
&ptr	SpecialLink;	nur fürs System

In »Class können die Werte auftauchen, die im zugehörigen Fenster ein Flag bei »IDCMPFlags« gesetzt haben.

Tabelle 11. Diese Struktur lesen Sie mit »GetMsg« aus dem Message-Port Ihres Windows

liegt, wird die Adresse der entsprechenden IntuiMessage-Struktur geliefert, ansonsten »0&«. Ist eine Nachricht da, werden einige Parameter ausgelesen und der Empfang der Nachricht mit »ReplyMsg« quittiert. Beachten Sie beim Arbeiten mit Tasks, daß Intuition darauf besteht, daß Nachrichten beantwortet werden — andernfalls mischt sich der Guru ein. Noch etwas ist bei Verwendung der Messages zu beachten: Wenn Sie Nachrichten in einem durch Basic geöffneten Window verarbeiten wollen, werden Sie Probleme bekommen: Das Basic-Window ist dann nämlich Diener dreier Herren: Intuition, Basic-Interpreter und Ihr Task beanspruchen denselben Message-Port. Mit Intuition klappt alles hervorragend, aber der Interpreter wird mit Ihnen um die Message kämpfen. Sie können nie vorhersehen, ob Sie eine bestimmte Message bekommen oder der Interpreter. Deshalb sollten Sie immer mit eigenen Fenstern arbeiten.

Nach all diesen Vorbereitungsarbeiten kommt nun endlich das Programm, in dem diese Bedienungselemente verwendet werden. Ich will in den folgenden Absätzen kurz einige wichtige Abschnitte des Demoprogrammes (Listing 37) besprechen.

Zeile 13 bis 55: (»Menu1« bis »xMenu m4&...«)

Hier werden die Menüstrukturen definiert. Zu jeder Menu- und MenuItem-Struktur wird eine IntuiText-Struktur benötigt. Achten Sie bei der Analyse dieses Programmteiles auf die Verkettungszeiger. Jeweils zusammengehörende Menu Items und die Menüs selbst sind verkettet!

Zeile 56 bis 60: (»CLS« bis »rport&=PEEKL...«)

Hier wird das Fenster geöffnet. Achten Sie auf die IDCMP-Flags: diese Nachrichten wollen wir später empfangen.

Zeile 61:
(»CALL SetMenuStrip...«)

Mit dem Befehl SetMenuStrip wird die Menüleiste an das Fenster angehängt. Achten Sie darauf, wirklich das erste (dies ist nicht das am weitesten links stehende, sondern das zuletzt definierte) Menu anzugeben — in unserem Falle ist das der Zeiger »m4&«.

Zeile 62 bis 109: (»NoMenu% =...« bis »END«)

Dies ist das eigentliche Hauptprogramm. Hier wird der MessagePort abgefragt und entsprechend reagiert. Wenn ein Menü angewählt wurde, wird mit ClearMenuStrip die Menüleiste gesperrt und in ein Unterprogramm verzweigt. Anschließend wird die Menüleiste wieder freigegeben.

Noch einige Besonderheiten: Wenn Gadgets angeklickt wurden, werden sie mit RemoveGadget aus der Gadgetliste entfernt. Das Image bleibt aber sichtbar, so daß mit SetRast der Bildschirm gelöscht wird und anschließend alle Gadgets wieder aufgefrischt werden (RefreshGadgets; beginnend mit dem ersten Gadget, dessen Adresse aus der Window-Struktur erfahren werden kann – Zeile 94). Im Falle des Proportional-Gadgets wird noch ein zweites Gadget dargestellt (Boolean), um das Entfernen des PropGadgets zu ermöglichen. Da man nicht will, daß ein PropGadget nach jedem Anklicken verschwindet, wurde ein Ausschalter in Form dieses BoolGadgets angebracht. Die beiden Gadgets sind verkettet, so daß mit dem Befehl RemoveGList (Zeile 98) beide entfernt werden können.

Das Beenden des Programms geschieht entweder durch Betätigen des Window Close-Gadgets (Zeile 107) oder durch Anwählen des Menüpunktes »Quit«.

Zeile 128 bis 154: (»PropGadget: bis »pg99:RETURN«)

Darstellung eines PropGadgets mit angehängtem BoolGadget.

Aufbau der Border-, PropInfo- und Gadget-Strukturen durch Aufruf der entsprechenden Unterprogramme. Dies geschieht nur, wenn das Gadget schieht nur, wenn das Gadget zum ersten Mal verwendet wird. Später ist pg1set%=1 -> das Gadget wird nur noch dargestellt. In Zeile 151 werden die beiden Gadgets mit AddGList in die Gadgetliste eingetragen und mit RefreshGadgets dargestellt.

Zeile 155 bis 189: (»TextGad:« bis »tg98:...RETURN«)

Hier wird das StringGadget aufgebaut und dargestellt. Für die vier Arten von StringGadgets gibt es eigene SubItems im Menü; es kann aber immer nur eines zugleich aktiv sein. Wir bauen uns daher nur eine Struktur auf und verändern entprechend der Darstellung (links-zentriert-rechts oder LongInt) die Flags. In Zeile 186 steht eine Besonderheit, die erst seit der Betriebssystem-Version 1.2 existiert: »ActivateGadget« kann ein StringGadget aktivieren, ohne daß dies mit der Maus angeklickt wird. Man kann also sofort nach Er-

```
 1 eM0 SUB xMenu(Men&, nextMenu&, LeftEdge%, TopEdge%, xWidth%, Hei
       ght%, Flags$, mName$, FirstItem&) STATIC
 2 L9  RESTORE mFlgDat
 3 UH  Flags%=0:i$=""
 4 Pe  WHILE i$<>"X"
 5 D12   READ i$,i%
 6 Z4    IF INSTR(Flags$, i$)<>0 THEN Flags%=Flags%+i%
 7 YM0 WEND
 8 J8  nl%=LEN(mName$)
 9 fD  MenuName&=AllocRemember&(0, nl%+1, 65540&)
10 of  FOR i%=1 TO nl%
11 MU2   POKE MenuName&+i%-1, ASC(MID$(mName$,i%,1))
12 HM0 NEXT
13 1k  Men&     =AllocRemember&(0, 30, 65540&)
14 J3  POKEL Men&   ,nextMenu&
15 43  POKEW Men&+ 4,LeftEdge%
16 4I  POKEW Men&+ 6,TopEdge%
17 fg  POKEW Men&+ 8,xWidth%
18 Bk  POKEW Men&+10,Height%
19 3x  POKEW Men&+12,Flags%
20 9e  POKEL Men&+14,MenuName&
21 yb  POKEL Men&+18,FirstItem&
22 Vj  EXIT SUB
23 eN  mFlgDat:
24 wR  DATA MENUENABLED, 1, X, 0
25 RT  END SUB
(C) 1988 M&T
```

Listing 34. Menüs werden in verbundenen Listen angelegt

```
 1 za0 SUB xMenI(MenI&, nextItem&, LeftEdge%, TopEdge%, xWidth%, He
       ight%, Flags$, MutualExclude&, ItemFill&, SelectFill&, Comman
       d%, SubItem&, NextSelect%) STATIC
 2 XE  RESTORE miFlgDat
 3 UH  Flags%=0:i$=""
 4 Pe  WHILE i$<>"X"
 5 D12   READ i$,i%
 6 ON    IF INSTR(Flags$,i$)<>0 THEN Flags%=Flags%+i%
 7 YM0 WEND
 8 0B  MenI&=AllocRemember&(0, 34, 65540&)
 9 04  POKEL MenI&   ,nextItem&
10 ZT  POKEW MenI&+ 4,LeftEdge%
11 oL  POKEW MenI&+ 6,TopEdge%
12 CD  POKEW MenI&+ 8,xWidth%
13 Gt  POKEW MenI&+10,Height%
14 T1  POKEW MenI&+12,Flags%
15 Lb  POKEL MenI&+14,MutualExclude&
16 3I  POKEL MenI&+18,ItemFill&
17 NP  POKEL MenI&+22,SelectFill&
18 87  POKE  MenI&+26,Command%
19 oN  POKEL MenI&+28,SubItem&
20 wI  POKEW MenI&+32,NextSelect%
21 Ui  EXIT SUB
22 ej  miFlgDat:
23 DM  DATA CHECKIT, 1, ITEMTEXT, 2, COMMSEQ, 4
24 i5  DATA MENUTOGGLE, 8, ITEMENABLED, 16
25 NI  DATA HIGHCOMP, 64, HIGHBOX, 128, HIGHNONE, 192, X, 0
26 SU  END SUB
(C) 1988 M&T
```

Listing 35. Menü-Items können mit dieser Routine eigene Sub-Items und Shortcuts erhalten

```
 1 C80 SUB AskIMsg(win&, GotIt%, classBit%, code%, Qualifier%, IAdd
       ress&, MouseX%, MouseY%) STATIC
 2 Zs  GotIt%=0
 3 f0  port&=PEEKL(win&+86)
 4 e3  msg&=GetMsg&(port&):IF msg&=0 THEN EXIT SUB
 5 DO  class&    =PEEKL(msg&+20)
 6 JY  classBit% =LOG(class&)/LOG(2)
 7 Oc  code%     =PEEKW(msg&+24)
 8 Zj  Qualifier%=PEEKW(msg&+26)
 9 md  IAddress& =PEEKL(msg&+28)
10 6J  MouseX%   =PEEKW(msg&+32)
11 JZ  MouseY%   =PEEKW(msg&+34)
12 Xq  CALL ReplyMsg(msg&)
13 vS  GotIt%=-1
14 GI  END SUB
(C) 1988 M&T
```

Listing 36. Das Message-System ist von Basic aus eine heikle Sache. »AskIMsg« nimmt Ihnen diese Arbeit ab.

scheinen des Gadgets Text in dieses eingeben.
Zeile 190 bis 235: (»BoolGad:« bis »bg01:...RETURN«)

Hier werden BoolGadgets entsprechend der SubMenü-Punkte aufgebaut. Ab dem zweiten Auftauchen werden die Strukturen wiederverwendet, um Speicher zu sparen — zu diesem Zweck werden die Flags »b..1set%« benutzt. Bringen Sie alle vier »BoolGadgets« auf den Bildschirm und versuchen Sie dann, das Fenster zu vergrößern oder verkleinern!
Zeile 238 bis 249: (»DMReq:« bis »dmset%=0...RETURN«)

Hier wird der noch immer sehr selten verwendete (Ausnahme: »Bard's Tale«) »DoubleMenuRequester« aufgebaut, der bei Doppelklick auf die rechte Maustaste erscheint. Der entsprechende Menüpunkt ist als Wechselschalter ausgeführt (»ON/OFF«), was in diesen Zeilen realisiert wird. Der eigentliche Aufbau findet im Unterprogramm statt; mit »SetDMRequest« wird der Requester aktiviert (dargestellt erst nach Doppelklick rechts) und mit »ClearDMRequest« wieder deaktiviert.
Zeile 250 bis 253: (»CustomReq:« bis »RETURN«)

Hier wird mit »Request« der Customrequester dargestellt. Der Requester selbst ist der gleiche wie bei »DMRequest«.
Zeile 254 bis 263: (»AutoReq:« bis »ERASE t$:RETURN«

Hier wird der AutoRequester aufgebaut.
Zeile 264 bis 272: (»Alert:« bis »ERASE t$,ty%:RETURN«)
Erzeugt einen Alert.
Zeile 273 bis 299: (»InitReq:« bis »—— subs —«)

Einrichten der erforderlichen Strukturen für den Custom-Requester.

Nach dieser sehr knappen Programmübersicht sollten Sie das Programm starten und alles ausprobieren. Wenn Sie die einzelnen Programmteile danach nochmals eingehend analysieren, lernen Sie die neuen Routinen am besten anzuwenden und eventuell Ihren Wünschen gemäß zu verändern.

Dateiauswahl ganz einfach

Im vorletzten Abschnitt unseres Kurses wollen wir alles, was bisher erarbeitet wurde, in einem sinnvollen Programm anwenden.

Wir werden einen File Requester programmieren, der folgende Eigenschaften haben soll:
— Selbständiges Erkennen aller angeschlossenen Laufwerke (inklusive resetfeste RAMdisk vd0:)
— Automatische Anzeige des aktuellen Verzeichnisses beim Aufruf
— Rollbalken für die Dateien
— Auswahl von Unterverzeichnissen durch Anklicken im Dateifenster
— Aufwärtsbewegen in der Verzeichnis-Struktur durch ein UPDIR-Gadget
— direkte Eingabe von Suchpfaden
— Erkennen von Suffices mit Anzeige einer Untermenge der vorhandenen Dateien.

Bild 2 zeigt, wie unser »FileRequester« aussehen wird. Er kann mehr als die meisten in professionellen Programmen angebotenen Dateiauswahlfenster.

Zuerst müssen wir den Suchpfad zum aktuellen Verzeichnis ermitteln. Das notwendige Unterprogramm ist in Listing 38 gezeigt. Die Suche startet beim aktuellen Verzeichnis und »hangelt« sich aufwärts bis zum Root-Verzeichnis. An Informationen über eine Datei kommen wir durch verschiedene Funktionen der »dos.library«. Zuerst müssen wir mit »Lock&« anmelden, daß wir Zugriff auf eine Datei benötigen. Die Syntax ist

```
l&=Lock&
(namptr&, access%)
```

wobei »namptr&« die Adresse eines Strings mit dem Namen der Datei (bzw. des Verzeichnisses) ist, »access%« gibt an, ob andere Dateien Zugriff haben sollen (»SHARED_ACCESS« oder -2 beziehungsweise »EXCLUSIVE_ACCESS« oder -1), und in »l&« wird die Adresse einer Lock-Struktur zurückgeliefert. Dies geschieht in Form eines »BPTR« (Zeiger auf ein Langwort, also Byte-Adresse dividiert durch 4), wie sie in der Programmiersprache »BCPL« verwendet werden. Die Lock-Struktur selbst braucht uns nicht weiter zu interessieren, sie ist nur als Parameter für die weiteren Funktionen nötig.

Wir übergeben in unserem Beispiel der Lock-Funktion keinen Namen, daher erhalten wir die Lock-Struktur des aktuellen Verzeichnisses zurück. Hinterher wird mit der Funktion

```
error%=Examine(l&, fib&)
```

die Information zu einer Datei (oder einem Verzeichnis) be-

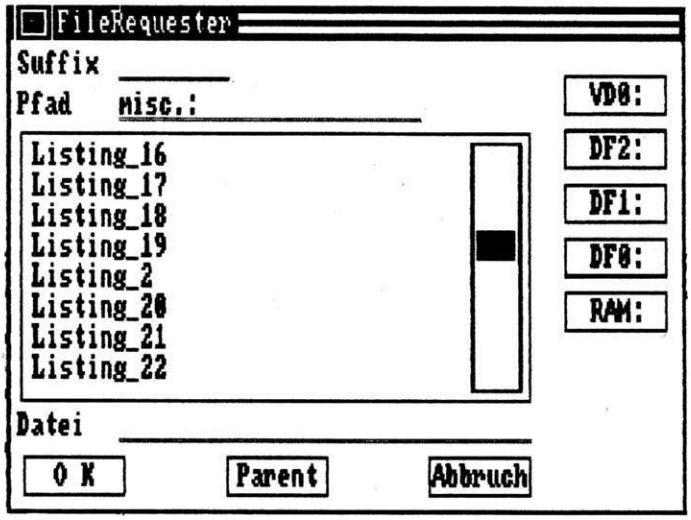


Bild 2. So sollte Ihr fertiger Filerequester aussehen

```
 1 SdO DECLARE FUNCTION AllocRemember& LIBRARY
 2 J1  DECLARE FUNCTION OpenWindow& LIBRARY
 3 Pn  DECLARE FUNCTION GetMsg& LIBRARY
 4 2B  DECLARE FUNCTION AutoRequest% LIBRARY
 5 LH  DECLARE FUNCTION Request% LIBRARY
 6 ZE  DECLARE FUNCTION AddGadget% LIBRARY
 7 Hi  DECLARE FUNCTION AddGList% LIBRARY
 8 FB  DECLARE FUNCTION RemoveGadget% LIBRARY
 9 He  LIBRARY"intuition.library"
10 iU  LIBRARY"exec.library"
11 8P  LIBRARY"graphics.library"
12 eZ  PRINT "Bitte haben Sie etwas Geduld, ich baue 37 Strukturen
       auf ...";
13 88  'Menu1
14 ho  xIText mi1t&,1,3,"JAM1",0,1,0&,"Quit",0&
15 ht  xMenI mi1&,0&,0,0,80,10,"ITEMTEXT | ITEMENABLED | HIGHCOMP
       | COMMSEQ",0&,mi1t&,0&,81,0&,0
16 do  xMenu m1&,0&,0,0,80,10,"MENUENABLED","Program",mi1&
17 EF  'Menu2
18 ZH  'Subitems
19 0x  xIText mi21t&,1,3,"JAM1",0,1,0&,"TopLeft",0&
20 pf  xMenI mi21&,0&,75,0,160,10,"ITEMTEXT | ITEMENABLED | HIGHCO
       MP | COMMSEQ",0&,mi21t&,0&,84,0&,0
21 Yk  xIText mi22t&,1,3,"JAM1",0,1,0&,"Right",0&
22 xI  xMenI mi22&,mi21&,75,10,160,10,"ITEMTEXT | ITEMENABLED | HI
       GHCOMP | COMMSEQ",0&,mi22t&,0&,86,0&,0
23 PL  xIText mi23t&,1,3,"JAM1",0,1,0&,"Bottom",0&
24 BT  xMenI mi23&,mi22&,75,20,160,10,"ITEMTEXT | ITEMENABLED | HI
       GHCOMP | COMMSEQ",0&,mi23t&,0&,66,0&,0
25 W5  xIText mi24t&,1,3,"JAM1",0,1,0&,"BottomRight",0&
26 Um  xMenI mi24&,mi23&,75,30,160,10,"ITEMTEXT | ITEMENABLED | HI
       GHCOMP | COMMSEQ",0&,mi24t&,0&,85,0&,0
27 Zr  xIText mi31t&,1,3,"JAM1",0,1,0&,"Text (left)",0&
28 QM  xMenI mi31&,0&,75,0,145,10,"ITEMTEXT | ITEMENABLED | HIGHCO
       MP | COMMSEQ",0&,mi31t&,0&,76,0&,0
29 Dj  xIText mi32t&,1,3,"JAM1",0,1,0&,"Text (center)",0&
30 FW  xMenI mi32&,mi31&,75,10,145,10,"ITEMTEXT | ITEMENABLED | HI
       GHCOMP | COMMSEQ",0&,mi32t&,0&,90,0&,0
31 id  xIText mi33t&,1,3,"JAM1",0,1,0&,"Text (right)",0&
32 bs  xMenI mi33&,mi32&,75,20,145,10,"ITEMTEXT | ITEMENABLED | HI
       GHCOMP | COMMSEQ",0&,mi33t&,0&,82,0&,0
33 rk  xIText mi34t&,1,3,"JAM1",0,1,0&,"LongInt",0&
34 uA  xMenI mi34&,mi33&,75,30,145,10,"ITEMTEXT | ITEMENABLED | HI
       GHCOMP | COMMSEQ",0&,mi34t&,0&,73,0&,0
35 tV  'Items
36 qm  xIText mi2t&,1,3,"JAM1",0,1,0&,"Boolean",0&
37 UD  xMenI mi2&,0&,-20,0,140,10,"ITEMTEXT | ITEMENABLED | HIGHCO
       MP",0&,mi2t&,0&,0,mi24&,0
38 ak  xIText mi3t&,1,3,"JAM1",0,1,0&,"String",0&
39 Le  xMenI mi3&,mi2&,-20,10,140,10,"ITEMTEXT | ITEMENABLED | HIG
       HCOMP",0&,mi3t&,0&,0,mi34&,0
40 9v  xIText mi4t&,1,3,"JAM1",0,1,0&,"Proportional",0&
41 kQ  xMenI mi4&,mi3&,-20,20,140,10,"ITEMTEXT | ITEMENABLED | HIG
       HCOMP | COMMSEQ",0&,mi4t&,0&,80,0&,0
42 7U  xMenu m2&,mi&,80,0,80,80,"MENUENABLED","Gadgets",mi4&
```

Listing 37. Dieses Steuerprogramm demonstriert die Anwendung der Listings 27 bis 36. Sie benötigen dazu auch Listing 17 und Listing 23.

```
 43 gi  'Menu3
 44 RF  xIText mi5t&,1,3,"JAM1",0,1,0&,"Auto",0&
 45 16  xMenI mi5&,0&,-10,0,130,10,"ITEMTEXT | ITEMENABLED | HIGHCO
        MP | COMMSEQ",0&,mi5t&,0&,83,0&,0
 46 wt  xIText mi6t&,1,3,"JAM1",0,1,0&,"Custom",0&
 47 au  xMenI mi6&,mi5&,-10,10,130,10,"ITEMTEXT | ITEMENABLED | HIG
        HCOMP | COMMSEQ",0&,mi6t&,0&,67,0&,0
 48 Y4  xIText mi7t&,1,3,"JAM1",0,1,0&,"DM-Req ON ",0&
 49 lw  dmtxt&=IText&
 50 yI  xMenI mi7&,mi6&,-10,20,130,10,"ITEMTEXT | ITEMENABLED | HIG
        HCOMP | COMMSEQ",0&,mi7t&,0&,68,0&,0
 51 cY  xMenu m3&,m2&,160,0,90,10,"MENUENABLED","Requester",mi7&
 52 ru  'Menu4
 53 sN  xIText mi8t&,1,3,"JAM1",0,1,0&,"Do Alert!",0&
 54 oM  xMenI mi8&,0&,0,0,110,10,"ITEMTEXT | ITEMENABLED | HIGHCOMP
        | COMMSEQ",0&,mi8t&,0&,65,0&,0
 55 yM  xMenu m4&,m3&,250,0,60,10,"MENUENABLED","Alert",mi8&
 56 SY  CLS
 57 CI  s&=PEEKL(WINDOW(7)+46)
 58 ZU  NewWindow w&,100,40,500,155,1,3,"CLOSEWINDOW | MENUPICK | G
        ADGETUP","GIMMEZEROZERO | WINDOWCLOSE | WINDOWSIZING | ACTIV
        ATE",0&, 0&,"Bedienungselemente - Demo",s&,0&,120,40,540,200
        ,"WBE 59 pb  NCHS
 59 pb  NCHS
 60 L9  rport&=PEEKL(w&+50)
 61 X1  CALL SetMenuStrip(w&,m4&)
 62 pQ  NoMenu%=31:NoItem%=63:NoSub%=31
 63 vc  WHILE NOT aus%
 64 3Y  Ask:
 65 wp2   AskIMsg w&,GotIt%,class%,code%,q%,IAdd&,mx%,my%
 66 nG    IF GotIt%=0 GOTO Ask
 67 81    IF class%=9 THEN
 68 7Z4     aus%=-1
 69 rS2   ELSEIF class%=8 THEN
 70 Sg4     code&=code%:IF code&<0 THEN code&=code&+65536&
 71 Da      MenuNum%= code&      AND 31&
 72 Jr      ItemNum%=(code&\32)  AND 63&
 73 qm      SubNum% =(code&\2048)AND 31&
 74 4J      IF MenuNum%<>NoMenu% THEN
 75 fI6       CALL ClearMenuStrip(w&)
 76 to        ON (MenuNum%+1)GOSUB Alert, Requester, Gadgets, Progr
              am
 77 nH        CALL SetMenuStrip(w&,m4&)
 78 C54     END IF
 79 pY2   ELSEIF class%=6 THEN
 80 884     id%=PEEKW(IAdd&+38)
 81 v6      IF (id%=16) THEN
 82 OF6       tbuf&=buf&:GOSUB showTgTxt
 83 p14     ELSEIF (id%>=17)AND(id%<=21) THEN
 84 ki6       IF (id%=17) THEN gpos%= RemoveGadget%(w&,bgtlg&)
 85 gj        IF (id%=18) THEN gpos%= RemoveGadget%(w&,bgrg&)
 86 VE        IF (id%=19) THEN gpos%= RemoveGadget%(w&,bgbg&)
 87 K5        IF (id%=20) THEN gpos%= RemoveGadget%(w&,bgbrg&)
 88 d9        IF (id%=21) THEN
 89 6d7        gpos%= RemoveGadget%(w&,tg&)
 90 jd         tbuf&=tgbuf&:GOSUB showTgTxt
 91 wx         tgact%=0
 92 QJ6       END IF
 93 ty        CALL SetRast(rport&, 0)
 94 Im        fg&=PEEKL(w&+62):CALL RefreshGadgets(fg&,w&,0&)
 95 tA4     ELSEIF (id%=22) THEN
 96 xk6       GOSUB showPgTxt
 97 0I4     ELSEIF (id%=23) THEN
 98 wX6       CALL RemoveGList(w&,pg&,2)
 99 z4        CALL SetRast(rport&, 0)
100 Os        fg&=PEEKL(w&+62):CALL RefreshGadgets(fg&,w&,0&)
101 yv        pgact%=0
102 aT4    END IF
103 55     IF pgact%=1 THEN GOSUB showPgTxt
104 cV2  END IF
105 8w0 WEND
106 by  xit:
107 kz  CALL CloseWindow(w&)
108 tT  CALL FreeRemember(0,-1)
109 VQ  END
110 15  '--- ups ----------------------------
111 sm  showTgTxt:
112 nW4     CLS:LOCATE 1,1
113 cc      j%=0
114 GMO rt: i%=PEEK(tbuf&+j%)
115 iv4     IF i%<>0 THEN PRINT CHR$(i%);:j%=j%+1:GOTO rt
116 Eq0 RETURN
117 ak  showPgTxt:
118 eN1  hp&=PEEKW(pgPInfo&+2):hp&=hp&+65536&*(hp&<0)
119 dj   tp%=(LEN(pg$)-20)*hp&/65536&
```

```
120 vz   CALL Move(rport&,20,30)
121 t9   CALL Text(rport&,SADD(pg$)+tp%,20)
122 Kw0 RETURN
123 lh  Program:
124 1y  IF ItemNum%=0 THEN aus%=-1
125 Nz  RETURN
126 dD  Gadgets:
127 bM  ON (ItemNum%) GOTO TextGad, BoolGad
128 yg  PropGad:
129 up  IF pg1set%=1 GOTO pg01
130 mY  pg1set%=1
131 eK  pgxy&=AllocRemember&(0,20,65540&)
132 iE  RESTORE pgbordat:FOR i%=0 TO 9:READ j%:POKEW pgxy&+2*i%,j%:
        NEXT
133 T1  xBorder pg1bor&,0,0,1,2,"JAM1",5,pgxy&,0&
134 bO  pgbordat:
135 Iv  DATA -2,-2,162,-2,162,10,-2,10,-2,-2
136 EM  pg$="Dies ist ein sehr langer Text,"
137 KC  pg$=pg$+" von dem Sie mit Hilfe des "
138 9O  pg$=pg$+"Schiebereglers Ausschnitte "
139 gE  pg$=pg$+"betrachten können."
140 MA  hb%=65536&*20/LEN(pg$)
141 7v  xPropInfo pgPInfo&,"AUTOKNOB | FREEHORIZ", 0, &HFFFF, hb%,
        &HFFFF
142 dU  xGadget pg1&,0&,20,50,160,8,"","RELVERIFY | GADGIMMEDIATE",
        "PROPGADGET",pg1bor&,0&,0&,0&,pgPInfo&,22,0&
143 qW  pgxy&=AllocRemember&(0,20,65540&)
144 3q  RESTORE g2bordat:FOR i%=0 TO 9:READ j%:POKEW pgxy&+2*i%,j%:
        NEXT
145 bN  xBorder pgbor&,0,0,1,1,"JAM1",5,pgxy&,0&
146 vd  xIText  pgtxt&,3,0,"JAM1",7,3,0&," OK ",0&
147 5S  xGadget pg&,pg1&,75,62,46,13,"","RELVERIFY | GADGIMMEDIATE"
        ,"BOOLGADGET",pgbor&,0&,pgtxt&,0&,0&,23,0&
148 Kc  pg01:
149 NL  IF pgact%=1 GOTO pg99
150 pn  pgact%=1
151 SV  gpos%=AddGList%(w&, pg&,-1,-1, 0&)
152 13  CALL RefreshGadgets(pg&, w&, 0&)
153 dC  pg99:
154 qS  RETURN
155 oa  TextGad:
156 z2  IF tg1set%=1 GOTO tg01
157 LB  tg1set%=1
158 Dx  tgxy&=AllocRemember&(0,20,65540&)
159 bF  RESTORE tgbordat:FOR i%=0 TO 9:READ j%:POKEW tgxy&+2*i%,j%:
        NEXT
160 TO  xBorder tgbor&,0,0,1,2,"JAM1",5,tgxy&,0&
161 Ad  tgbordat:
162 tq  DATA -2,-2,202,-2,202,10,-2,10,-2,-2
163 wa  xStrInfo tgSInfo&,tgbuf&,tgu&,0,40,0,1989&
164 iT  xGadget tg&,0&,20,30,200,8,"SELECTED","RELVERIFY | GADGIMME
        DIATE","STRGADGET",tgbor&,0&,0&,0&,tgSInfo&,21,0&
165 j5  tg01:
166 CH  IF tgact%=1 GOTO tg98
167 EG  tgact%=1
168 vN  ON (SubNum%) GOTO txtr,txtc,txtl
169 wU  lngint:
170 Tv  POKEW tg&+14,(PEEKW(tg&+14)AND &HF1FF) OR &H800
171 8E  l$=STR$(PEEKL(tgSInfo&+28)):IF LEFT$(l$,1)=" " THEN l$=MID$
        (l$,2)
172 zE  FOR tgp%=0 TO LEN(l$)-1:POKE tgbuf&+tgp%,ASC(MID$(l$,tgp%+1
        )):NEXT:POKE tgbuf&+tgp%,0
173 VV  POKEW tgSInfo&+8,tgp%:POKEW tgSInfo&+12,0:GOTO tg97
174 K6  txtr:
175 93  POKEW tg&+14,(PEEKW(tg&+14)AND &HF1FF) OR &H400:GOTO tg99
176 9g  txtc:
177 z1  POKEW tg&+14,(PEEKW(tg&+14)AND &HF1FF) OR &H200:GOTO tg99
178 ua  txtl:
179 EC  POKEW tg&+14,(PEEKW(tg&+14)AND &HF1FF)
180 Cp  tg99:
181 by  tgp%=0:WHILE PEEK(tgbuf&+tgp%)<>0:tgp%=tgp%+1:WEND
182 E1  POKEW tgSInfo&+8,tgp%:POKEW tgSInfo&+12,0
183 5g  tg97:
184 nZ  gpos%=AddGadget%(w&, tg&, -1)
185 Qm  CALL RefreshGadgets(tg&, w&, 0&)
186 HE  CALL ActivateGadget(tg&, w&, 0&)
187 Eq  tg98:
188 MO  LOCATE 3,1:PRINT USING"# # # # # #";PEEKW(tg&+14);
189 P1  RETURN
190 3Q  BoolGad:
191 DN  IF bbset%=1 GOTO bg99
192 87  bbset%=1
193 Ce  bgxy&=AllocRemember&(0,20,65540&)
194 8C  RESTORE bgbordat:FOR i%=0 TO 9:READ j%:POKEW bgxy&+2*i%,j%:
        NEXT
```

```
195 Kd  xBorder bgbor&,0,0,1,0,"JAM1",5,bgxy&,0&
196 9K  bgbordat:
197 9f  DATA 0,0,99,0,99,11,0,11,0,0
198 uF  bg99:
199 bS  ON (SubNum%) GOTO boolb, boolr, booltl
200 h1  boolbr:
201 si  IF bbr1set%=1 GOTO bg04
202 sA  bbr1set%=1
203 l1  xIText bgbrt&,1,0,"JAM1",5,3,0&,"BOTTOMRIGHT",0&
204 kf  xGadget bgbrg&,0&,-120,-30,100,12,"GRELRIGHT | GRELBOTTOM",
        "RELVERIFY | GADGIMMEDIATE","BOOLGADGET",bgbor&,0&,bgbrt&,0&
        ,0&,20,0&
205 29  bg04:
206 2M  gpos%=AddGadget%(w&, bgbrg&, -1)
207 gd  CALL RefreshGadgets(bgbrg&, w&, 0&)
208 iK  RETURN
209 mm  boolb:
210 ZO  IF bb1set%=1 GOTO bg03
211 Or  bb1set%=1
212 I8  xIText bgbt&,1,0,"JAM1",5,3,0&,"BOTTOMLEFT",0&
213 lR  xGadget bgbg&,0&,20,-30,100,12,"GRELBOTTOM","RELVERIFY | GA
        DGIMMEDIATE","BOOLGADGET",bgbor&,0&,bgbt&,0&,0&,19,0&
214 6C  bg03:
215 Hq  gpos%=AddGadget%(w&, bgbg&, -1)
216 29  CALL RefreshGadgets(bgbg&, w&, 0&)
217 rT  RETURN
218 TR  boolr:
219 Dz  IF br1set%=1 GOTO bg02
220 J2  br1set%=1
221 jf  xIText bgrt&,1,0,"JAM1",5,3,0&,"TOPRIGHT",0&
222 Jr  xGadget bgrg&,0&,-120,20,100,12,"GRELRIGHT","RELVERIFY | GA
        DGIMMEDIATE","BOOLGADGET",bgbor&,0&,bgrt&,0&,0&,18,0&
223 AF  bg02:
224 iX  gpos%=AddGadget%(w&, bgrg&, -1)
225 Da  CALL RefreshGadgets(bgrg&, w&, 0&)
226 0c  RETURN
227 gk  booltl:
228 fC  IF btl1set%=1 GOTO bg01
229 nH  btl1set%=1
230 6B  xIText bgtlt&,1,0,"JAM1",5,3,0&,"TOPLEFT",0&
231 U3  xGadget bgtlg&,0&,20,20,100,12,"","RELVERIFY | GADGIMMEDIAT
        E","BOOLGADGET",bgbor&,0&,bgtlt&,0&,0&,17,0&
232 EI  bg01:
233 LL  gpos%=AddGadget%(w&, bgtlg&, -1)
234 nw  CALL RefreshGadgets(bgtlg&, w&, 0&)
235 9l  RETURN
236 qR  Requester:
237 lH  ON (ItemNum%) GOTO CustomReq, AutoReq
238 e4  DMReq:
239 lH  IF dmset%=0 THEN
240 TF1  GOSUB InitReq
241 Rb   CALL SetDMRequest(w&,req&)
242 gD   POKE dmtxt&+8,ASC("F"):POKE dmtxt&+9,ASC("F")
243 Yk   dmset%=1
244 XG0 ELSE
245 8c1  CALL ClearDMRequest(w&)
246 4T   POKE dmtxt&+8,ASC("N"):POKE dmtxt&+9,ASC(" ")
247 Yj   dmset%=0
248 wp0 END IF
249 Nz  RETURN
250 Sm  CustomReq:
251 eQ1  GOSUB InitReq
252 ow   ok%= Request%(req&,w&)
253 R3   RETURN
254 js0 AutoReq:
255 RG1  f$="\           \"
256 90   t$(0)="Dies ist ein Auto-Requester,"
257 8u   t$(1)="die   einfachste   Form  des"
258 BH   t$(2)="Requesters."
259 T2   xAutoReq GotWhich%,w&,3,t$(),"Date","Time",350,80
260 r6   LOCATE 8+GotWhich%,1
261 S9   IF GotWhich%=0 THEN PRINT USING f$;TIME$ ELSE PRINT USING
         f$;DATE$
262 zm   ERASE t$
263 bD   RETURN
264 2E0 Alert:
265 m3  anz%=3
266 PR  DIM t$(anz%-1),ty%(anz%-1)
267 zP  ty%(0)=30:t$(0)="Sie haben soeben einen INTUITION-ALERT aus
        gelöst !!!"
268 0p  ty%(1)=42:t$(1)="Drücken Sie die linke Maustaste,"
269 MQ  ty%(2)=54:t$(2)="um ins Programm zurückzukehren"
270 Da  xAlert anz%,t$(),ty%(),"CENTER","RECOVERY",80
271 LQ  ERASE t$,ty%
272 kM  RETURN
```

```
273 ei  InitReq:
274 qL1  IF rq1set%=1 GOTO rqs
275 fd   rq1set%=1
276 5k   g1xy&=AllocRemember&(0,20,65540&)
277 C2   RESTORE g1bordat:FOR i%=0 TO 9:READ j%:POKEW g1xy&+2*i%,j%
         :NEXT
278 Aq   g2xy&=AllocRemember&(0,20,65540&)
279 NA   RESTORE g2bordat:FOR i%=0 TO 9:READ j%:POKEW g2xy&+2*i%,j%
         :NEXT
280 bT   rqxy&=AllocRemember&(0,20,65540&)
281 nt   RESTORE rqbordat:FOR i%=0 TO 9:READ j%:POKEW rqxy&+2*i%,j%
         :NEXT
282 pz   xBorder g1bor&,0,0,0,1,"JAM1",5,g1xy&,0&
283 18   xBorder g2bor&,0,0,0,1,"JAM1",5,g2xy&,0&
284 00   xBorder rqbor&,0,0,1,3,"JAM1",5,rqxy&,0&
285 FC   xIText  g2txt&,1,0,"JAM1",7,3,0&," OK ",0&
286 6K   xIText  rqtxt&,0,0,"JAM1",80,5,0&,"Custom-Requester",0&
287 aq   xStrInfo SInfo&,buf&,u&,0,40,0,0&
288 Fy   xGadget g1&,0&,20,30,160,8,"SELECTED","RELVERIFY | GADGIMM
         EDIATE","REQGADGET | STRGADGET",g1bor&,0&,0&,0&,SInfo&,15,0
         &
289 gf   xGadget g2&,g1&,-60,-20,46,13,"GRELBOTTOM | GRELRIGHT","RE
         LVERIFY | GADGIMMEDIATE | ENDGADGET","REQGADGET | BOOLGADGE
         T",g2bor&,0&,g2txt&,0&,0&,16,0&
290 h50 rqs:
291 oP1  xReq req&,100,20,300,120,0,0,g2&,rqbor&,rqtxt&,"",2
292 4g0 RETURN
293 Ge  g1bordat:
294 Bu  DATA -4,-2,162,-2,162,10,-4,10,-4,-2
295 Lk  g2bordat:
296 gk  DATA 0,0,45,0,45,12,0,12,0,0
297 mN  rqbordat:
298 lK  DATA 4,2,295,2,295,116,4,116,4,2
299 7r  '---- subs ----------------------------
(C) 1988 M&T
```

Listing 37. (Schluß)

```
 1 yd0 SUB CurrentDir(path$) STATIC 'returns complete path for curr
       ent dir in path$
 2 b82   IF fib&=0 THEN fib&=AllocRemember&(0, 260, 65540&)
 3 ib    l&=Lock&(SADD(CHR$(0)), -2)
 4 Zh    path$=""
 5 jb0 cd1:
 6 pZ2   e%=Examine%(l&, fib&)
 7 TC    i$="":i%=0
 8 qj0 cd2:
 9 uI2   j%=PEEK(fib&+8+i%):IF j%<>0 THEN i$=i$+CHR$(j%):i%=i%+1:
         GOTO cd2
10 Ij    pl&=ParentDir&(l&)
11 bU    CALL Unlock(l&)
12 o5    l&=pl&
13 bY    IF l&<>0 THEN path$=i$+"/"+path$:GOTO cd1
14 Ds    path$=i$+":"+path$
15 Ye    IF RIGHT$(path$,1)<>":" THEN path$=LEFT$(path$,LEN(path$
         )-1)
16 IK0 END SUB
(C) 1988 M&T
```

Listing 38. Mit »CurrentDir« ermitteln Sie den Pfad zum aktuellen Directory

```
 1 mp0 SUB ReadDir (count%, na$, fin$(), pat$, pmod%) STATIC
 2 Z12   na$=na$+CHR$(0):pat$=UCASE$(pat$)
 3 6j    IF pat$<>oldpat$ THEN oldna$=""
 4 w3    IF na$=oldna$ GOTO xit2  'dir already read
 5 BW    oldna$=na$:oldpat$=pat$
 6 Av    lo&=Lock&(SADD(na$),-2)
 7 yw    IF lo&=0 THEN count%=-1:EXIT SUB
 8 1N    IF fib&=0 THEN fib&=AllocRemember&(0&,260,65540&)
 9 Z4    e%=Examine%(lo&,fib&)
10 e1    count%=0
11 re0 en:
12 sb2   e%=ExNext%(lo&,fib&)
13 eh    IF e%=0 GOTO xit
14 pv    k%=SGN(PEEKL(fib&+4))
15 VZ    GetStr f$,(fib&+8)
16 gr    IF k%=1 THEN f$=CHR$(1)+f$+" (dir)"
17 Q6    IF (k%=1)OR(pmod%=0)OR((pmod%=1)AND(INSTR(UCASE$(f$),pat$)
         =1))OR((pmod%=2)AND(INSTR(UCASE$(f$),pat$)<>0))OR((pmod%=
         3)AND(RIGHT$(UCASE$(f$),LEN(pat$))=pat$)) THEN
```

Listing 39. »ReadDir« liest die Informationen des »FileInfoBlocks«

```
18 bq4      fin$(count%)=f$
19 7s       count%=count%+1
20 G92    END IF
21 1N     GOTO en
22 Fc0 xit:
23 g52    CALL Unlock(lo&)
24 9K     FOR i%=count% TO count%+8
25 gA4      fin$(i%)=CHR$(0)
26 Va2    NEXT
27 C10 xit2:
28 lj2    na$=LEFT$(na$,LEN(na$)-1)
29 VX0 END SUB
(C) 1988 M&T
```

Listing 39. (Schluß)

reitgestellt. Für diese Informationen haben wir einen Speicherbereich von 260 Byte reserviert. Bei wiederholtem Aufruf des Unterprogramms wird immer derselbe Speicherbereich verwendet.

Aus diesem »FileInfoBlock« lesen wir den Namen des aktuellen Verzeichnisses in die Variable »i$« aus. Dann ermitteln wir mit

```
pl&=ParentDir(l&)
```

die Lockstruktur des Stammverzeichnisses. Anschließend geben wir die Zugriffs-Reservierung mit »Unlock(l&)« wieder frei. Wenn noch ein Stammverzeichnis gefunden wurde, stellen wir den Namen — durch »/« getrennt — vor den bisherigen Pfadnamen.

Andernfalls war der letzte gefundene Name bereits die Root. In diesem Fall stellen wir den Namen mit einem Doppelpunkt abgeschlossen vor den Pfadnamen.

Abschließend wird ein eventuell vorhandener Trennstrich am Ende des Pfades abgeschnitten.

Einlesen eines Verzeichnisses

Listing 39 zeigt nun, wie man ein Verzeichnis in ein String-Array einliest. Diese Routine kann nach den Kriterien »Prefix«, »Suffix« und »Wildcard« eine Untermenge der Filenamen auswählen. Dafür gibt es die Variable »pat$« und die Variable »pmod%«, die angibt, ob »pat$« als Suffix, Prefix oder Teilkette gewertet werden soll.

Zu Beginn der Routine wird geprüft, ob das gleiche Verzeichnis eben schon eingelesen wurde. Wenn dies der Fall ist, wird die Routine verlassen. Mit »Lock« wird der Suchpfad reserviert. Mit »Examine« erhalten wir den ersten Eintrag. Die weiteren Einträge werden mit

```
e%=ExNext(lo&, fib&)
```

ermittelt (beachten Sie, daß hier nur ein »Lock« erforderlich ist).

Im »FileInfoBlock« steht auch, ob es sich um ein Verzeichnis oder eine Datei handelt. Wenn wir ein Verzeichnis vor uns haben, wird dem Filenamen ein CHR$(1) vorangestellt. Das führt beim späteren Sortieren dazu, daß alle Unterverzeichnisse zu Beginn der Liste erscheinen.

Welche Laufwerke sind angeschlossen?

Listing 40 ermittelt alle angeschlossenen Laufwerke. Es benötigt ein im Hauptprogramm als »SHARED« deklariertes String-Array mit dem Namen »unit$()«. Das Programm verwendet einen kleinen Trick — alles andere wäre in Basic zu langsam. Dieser Trick besteht in der Verwendung des CLI-Kommandos »info«. Mit dem »dos.library«-Befehl

```
e%=Execute%
(befstrptr&, 0, 0)
```

wird ein CLI-Befehl ausgeführt; in unserem Fall »info >ram:info.tmp«. Aus der erzeugten Datei werden alle »trackdisk.device«-Namen ausgelesen und die Datei wieder gelöscht.

Das Hauptprogramm des FileRequesters (Listing 41) ist gut dokumentiert, so daß Sie es sicher auch ohne weitere Erklärung leicht verstehen werden. Lediglich einen kleinen Trick, den ich zur Beschleunigung verwendet habe, möchte ich Ihnen noch erklären:

Wir benötigen eine ganze Reihe ähnlicher Strukturen des gleichen Typs. Wenn wir jedesmal die entsprechende Subroutine aufrufen, kostet das eine Menge Zeit. Daher reserviere ich ausreichend Speicherplatz für die benötigten Strukturen, initialisiere die erste mit der entsprechenden Subroutine und kopiere diese mit »CopyMemQuick« in den reservierten Speicher. »CopyMemQuick« können wir hier verwenden, da alle Strukturen eine durch 4 teilbare Länge haben und »AllocRemember« immer »long-aligned« arbeitet.

Ich möchte Ihnen nochmals ans Herz legen, das Programm eingehend zu analysieren, Sie lernen sicherlich dabei mehr als beim Lesen einer langatmigen Programmbeschreibung!

Device-IO am Beispiel des Druckers

Zum Abschluß unseres Kurses wollen wir noch eine Hardcopy-Routine entwickeln, die die einfache Auswahl eines beliebigen Fensters in einem beliebigen Screen (der nicht von Basic erzeugt sein muß) ermöglicht. Die ziemlich umfangreiche Subroutine ist in Listing 42 abgedruckt.

Zur Markierung des gewählten Fensters verwenden wir ein Sprite, das zu Beginn initialisiert wird. Die notwendige »SimpleSprite«-Struktur ist in Tabelle 11 gezeigt.

Nach Öffnen eines Fensters werden alle Zeiger auf Screen-Strukturen eingelesen. Dafür ist es von Nutzen, daß der aktuelle Screen immer der erste in der Liste ist. Alle Screens außer dem aktuellen (der in unserem Fall der WorkBench-Screen ist) werden so verschoben, daß ihr oberer Rand direkt unterhalb des Kommunikationsfensters liegt.

Nach Auswahl eines Screens kann man zwischen allen in diesem Screen geöffneten Fenstern wählen oder den ganzen Screen drucken. Wenn ein Fenster gewählt ist, wird es nach vorne geholt und das Sprite in die linke obere Ecke positioniert.

Sowohl der Screen- als auch der Fenstertitel werden in der Titelleiste des Kommunikationsfensters angezeigt.

Die Vorbereitung der für den Druck notwendigen Strukturen ist im Listing ausführlich dokumentiert, so daß eine weitere Erklärung des Programms (Listing 43) überflüssig ist. Zu den Flags der »IODRPReq«-Struktur sollten Sie sich Tabelle 12 ansehen.

Im Laufe dieses Kurses haben wir uns von einfachen Text- und Zeichenbefehlen über Window-Manipulationen in Bereiche vorgewagt, die bisher als für Basic-Programmierer unzugänglich galten.

Es gäbe noch sehr viel mehr zu tun, aber damit würden wir die Grenzen eines Kurses sprengen. Wenn Sie noch tiefer in die Geheimnisse Ihres AMIGA eindringen wollen, sollten Sie einen Blick in das »Amiga-Programmierhandbuch« von Frank Kremser und Jörg Koch (M&T-Verlag) werfen. Mit den hier erworbenen Informationen dürfte es kein Problem sein, auch mit Basic noch weiter abzutauchen in die Tiefen des Systems. *(Dr. Peter Gründler/so)*

Die SimpleSprite-Struktur

&ptr	posctldata;	Zeiger auf Daten
%	height;	
%	x,y;	aktuelle Position
%	num;	Spritenummer

Tabelle 12. Die SimpleSprite-Struktur dient zur vereinfachten Verwaltung von Sprites

Die IODRPR-Struktur

struct	Message io_Message;	
struct	Device *io_Device;	Zeiger auf zugehöriges Device
struct	Unit *io_Unit;	
UWORD	io_Command;	Kommando für Device
UBYTE	io_Flags;	
BYTE	io_Error;	Fehlernummer
struct	RastPort *io_RastPort;	Zeiger auf RastPort
struct	ColorMap *io_ColorMap;	Zeiger auf Farbtabelle
ULONG	io_Modes;	
UWORD	io_SrcX;	
UWORD	io_SrcY;	
UWORD	io_SrcWidth;	
UWORD	io_SrcHeight;	
LONG	io_DestCols;	
LONG	io_DestRows;	
UWORD	io_Special;	

Tabelle 13. Die IODRPReq-Struktur wird in der Hardcopy-Routine verwendet

```
 1 ZdO SUB FindUnits(i%) STATIC    'makes use of the DOS-Info comma
       nd to find all
 2 JUS                             'mounted units
 3 DFO e%=Execute%(SADD("info >ram:info.tmp"+CHR$(0)), 0, 0)
 4 dL  OPEN "i",1,"ram:info.tmp"
 5 34  FOR i%=1 TO 3:LINE INPUT #1,i$:NEXT   'discard first three
       lines
 6 rq  i%=0
 7 4z  11:
 8 q6  LINE INPUT #1,i$
 9 dA  IF LEN(i$)>4 THEN
10 gA2   unit$(i%)=LEFT$(i$,4)
11 CE    i%=i%+1
12 9d    GOTO 11
13 920 END IF
14 2m  CLOSE 1
15 rr  KILL"ram:info.tmp"
16 IK  END SUB
(C) 1988 M&T
```

Listing 40. »FindUnits« ermittelt alle angeschlossenen Laufwerke

```
 1 cAO CLEAR ,50000&
 2 Te  DECLARE FUNCTION AllocRemember& LIBRARY
 3 Kj  DECLARE FUNCTION OpenWindow& LIBRARY
 4 b5  DECLARE FUNCTION Lock& LIBRARY
 5 os  DECLARE FUNCTION ParentDir& LIBRARY
 6 ua  DECLARE FUNCTION Examine% LIBRARY
 7 yk  DECLARE FUNCTION ExNext% LIBRARY
 8 Us  DECLARE FUNCTION GetMsg& LIBRARY
 9 zX  DECLARE FUNCTION Execute% LIBRARY
10 bf  LIBRARY"dos.library"
11 jV  LIBRARY"exec.library"
12 9Q  LIBRARY"graphics.library"
13 Li  LIBRARY"intuition.library"
14 Ho  '------------------------------------
15 de  DIM SHARED unit$(6)
16 XC  FindUnits uAnz%   'which and how many Diskunits are mounted
       ?
17 Kr  '------------------------------------
18 EO  gAnz%=15+uAnz%
19 eM  DIM SHARED gTxt&(gAnz%), g&(gAnz%), w&
20 fz  DIM gBor&(gAnz%)
21 gw  DIM SHARED fileName$(200)
22 7Z  n$=CHR$(0)
23 Vx  lenIText%=20:lenBor%=16:lenGad%=44  'lengths of structures
24 Ry  '------------------------------------
25 IV  '*** prepare GadgetText ***
26 qB  xIText gTxt&(1),1,0,"",0,0,0&,"",0&
27 lF  gTxtMem&=AllocRemember&(0, lenIText%*(gAnz%-1), 65540&)
28 D2  'we want to be quick, so we copy predefined structures into
       reserved RAM
29 si  'instead of calling the slow xIText subroutine repeatedly !
30 e0  FOR i%=0 TO gAnz%-2
31 412   gTxt&(i%+2)=gTxtMem&+lenIText%*i%
32 b6    CALL CopyMemQuick(gTxt&(1),gTxt&(i%+2),lenIText%)
33 ch0 NEXT
34 Tu  t$="  O K  "+n$+"Abbruch"+n$
35 KO  t$=t$+"Suffix"+n$+"Pfad"+n$+"Datei"+n$+"Parent"+n$
36 BI  FOR i%=0 TO uAnz%:t$=t$+unit$(i%)+n$:NEXT
37 XH  i&=40+5*uAnz%
38 tS  bg&=AllocRemember&(0,i&,65540&)
39 ae  FOR i%=1 TO LEN(t$):POKE bg&+i%-1,ASC(MID$(t$,i%)):NEXT
40 ZM  POKEW gTxt&(9)+4,2:POKEW gTxt&(9)+6,2:POKEL gTxt&(9)+12,bg&
41 1A  POKEW gTxt&(10)+4,2:POKEW gTxt&(10)+6,2:POKEL gTxt&(10)+12,
       bg&+8
42 Pu  POKEW gTxt&(11)+4,-60:POKEL gTxt&(11)+12,bg&+16
43 Qv  POKEW gTxt&(12)+4,-60:POKEL gTxt&(12)+12,bg&+23
44 nP  POKEW gTxt&(13)+4,-60:POKEL gTxt&(13)+12,bg&+28
45 9E  POKEW gTxt&(15)+4,6:POKEW gTxt&(15)+6,2:POKEL gTxt&(15)+12,
       bg&+34
46 jI  FOR i%=0 TO uAnz%-1
47 2f2   POKEW gTxt&(16+i%)+4,15:POKEW gTxt&(16+i%)+6,2
48 61    POKEL gTxt&(16+i%)+12,bg&+41+5*i%
49 sx0 NEXT
50 cD  '*** prepare GadgetBorder Data ***
51 Nk  BorXY&=AllocRemember&(0,42,65540&)
52 yg  RESTORE Bordat
53 aq  FOR i%=0 TO 20:READ j%:POKEW BorXY&+2*i%,j%:NEXT
54 pe  Bordat:
55 2z  DATA 0,0,59,0,59,11,0,11,0,0,8,65,8,0,8,178,8,0,8,242,8
56 sD  '*** prepare GadgetBorders ***
57 vJ  xBorder gBor&(1),0,0,2,0,"",1,0&,0&
```

```
58 VB  gBorMem&=AllocRemember&(0, lenBor%*(gAnz%-1),65540&)
59 lX  RESTORE BordatAdd
60 8U  FOR i%=0 TO gAnz%-2
61 cw2   gBor&(i%+2)=gBorMem&+lenBor%*i%
62 Ce    CALL CopyMemQuick(gBor&(1),gBor&(i%+2),lenBor%)
63 iE    READ j%:POKE  gBor&(i%+2)+4,j% 'FrontPen
64 o2    READ j%:POKE  gBor&(i%+2)+7,j% 'Count
65 5L    READ j%:POKEL gBor&(i%+2)+8,BorXY&+j%
66 9EO NEXT
67 lD  BordatAdd:
68 Iu  'here we find the additional data
69 QD  'we make optimal use of Bordat: by using overlapping coord.
       data
70 LD  DATA 2,1,0, 2,1,0, 2,1,0, 2,1,0,  2,1,0,  2,1,0
71 LO  DATA 2,1,0, 1,5,0, 1,5,0, 2,2,18, 2,2,26, 2,2,34
72 U7  DATA 2,1,0, 1,5,0, 1,5,0, 1,5,0,  1,5,0,  1,5,0
73 cQ  DATA 1,5,0, 1,5,0, 1,5,0, 1,5,0
74 f9  '*** prepare SpecialInfos ***
75 G3  xStrInfo si11&,g11buf&,g11undo&,0,32,0,0&
76 nT  xStrInfo si12&,g12buf&,g12undo&,0,64,0,0&
77 cV  xStrInfo si13&,g13buf&,g13undo&,0,32,0,0&
78 Ms  xPropInfo si14&,"AUTOKNOB | FREEVERT",0,8200,0,24000
79 QY  '*** prepare Gadgets ***
80 FF  xGadget g&(1),0&,15,40,240,8,"","RELVERIFY | GADGIMMEDIATE"
       ,"BOOLGADGET",gBor&(1),0&,gTxt&(1),0&,0&,1,0&
81 vN  gMem&=AllocRemember&(0,(gAnz%-1)*lenGad%,65540&)
82 jG  RESTORE GaddatAdd
83 Vr  FOR i%=0 TO gAnz%-2
84 PG2   g&(i%+2)=gMem&+i%*lenGad%:g&=g&(i%+2)
85 8K    CALL CopyMemQuick(g&(1),g&,lenGad%)
86 fW    POKEL g&,g&(i%+1) 'link list
87 Y3    POKEL g&+18,gBor&(i%+2)
88 z8    POKEL g&+26,gTxt&(i%+2)
89 0t    POKEW g&+38,i%+2
90 U2    IF i%<7 THEN
91 Vj4     READ j%:POKEW g&+ 6,j% 'TopEdge
92 6R2   ELSEIF i%<14 THEN
93 RQ4     READ j%:POKEW g&+ 4,j% 'LeftEdge
94 Ym      READ j%:POKEW g&+ 6,j% 'TopEdge
95 VZ      READ j%:POKEW g&+ 8,j% 'Width
96 vU      READ j%:POKEW g&+10,j% 'Height
97 bq      READ j%:POKEW g&+12,j% 'flags
98 c6      READ j%:POKEW g&+16,j% 'type
99 Cv2   ELSE
100 xq4     POKEW g&+ 4, -70: POKEW g&+ 6, 20+16*(i%-14)
101 mF      POKEW g&+ 8,  60: POKEW g&+10, 12
102 Y3      POKEW g&+12,  16: POKEW g&+16,  1
103 bU2   END IF
104 lq0 NEXT
105 x2  GaddatAdd:
106 2p  DATA 49,58,67,76,85,94,103
107 Ie  DATA 10,-16,60,12,8,1
108 pY  DATA -150,-16,60,12,24,1
109 ws  DATA 66,13,65,8,0,4
110 fB  DATA 66,25,178,8,0,4
111 s4  DATA 66,-29,242,8,8,4
112 2k  DATA -125,40,30,74,16,3
113 ub  DATA 130,-16,60,12,8,1
114 H8  POKEL g&(11)+34,si11&
115 QE  POKEL g&(12)+34,si12&
116 ZK  POKEL g&(13)+34,si13&
117 iQ  POKEL g&(14)+34,si14&
118 xU  '------------------------------------
119 T4  '*** open window and read current directory ***
120 58  main:
121 6b  NewWindow w&,120,20,400,150,0,1,"CLOSEWINDOW | GADGETUP","W
        INDOWCLOSE | WINDOWDRAG | ACTIVATE | GImMEZEROZERO",g&(15+uA
        nz%),0&,"FileRequester",0&,0&,0,0,0,0,"WBENCHSCREEN"
122 Eg  XY&=AllocRemember&(0,20,65540&)
123 rP  RESTORE Bordat2
124 OX  FOR i%=0 TO 9:READ j%:POKEW XY&+2*i%,j%:NEXT
125 f1  xBorder bor&,0,0,1,0,"",5,XY&,0&
126 se  rp&=PEEKL(w&+50):CALL DrawBorder(rp&,bor&,8,37)
127 IP  Bordat2:
128 Sv  DATA 0,0,300,0,300,79,0,79,0,0
129 3i  CurrentDir v$
130 NE  loadfile$="":pat$="":pmod%=0
131 8L  GOSUB doReadDir
132 fb  'here the main loop starts. We can get out by closing the
133 VZ  'window ]] selecting OK ]] selecting Abbruch.
134 Sg  aus%=0
```

Listing 41. Der Filerequester. Sie benötigen außerdem Listing 38 bis 40, Listing 30 bis 36, Listing 23, Listing 17 und Listing 36.

```
135 5m  WHILE NOT aus%
136 yW2   GOSUB getTopLin
137 Mi    IF (toplin%<>lastlin%)AND(cnt%>8) THEN
138 9n4     lastlin%=toplin%
139 zp      ShowFiles toplin%
140 C52   END IF
141 LI    AskIMsg w&,GotMsg%,class%,code%,q%,IAdd&,mx%,my%
142 Fu    IF GotMsg%=0 GOTO loop
143 rR    IF class%=9 THEN  'we closed the window
144 Ln4     aus%=-1
145 Oi2   ELSEIF class%=6 THEN 'we released a Gadget (we check this
           instead of selecting a Gadget
146 v8N                       'because this is signalled when we p
                              ress RETURN in a string-Gadget!
147 CQ4     FOR i%=1 TO gAnz% 'we check the IAdd structure Member t
            o find out Gadget #
148 XW6       IF IAdd&=g&(i%) THEN j%=i%:i%=gAnz%
149 UZ4     NEXT
150 dA      IF (j%<9)AND(j%<=cnt%) THEN 'filename, but ignore emp
            ty fields
151 HH6       id%=PEEKW(IAdd&+38)
152 OZ        lin%=toplin%+id%-1
153 Qy        f$=fileName$(lin%)
154 XJ        IF ASC(f$)=1 THEN 'a subdirectory
155 dZ8         IF (RIGHT$(v$,1)<>":")AND(LEN(v$)>0) THEN v$=v$+
              "/"
156 ky          v$=v$+MID$(f$,2,LEN(f$)-7)
157 Yl          GOSUB doReadDir
158 oO6       ELSE 'a true file
159 MJ8         CALL CopyMem(SADD(f$),g13buf&,LEN(f$))
160 zJ          POKE g13buf&+i%-1,0
161 sn          fil$=f$
162 Ky          CALL RefreshGadgets(g&(13),w&,0&)
163 ZS6       END IF
164 Cz4     ELSEIF j%=9 THEN 'OK - Gadget
165 YV6       IF (RIGHT$(v$,1)<>":") THEN v$=v$+"/"
166 3E        loadfile$=v$+fil$
167 iA        aus%=-1
168 LK4     ELSEIF j%=10 THEN 'Abbruch-Gadget
169 kC6       aus%=-1
170 v74     ELSEIF j%=11 THEN 'Suffix
171 XJ6       GetStr pat$,(g11buf&)
172 OM        pmod%=3
173 o1        GOSUB doReadDir
174 o54     ELSEIF j%=12 THEN 'We entered a path (*WHY ???*)
175 dS6       GetStr v$,(g12buf&)
176 r4        GOSUB doReadDir
177 LC4     ELSEIF j%=13 THEN 'We entered a filename. This may be u
            sed if we want
178 ziM                       'to use the filerequester for saving
179 xF6       GetStr fil$,(g13buf&)
180 rZ4     ELSEIF (j%=14)AND(cnt%>8) THEN 'we clicked in the Scro
            llbar
181 6u6       GOSUB getTopLin:lastlin%=toplin%
182 gW        ShowFiles toplin%
183 4w4     ELSEIF (j%=15) THEN 'Parent dir
184 VI6       trunc%=0
185 1w        WHILE NOT trunc%
186 3v8         i$=RIGHT$(v$,1)
187 xD          IF (i$="/") OR (i$=":") THEN trunc%=-1
188 aA          IF i$<>":" THEN v$=LEFT$(v$,LEN(v$)-1)
189 UI6       WEND
190 5I        GOSUB doReadDir
191 Nf4     ELSEIF (j%>15) THEN 'we selected a new volume
192 Mu6       v$=unit$(j%-16)
193 8L        GOSUB doReadDir
194 4x4     END IF
195 5y2   END IF
196 Vt0 loop:
197 cQ  WEND
198 hU  CleanExit:
199 ET  CALL CloseWindow(w&)
200 Nx  CALL FreeRemember(0,-1)
201 kD  PRINT "You selected : ";loadfile$
202 Ov  END
203 OV  getTopLin:
204 s9  vp&=PEEKW(si14&+4):IF vp&<0 THEN vp&=vp&+65536&
205 gX  toplin%=vp&*(cnt%-8)/65536&
206 gI  RETURN
207 Do  doReadDir:
208 ak  ReadDir cnt%,v$,fileName$(),pat$,pmod%
209 FO  SortDir cnt%, fileName$()
210 vA  CALL CopyMem(SADD(v$),g12buf&,LEN(v$))
211 k3  POKE g12buf&+i%-1,0
212 62  '*** adjust PropGadget ***
213 LG  IF cnt%=0 THEN vbody&=65535& ELSE vbody&=65535&*(8/cnt%)
214 S1  IF vbody&>65535& THEN vbody&=65535&
215 PH  IF vbody&>32767 THEN vbody&=vbody&-65536&
216 aS  vbody%=vbody&:vpot%=0
217 gY  CALL ModifyProp(g&(14),w&,0&,5,&HFFFF,vpot%,&HFFFF,vbody%)
218 Ku  CALL RefreshGadgets(g&(14),w&,0&)
219 RI  '*** display first eight (or all) files ***
220 AU  ShowFiles 0
221 vX  RETURN
222 bu  '====== OWN SUBS ================
223 z8  SUB ShowFiles(po%) STATIC
224 sC  'SHARED rp&
225 wE  rp&=PEEKL(w&+50)
226 um  n&=FRE(0) 'garbage collection, if necessary
227 DI  FOR i%=1 TO 8
228 cV2   IF ASC(fileName$(po%+i%-1))=1 THEN j%=1 ELSE j%=0
229 yy    POKEL gTxt&(i%)+12,SADD(fileName$(po%+i%-1))+j%
230 ns0 NEXT
231 xY  CALL SetAPen(rp&,0)
232 X8  CALL RectFill(rp&,15,40,242,111)
233 Z9  CALL RefreshGadgets(g&(14),w&,0&)
234 oq  END SUB
235 td  '--------------------------------------
236 i9  SUB SortDir(count%, fin$()) STATIC
237 fv  'shellsort array of filenames
238 fj  g%=count%/2
239 4G  WHILE g%>0
240 YR2   FOR i%=g% TO count%
241 xa4     FOR j%=i%-g%-1 TO ug% STEP -g%
242 VZ6       IF UCASE$(fin$(j%))>UCASE$(fin$(j%+g%)) THEN
243 qp8         SWAP fin$(j%),fin$(j%+g%)
244 sl6       END IF
245 274     NEXT
246 382   NEXT
247 x0    g%=g%/2
248 RF0 WEND
249 35  END SUB
250 8s  '--------------------------------------
251 cJ  SUB GetStr(a$,add&) STATIC
252 a4  a$=""
253 Tu  FOR i%=0 TO 1000
254 W62   j%=PEEK(add&+i%)
255 8A    IF j%<>0 THEN a$=a$+CHR$(j%) ELSE i%=1000
256 DI0 NEXT
257 BD  END SUB
(C) 1988 M&T
```

Listing 41. (Schluß)

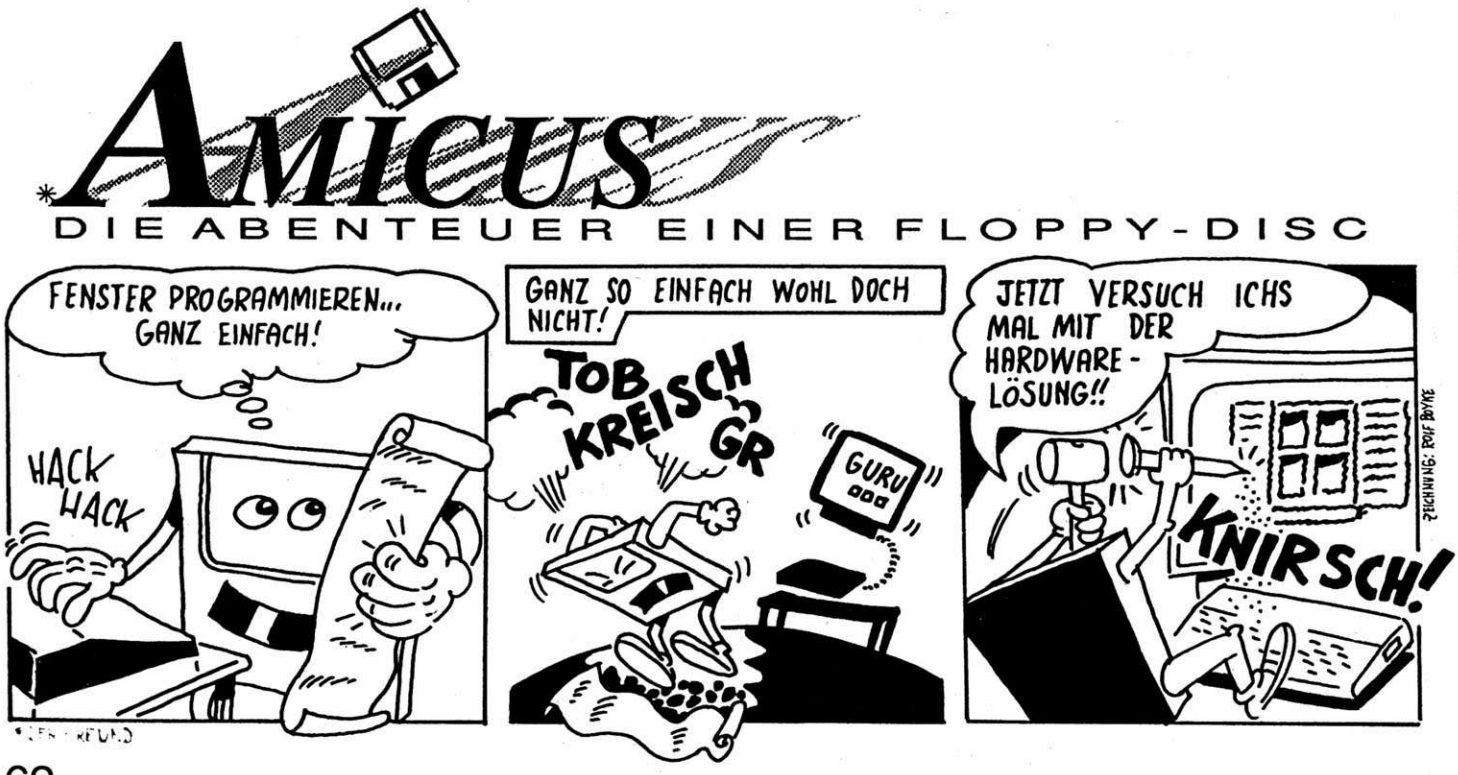
```
  1 Ej0 SUB PrintReq STATIC
  2 gu  'initialize sprite
  3 MY  sdat&=AllocRemember&(0,44,65538&)
  4 XU  FOR i%=0 TO 21:READ j%:POKEW sdat&+2*i%,j%:NEXT
  5 Ww  DATA 0,0,&H3E00,&H8080,&H6300,&H4100,&Hc180,&H2200,&H8880,&
        H1400
  6 2c  DATA &H9c80,&H0800,&H8880,&H1400,&Hc180,&H2200,&H6300,&H410
        0,&H3e00,&H8080,0,0
  7 zq  ssp&=AllocRemember&(0,12,65537&)
  8 mr  POKEL ssp&,sdat&:POKEW ssp&+4,9
  9 T0  WINDOW 2," ",(0,0)-(631,35),20
 10 fx  WINDOW OUTPUT 2
 11 G8  tit&=AllocRemember&(0,80,65537&)  'Platz fuer WindowTitle
 12 Ta  CALL ActivateWindow(WINDOW(7))
 13 cS  t$="PrintReq"+CHR$(0)
 14 so  CALL CopyMem(SADD(t$),tit&,LEN(t$))
 15 f1  CALL SetWindowTitles(WINDOW(7),tit&,-1)
 16 dh  PRINT "Roll screens with 'crsr-up' & 'crsr-down'; select sc
        reen in front with 'RETURN'"
 17 6j  DIM s&(10),w&(10),dy%(10)
 18 Aw  'read pointers of all screens
 19 kf  i%=0:s&(0)=PEEKL(WINDOW(7)+46)
 20 4N  pr1:
 21 t0  IF PEEKL(s&(i%))<>0 THEN
 22 cn2   s&(i%+1)=PEEKL(s&(i%))
 23 0Q    i%=i%+1
 24 Xf    dy%(i%)=52-PEEKW(s&(i%)+10)
 25 R3    CALL MoveScreen(s&(i%),0,dy%(i%))
 26 kc    GOTO pr1
 27 NG0 END IF
 28 Lh  IF i%=0 GOTO pr3
```

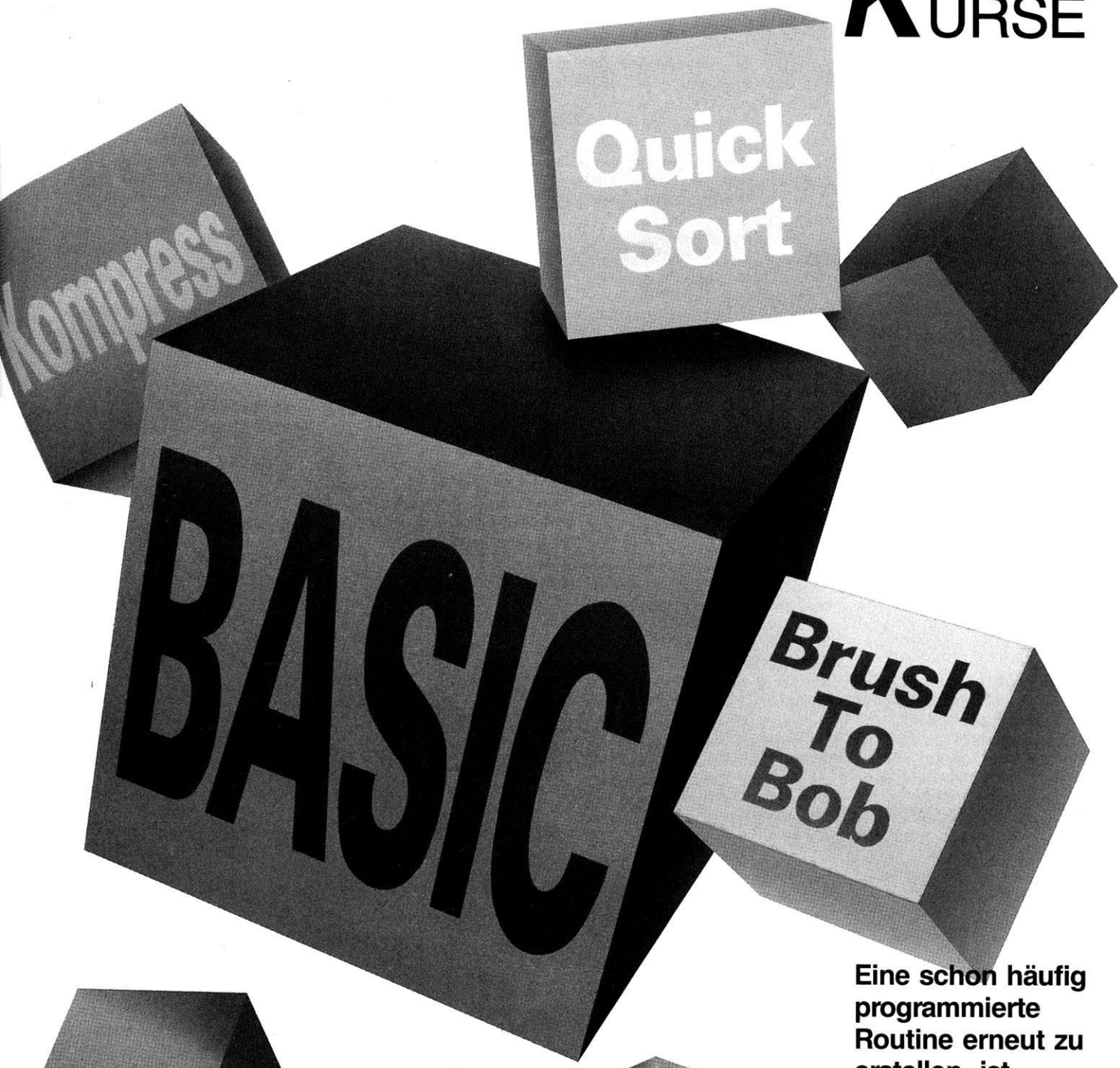
```
29 DN  j%=1:CALL ScreenToFront(s&(1))
30 Ic  pr2:
31 x7  i$="":WHILE i$="":i$=INKEY$:WEND
32 ys  IF i$=CHR$(28) THEN  'crsr-down
33 1L2   j%=j%+1:IF j%>i% THEN j%=0
34 YN    CALL ScreenToFront(s&(j%))
35 gD0 ELSEIF i$=CHR$(29) THEN 'crsr-up
36 9n2   j%=j%-1:IF j%<0 THEN j%=i%
37 bQ    CALL ScreenToFront(s&(j%))
38 9o0 ELSEIF i$=CHR$(13) THEN 'return
39 5z2   GOTO pr3
40 aT0 END IF
41 3w  GOTO pr2
42 8k  pr3:    'WindowTitle setzen
43 Wx  t$=LEFT$(t$,LEN(t$)-1)+"  SCREEN: ":t&=PEEKL(s&(j%)+26):i1%
       =0
44 e0  pr4:
45 L6  k%=PEEK(t&+i1%):IF k%<>0 THEN t$=t$+CHR$(k%):i1%=i1%+1:GO
       TO pr4
46 Ix  t$=t$+CHR$(0):CALL CopyMem(SADD(t$),tit&,LEN(t$))
47 BX  CALL SetWindowTitles(WINDOW(7),tit&,-1)
48 KQ  CLS
49 58  PRINT "OK!! Now select Window with 'crsr-up'/'crsr-down' &
       'RETURN'"
50 nM  PRINT "or press 'ESC' to select whole screen."
51 nL  'get sprite
52 U7  vp&=s&(j%)+44
53 2r  spn%=GetSprite%(ssp&, 1)
54 8c  'read pointers of all windows
55 e1  w&(0)=PEEKL(s&(j%)+4):i1%=0  'Screen.FirstWindow
56 uH  pr5:
57 tj  IF PEEKL(w&(i1%))<>0 THEN
58 WY2   w&(i1%+1)=PEEKL(w&(i1%))
59 jp    i1%=i1%+1
60 YU    GOTO pr5
61 vo0 END IF
62 NF  k%=0:w&=w&(0):GOSUB prs1  'show sprite
63 5T  pr6:
64 Ue  i$="":WHILE i$="":i$=INKEY$:WEND
65 AL  IF i$=CHR$(27) THEN
66 mP2   rp&=s&(j%)+84
67 aO    type%=0    'Screen
68 iq    GOTO pr10
69 It0 ELSEIF i$=CHR$(28) THEN  'crsr-down
70 zL2   k%=k%+1:IF k%>i1% THEN k%=0
71 3e    w&=w&(k%):GOSUB prs1
72 Ho0 ELSEIF i$=CHR$(29) THEN 'crsr-up
73 aZ2   k%=k%-1:IF k%<0 THEN k%=i1%
74 6h    w&=w&(k%):GOSUB prs1
75 kP0 ELSEIF i$=CHR$(13) THEN 'return
76 Ah2   type%=1    'Window
77 88    GOTO pr6a
78 C50 END IF
79 vs  GOTO pr6
80 sv  pr6a:    'WindowTitle setzen
81 Ow  t$=LEFT$(t$,LEN(t$)-1)+"  WINDOW: ":t&=PEEKL(w&(k%)+32):i1%
       =0
82 8U  pr6b:
83 SX  k1%=PEEK(t&+i1%):IF k1%<>0 THEN t$=t$+CHR$(k1%):i1%=i1%+1
       :GOTO pr6b
84 uZ  t$=t$+CHR$(0):CALL CopyMem(SADD(t$),tit&,LEN(t$))
85 n9  CALL SetWindowTitles(WINDOW(7),tit&,-1)
86 Wv  pr7:
87 HY  wof%=0
88 tX  wflg%=PEEKW(w&(k%)+26)    'niederwertiges WORD der Window-F
       lags
89 cV  IF (wflg% AND 1024) THEN wof%=104 'Abstand von GZZxy zu xy
90 uv  rp&=PEEKL(w&(k%)+50)
91 vO  pr10:
92 TN  'Einfache Abfrage ueber die Größe des Ausdruckes
93 SR  effects%=128
94 4A  CLS
95 MG  PRINT "Wie breit soll der Ausdruck werden (in mm; Return =
       1:1)"
96 Tp  INPUT "-->", i$:dbr&=VAL(i$)*39.37 'Umrechnung in 1/1000"
97 Kk  'einige Daten brauchen wir noch:
98 gC  cm&=PEEKL(vp&+4)     'ColorMap (Viewport hatten wir schon)
99 vc  modes&=PEEKW(vp&+32) 'ViewMode (z.B.: LACE, HAM, HIRES)
100 Ex IF type%=1 THEN          'Window-Rastport
101 OJ2  br%=PEEKW(w&(k%)+ 8+wof%)
102 Oq   h% =PEEKW(w&(k%)+10+wof%)
103 Wz0 ELSEIF type%=0 THEN      'Screen-Rastport
104 E42  br%=PEEKW(s&(j%)+12)
105 YC   h% =PEEKW(s&(j%)+14)
106 eX0 END IF
107 HE IF dbr&<>0 THEN    'Es wurde bereits eine Breite gewählt
108 BE2   dh&=dbr&*(h%/br%)
109 OD    effects%=effects% OR 1
110 N60 ELSE
111 dP2   dbr&=br%:dh&=h%
112 kd0 END IF
113 4e 'Jetzt noch die Druckdichte abfragen
114 zs CLS:PRINT "Geben Sie die Druckdichte an (1-4) :"
115 Ty pr12:
116 KU i$="":WHILE i$="":i$=INKEY$:WEND
117 S2 den%=VAL(i$):IF (den%<1)OR(den%>4) GOTO pr12
118 pY effects%=effects% OR (256*den%)
119 XK 'der RastPort ist bekannt, es kann also losgehen:
120 rL 'zuerst muessen wir ein Signal-Bit pachten und den dazugeho
       erigen
121 jk '(also den aktuellen) Task finden
122 O1 sig%=AllocSignal%(-1)
123 DQ tsk&=FindTask&(0)  'als Parameter wird ein Zeiger auf den T
       asknamen
124 35J                  'angegeben. 0 = aktueller Task
125 p00 'als zweites muessen wir einen MsgPort einrichten
126 9Y mp&=AllocRemember&(0,38,65537&)
127 L6 POKE  mp&+ 8, 4      'Node substructure ; Node.Type = NT_MS
       GPORT
128 Jf POKEL mp&+10, mp&+34 'den Node-Namen haengen wir gleich an
129 aB POKE  mp&+15, sig%
130 aX POKEL mp&+16, tsk&
131 BP POKEL mp&+20, mp&+24 'zur Verkettung der Msg-Ports. Dieser
       hier steht
132 u7 POKEL mp&+28, mp&+20 'alleine da. Daher zeigen die Verkette
       r aufeinander.
133 he POKE  mp&+34,ASC("P"):POKE mp&+35,ASC("R"):POKE mp&+36,ASC(
       "T") 'Name
134 LO CALL AddPort(mp&)
135 q5 'nun brauchen wir eine IODRPReq-struktur (IO-DumpRastPort-R
       equest)
136 Fb drp&=AllocRemember&(0, 62, 65537&)
137 n2 POKE  drp&+ 8,5      'NT_MESSAGE
138 gh POKEL drp&+14,mp&    'MsgPort
139 4Y POKEW drp&+18,12     'msg_length
140 8q POKEW drp&+28,11     'DumpRastPort
141 83 POKEL drp&+32, rp&   'Rastport-Adresse
142 PQ POKEL drp&+36, cm&   'ColorMap
143 hQ POKEL drp&+40, modes&  'Viewmode (fuer SPECIAL_ASPECT wicht
       ig)
144 23 POKEW drp&+44, 0    'x-offset von links oben
145 6S POKEW drp&+46, 0    'y-offset
146 fx POKEW drp&+48, br%  'Breite in Pixel
147 Wx POKEW drp&+50, h%   'Hoehe in Pixel
148 YP POKEL drp&+52, dbr& 'Druckbreite in Punkten
149 Rq POKEL drp&+56, 0    'Druckhoehe in Punkten, wird wegen SPEC
       IAL_ASPECT vom
150 Vh POKEW drp&+60, effects%                    'Druckertre
       iber berechnet
       =0
151 l1 'jetzt brauchen wir nur noch das device zu oeffnen und die
       IO zu starten
152 k3 dev$="printer.device"+CHR$(0)
153 QD er&=OpenDevice&(SADD(dev$),0,drp&,0) 'die Nullen sind :Unit
       # bzw. Flags
154 OZ er&=DoIO&(drp&)
155 uG WINDOW OUTPUT 1
156 o7 IF er&=1 THEN PRINT "Druck auf Ihren Wunsch abgebrochen"CHR
       $(7)
157 4S CALL CloseDevice(drp&)  'alle Resourcen (IOPort, MsgPort, S
       ignalBit)
158 kk CALL RemPort(mp&)       'wieder freigeben
159 jS CALL FreeSignal(sig%)
160 eH CALL FreeSprite(1)      'ebenso das Sprite
161 nT WINDOW CLOSE 2
162 rf FOR k%=0 TO i%
163 ki2   CALL MoveScreen(s&(k%),0,-dy%(k%))
164 jo0 NEXT
165 o2 EXIT SUB
166 T1 prs1:
167 8S CALL WindowToFront(w&)
168 Am x%=PEEKW(w&+4)+4:y%=PEEKW(w&+6)+4
169 UC CALL MoveSprite(vp&, ssp&, x%, y%)
170 61 RETURN
171 np END SUB
(C) 1988 M&T
```

Listing 42. »PrintReq« druckt einen beliebigen »RastPort« aus

```
 1 cCO DECLARE FUNCTION DoIO& LIBRARY
 2 OM  DECLARE FUNCTION OpenDevice& LIBRARY
 3 55  DECLARE FUNCTION AllocSignal% LIBRARY
 4 tu  DECLARE FUNCTION FindTask& LIBRARY
 5 iw  DECLARE FUNCTION GetSprite% LIBRARY
 6 Xi  DECLARE FUNCTION AllocRemember& LIBRARY
 7 fR  LIBRARY"exec.library"
 8 Gd  LIBRARY"intuition.library"
 9 6N  LIBRARY"graphics.library"
10 LU  RANDOMIZE TIMER
11 o3  Display 1
12 FF  WINDOW 1
13 Ub  CALL ActivateWindow(WINDOW(7))
14 GV  PRINT "Press any key to activate PrintReq ... "
15 e5  WHILE INKEY$="":SLEEP:WEND:CLS
16 5Q  PrintReq
17 KK  WINDOW 1
18 Zg  CALL ActivateWindow(WINDOW(7))
19 ik  WHILE INKEY$="":SLEEP:WEND
20 s6  Display 0
21 U4  CALL FreeRemember(0,-1)
22 61  END
23 zT  '==== OWN SUBS =============
24 io  SUB Display(mode%) STATIC
25 nz  IF mode%=1 THEN
26 mH2   wb&=PEEKL(WINDOW(7)+46)
27 9w    SCREEN 2,640,256,2,2
28 SO    WINDOW 21,"Boxes1",(20,60)-(420,180),16,2
29 W5    WINDOW OUTPUT 21
30 lr    s&=PEEKL(WINDOW(7)+46)
31 xO    t$="Boxes Screen"+CHR$(0)+"Lines Screen"+CHR$(0)+"Circles
         Screen"+CHR$(0)
32 20    st&=AllocRemember&(0,LEN(t$),65537&)
33 pe    CALL CopyMem(SADD(t$),st&,LEN(t$))
34 gT    t$=""
35 bP    vp&=s&+44
36 8u    CALL SetRGB4(vp&,0,0,0,0):CALL SetRGB4(vp&,1,15,5,2)
37 uG    CALL SetRGB4(vp&,2,15,9,3):CALL SetRGB4(vp&,3,15,11,6)
38 gx    POKEL(s&+26),st&
39 J5    CALL SetWindowTitles(WINDOW(7),-1&,st&)
40 yB    FOR i%=0 TO 10
41 Ka4     LINE(400*RND(1),120*RND(1))-(400*RND(1),120*RND(1)),1+((
           3*RND(1))MOD 3),b
42 lq2   NEXT
43 3t    WINDOW 22,"Boxes2",(220,30)-(570,210),16,2
44 nN    WINDOW OUTPUT 22
45 PB    CALL SetWindowTitles(WINDOW(7),-1&,st&)
46 4H    FOR i%=0 TO 10
47 N24     LINE(350*RND(1),180*RND(1))-(350*RND(1),180*RND(1)),1+((
           3*RND(1))MOD 3),b
48 rw2   NEXT
49 34    SCREEN 3,320,256,1,1
50 og    WINDOW 31,"Lines1",(190,70)-(290,200),16,3
51 yT    WINDOW OUTPUT 31
52 7D    s&=PEEKL(WINDOW(7)+46)
53 th    vp&=s&+44
54 Ex    CALL SetRGB4(vp&,0,6,15,2):CALL SetRGB4(vp&,1,3,10,0)
55 79    POKEL(s&+26),st&+13
56 Qi    CALL SetWindowTitles(WINDOW(7),-1&,(st&+13&))
57 Vm    FOR i%=0 TO 50
58 6e4     LINE(100*RND(1),130*RND(1))-(100*RND(1),130*RND(1)),1
59 272   NEXT
60 zT    WINDOW 32,"Lines2",(90,40)-(290,170),16,3
61 Ag    WINDOW OUTPUT 32
62 Wo    CALL SetWindowTitles(WINDOW(7),-1&,(st&+13&))
63 bs    FOR i%=0 TO 50
64 Kp4     LINE(200*RND(1),130*RND(1))-(200*RND(1),130*RND(1)),1
65 8D2   NEXT
66 2t    SCREEN 4,640,256,4,2
67 B2    WINDOW 41,"Circles1",(100,80)-(460,190),16,4
68 Lm    WINDOW OUTPUT 41
69 OU    s&=PEEKL(WINDOW(7)+46)
70 Ay    vp&=s&+44
71 mE    CALL SetRGB4(vp&,0,3,0,12):CALL SetRGB4(vp&,1,9,7,15)
72 nC    FOR i%=2 TO 15:CALL SetRGB4(vp&,i%,i%,i%,15):NEXT
73 io    POKEL(s&+26),st&+26
74 tF    CALL SetWindowTitles(WINDOW(7),-1&,(st&+26&))
75 Xk    FOR i%=0 TO 10
76 003    CIRCLE (300*RND(1),110*RND(1)),30+50*RND(1),1+((15*RND(1)
          )MOD 15)
77 KP2   NEXT
78 eY    WINDOW 42,"Circles2",(400,60)-(580,160),16,4
79 YO    WINDOW OUTPUT 42
80 zL    CALL SetWindowTitles(WINDOW(7),-1&,(st&+26&))
81 dq    FOR i%=0 TO 10
82 vP3    CIRCLE (180*RND(1),100*RND(1)),20+40*RND(1),1+((15*RND(1)
          )MOD 15)
83 QV2   NEXT
84 pd    CALL ScreenToFront(wb&)
85 yh0 ELSE
86 FL2   SCREEN CLOSE 2
87 LS    SCREEN CLOSE 3
88 RZ    SCREEN CLOSE 4
89 NG0 END IF
90 UW  END SUB
91 o4  '---- merged subs ----------------------------------------
       -------
(C) 1988 M&T
```

Listing 43. Das Hauptprogamm der Hardcopy-Routine. Sie benötigen außerdem Listing 42.





Module für Ihre Basic-Bibliothek

Eine schon häufig programmierte Routine erneut zu erstellen, ist äußerst lästig. Das muß nicht so bleiben: Effektives Gegenmittel ist eine Bibliothek mit Unterroutinen. Den Grundstock für diese Sammlung legen Sie mit diesem Kurs. Sie glauben gar nicht, wieviel Zeit und Arbeit Sie damit sparen.

Dieser Kurs legt den Grundstein zu einer umfassenden Bibliothek von Basic-Routinen. Zusätzlich vermittelt Ihnen der Artikel Grundlagen wichtiger Algorithmen. Viele der im folgenden besprochenen Algorithmen eignen sich zudem sehr gut für die Umsetzung in Maschinensprache. Doch das Schwergewicht des Kurses liegt darauf, die Arbeitsweise der verschiedenen Rechenvorschriften zu erläutern. Daher wurden alle Programme in Amiga-Basic geschrieben.

Eine der wichtigsten und häufigsten Aufgaben, die sich jedem Programmierer stellt, ist das Sortieren von Daten. Die gängigsten Sortierverfahren werden wir Ihnen vorstellen. Weitere Themen sind der IFF-Datenstandard, Grafik und die Datenverschlüsselung.

Eng verbunden mit den Algorithmen sind die Datenstrukturen — daher werden wir uns zudem mit Binärbäumen und ähnlichen »Gewächsen« befassen.

Ein leidiges Thema ist die Rechengenauigkeit und die Rechengeschwindigkeit. Auch zu deren Hintergründen sind einige Bemerkungen fällig. Um einen möglichst umfassenden Einsatz der erläuterten Algorithmen zu gewährleisten, werden alle Programme als Unterroutinen implementiert, die leicht in eigene Anwendungen eingebaut werden kön-

Was ist kleiner?

nen. Bevor wir nun anfangen, noch eine Vereinbarung: In allen Arrays wird das Element 0 nur dann benutzt, wenn gesondert darauf hingewiesen wird. Ansonsten laufen in unseren Arrays die Indizes immer von 1 bis N, wobei N die Anzahl der Feldelemente ist.

Gehen wir nun gleich in die vollen und beginnen mit dem Sortieren. Dazu nehmen wir an, wir haben einen Array mit n Elementen, die der Größe nach sortiert werden sollen. Hier treffen wir gleich auf die erste Schwierigkeit, denn es ist zwar einfach, Zahlen nach ihrer Größe zu ordnen. Aber schon bei Strings müssen wir uns erst darauf einigen, wann ein String kleiner oder größer als ein anderer ist. Auf Computern ist es allgemein üblich, hierfür die ASCII-Codierung zu benutzen. Sind nun die jeweils ersten Zeichen der Strings gleich, so werden die beiden nächsten Zeichen verglichen und so fort. Um nun auch Strings unterschiedlicher Länge vergleichen zu können, wird vereinbart, daß »kein Zeichen« kleiner ist, als alle anderen. Aus diesen Schwierigkeiten können Sie schon ersehen, daß es manchmal wünschenswert ist, im Programm selbst festzulegen, was »größer« oder »kleiner« genau bedeuten soll. Wichtig für einen Sortier-Algorithmus ist, daß in einem zu sortierenden Feld für jedes beliebige Paar von Elementen eindeutig festliegt, welches Element größer oder kleiner ist. Außerdem muß aus »a < b« und »b < c« folgen: »a < c«.

Doch nun zurück zum eigentlichen Sortieren.

Einfügen

Die einfachste (und bei weitem langsamste) Methode entspricht genau dem Vorgehen beim Sortieren eines Kartenspiels: Wir gehen davon aus, daß sich die erste Karte schon an der richtigen Position befindet. Nun nehmen wir die nächste Karte und prüfen, ob sie größer oder kleiner als die erste Karte ist.

Im ersten Fall fügen wir die zweite Karte vor der ersten ein, im zweiten Fall hinter der ersten. Nun betrachten wir die nächste Karte und fügen sie in die richtige Position ein. Dieses Verfahren setzen wir fort, bis wir bei der letzten Karte ankommen. Entsprechend dem beschriebenen Vorgehen heißt dieses Verfahren dann auch »Sortieren durch direktes Einfügen« (Listing 1). Da dieses Verfahren sehr langsam ist, ist es nur für Felder zu empfehlen, die höchstens 50 Einträge besitzen. Dies ist das einzige der langsamen Verfahren, das hier besprochen werden soll. Sollten Sie also wissen, um was es sich bei »Bubblesort« handelt, so versuchen Sie, dieses Verfahren schnell zu vergessen. Falls Sie es nicht wissen, hoffen wir, daß Sie es niemals herausfinden.

Heap-Sort

Unser Favorit unter den Sortieralgorithmen ist »Heap-Sort«, denn er verbindet Geschwindigkeit mit Übersichtlichkeit. Zur Erklärung der Vorgehensweise verwenden wir das Bild einer neu erstellten Fabrik. Insgesamt sollen in unserer Fabrik »n« Personen arbeiten. Zuerst stellen wir n/2 Arbeiter an. Als nächstes stellen wir deren unmittelbare Vorgesetzte ein, wobei für jeweils zwei Arbeiter ein Vorgesetzter vorgesehen ist. Nun sind aber die Unterschiede in der Qualifikation zwischen den beiden Arbeitern und deren Vorgesetzten nicht allzu groß. Daher entschließen wir uns, den jeweiligen Vorgesetzten mit seinen beiden Untergebenen zu vergleichen und dem besten der drei die Stelle des Vorgesetzten zu übergeben. Anschließend fahren wir fort, indem für jeweils zwei Vorgesetzte ein neuer, eine Stufe höher stehender Vorgesetzter engagiert wird. Wieder vergleichen wir jeden dieser Vorgesetzten mit seinen beiden unmittelbar Untergebenen und wählen den besten dieser drei aus. So fahren wir fort, bis wir den »Big Boss« eingestellt haben. Sobald dieser unter Vertrag ist, schicken wir ihn in Rente und wählen den besseren seiner beiden Untergebenen zum Nachfolger. Dann schicken wir auch diesen in Rente. Wir fahren nun so lange fort, bis auch der letzte Arbeiter im Ruhestand ist. Durch dieses Verfahren ist gewährleistet, daß jeweils der fähigste (»größte«) Mitarbeiter die Firma leitet. Die Reihenfolge der Pensionierung entspricht daher den Fähigkeiten (der Größe) der Mitarbeiter. Diese Vorgehensweise entspricht der Darstellung in Bild 1, das entsprechende Unterprogramm finden Sie in Listing 2. Es läßt sich sehr schön erkennen, woher der Name Binärbaum kommt, auch wenn unser Baum auf dem Kopf steht.

Bei kleinen Feldgrößen sind die Geschwindigkeitsunterschiede zwischen dem normalen Sortieren durch Einfügen und Heap-Sort nicht allzu groß.

```
main:
  WIDTH 75
  n = 50
  DIM a(n),b(n),c(n),k%(n)
  FOR i = 1 TO n
    a(i) = INT(1000 * RND)
    PRINT a(i);
  NEXT i
  IndexSort a(),k%()
  PRINT
  FOR i = 1 TO n
    PRINT a(k%(i));
  NEXT i
END

SUB InsertSort( s(1) ) STATIC ' Sortiere Array von Gleitpunktzahlen
  max% = INT(UBOUND(s))       ' Bestimme obere Feldgrenze, untere=1
  FOR j% = 2 TO max%          ' Bearbeite alle Feldelemente
    b = s(j%)                 ' Merken des aktuellen Feldelements
    i% = j% - 1               ' Das letzte der ber. sort. Elemente
    WHILE (i% > 0) AND (s(i%) > b) ' Solange El., die größer als
       s(i%+1) = s(i%)        ' das akt., schiebe diese nach Rechts
       i% = i% - 1
    WEND
    s(i%+1) = b               ' Fuege das akt. El. an der richtigen
  NEXT j%                     ' Stelle ein, dann bearbeite n. El.
END SUB
```

Listing 1. Die Subroutine »InsertSort« übernimmt das Sortieren durch direktes Einfügen

```
SUB HeapSort( s(1) ) STATIC
  max% = INT(UBOUND(s))
  l%   = max% \ 2 + 1
  r%   = max%
  WHILE 1
    IF l% > 1 THEN          ' Einstellungs-Phase
      l% = l% - 1
      h = s(l%)
    ELSE                    ' Pensionierungs-Phase
      h    = s(r%)          ' Am Ende des Arrays Platz schaffen
      s(r%) = s(1)          ' Schicke den >>Boss<< dorthin in Rente
      r% = r% - 1
      IF r% = 1 THEN        ' Die letzte Pensionierung
        s(1) = h            ' Der schlechteste Arbeiter
        EXIT SUB            ' Fertig
      END IF
    END IF                  ' Beim Einstellen und beim Pensionieren
    i% = l%                 ' muessen die besseren in die höheren
    j% = i% + i%            ' Positionen aufsteigen
    WHILE r% >= j%
```

Daher ist für kleine Felder die übersichtlichere Methode zu empfehlen. Bei größeren Feldern jedoch sind die Geschwindigkeitsunterschiede erheblich, hier ist also Heap-Sort der Vorzug zu geben. Eine weitere Sortiermethode ist das sogenannte Quick-Sort, das im Mittel noch etwas schneller als

Ohne Rekursion

Heap-Sort ist. Beachten Sie dabei jedoch die Aussage »im Mittel«, denn in ungünstigen Fällen liegt die Sortiergeschwindigkeit von Quick-Sort nur im Bereich des Sortierens durch Einfügen, während in günstigen Fällen Quick-Sort der schnellste der bekannten Sortieralgorithmen ist.

Interessanterweise liegt der ungünstigste Fall genau dann vor, wenn das Array schon in

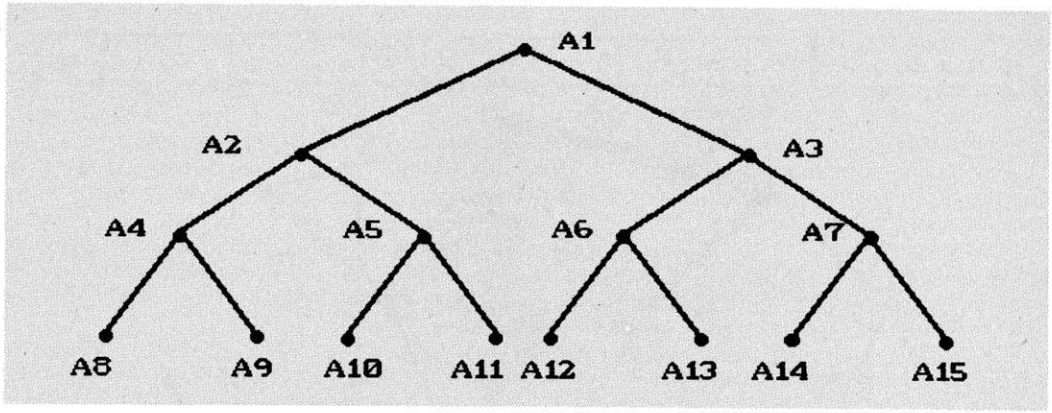


Bild 1. Das Prinzip des Heap-Sort

sortierter Form vorliegt. Ein weiterer spezieller Nachteil von Quick-Sort in Basic ist, daß Quick-Sort eigentlich ein rekursiver Algorithmus ist. Da aber Rekursion in Amiga-Basic leider nicht möglich ist, muß der zur Rekursion benötigte Datenstack simuliert werden. Diese Simulation macht das Unterprogramm noch unübersichtlicher, als es sowieso schon ist (Listing 3). Die grundlegende Idee von Quick-Sort ist, daß es günstiger ist, Array-Elemente über größere Entfernungen zu vergleichen und auszutauschen. Dazu wählt man willkürlich ein Element des Feldes (in unserem Algorithmus wählen wir das mittlere) und nennt es »x«. Dann wird das Feld von links beginnend so lange durchsucht, bis ein

```
      IF j% < r% THEN     ' mit den besseren auf der unteren
        IF s(j%) < s(j%+1) THEN j% = j% + 1     ' Stufe vergleichen
      END IF
      IF h < s(j%) THEN  ' degradiere h
        s(i%) = s(j%)
        i% = j%
        j% = i% + i%
      ELSE               ' h ist auf der richtigen Stufe
        j% = r% + 1
      END IF
    WEND
    s(i%) = h            ' setze h an seine richtige Position
  WEND
END SUB
```

Listing 2. Das Unterprogramm zum Heap-Sort demonstriert schnelles Sortieren

```
SUB QuickSort( a(1) ) STATIC
  DIM stack%(12,1)  ' keine Rekursion in Basic, simuliere dah. Stack
  n% = UBOUND(a)    ' jedoch seperat fuer linke und rechte Zerlegung
  s% = 1            ' des Arrays
  links% = 0        ' linker Teil
  rechts% = 1       ' rechter Teil
  stack%(1,links%) = 1  ' fange links
  stack%(1,rechts%) = n%  ' und rechts an
  REPEAT1:
    l% = stack%(s%,links%)
    r% = stack%(s%,rechts%)
    s% = s% - 1
    REPEAT2:
      i% = l%
      j% = r%
      x  = a((l%+r%) \ 2)   ' starte in der Mitte der akt. Zerlegung
      REPEAT3:
        WHILE a(i%) < x     ' finde das erste größere El. im linken
          i% = i% + 1       ' Teil (suche von unten)
        WEND
        WHILE x < a(j%)     ' finde erstes kleinere El. im rechten
          j% = j% - 1       ' Teil (suche von oben)
        WEND
        IF i% <= j% THEN    ' Tausche die beiden gef. Elemente
          SWAP a(j%),a(i%)
          i% = i% + 1
          j% = j% - 1
        END IF
      IF i% <= j% GOTO REPEAT3
      IF i% < r% THEN       ' Gehe 1 Stufe tiefer, zerlege erneut
        s% = s% + 1
        stack%(s%,links%) = i%
        stack%(s%,rechts%) = r%
      END IF
      r% = j%
    IF l% < r% GOTO REPEAT2
  IF s% <> 0 GOTO REPEAT1   ' Ist Stack leer? dann fertig
END SUB
```

Listing 3. »QuickSort« ist eigentlich ein rekursiver Algorithmus ist. Da aber Rekursion in Amiga-Basic leider nicht möglich ist, muß der zur Rekursion benötigte Datenstack simuliert werden.

```
SUB IndexSort( s(1), index%(1) ) STATIC
  n% = UBOUND(index%)      ' Ein normaler HeapSort, jedoch wird
  FOR j% = 1 TO n%         ' Array selbst nicht verändert, sondern
    index%(j%) = j%        ' die Rangfolge der Arrayelemente wird in
  NEXT j%                  ' einem sepereaten Feld gespeichert
  l% = n% \ 2 + 1
  ir% = n%
  WHILE 1
    IF l% > 1 THEN
      l% = l% - 1
      idx% = index%(l%)
      q = s(idx%)
    ELSE
      idx% = index%(ir%)
      q = s(idx%)
      index%(ir%) = index%(1)
      ir% = ir% - 1
      IF ir% = 1 THEN
        index%(1) = idx%
        EXIT SUB
      END IF
    END IF
    i% = l%
    j% = l% + l%
    WHILE j% < ir%
      IF j% < ir% THEN
        IF s(index%(j%)) < s(index%(j%+1)) THEN j% = j% + 1
      END IF
        IF q < s(index%(j%)) THEN
          index%(i%) = index%(j%)
          i% = j%
          j% = i% + i%
        ELSE
          j% = ir% + 1
        END IF
      WEND
      index%(i%) = idx%
  WEND
END SUB
```

Listing 4. »Index-Sort« hat viele Vorteile

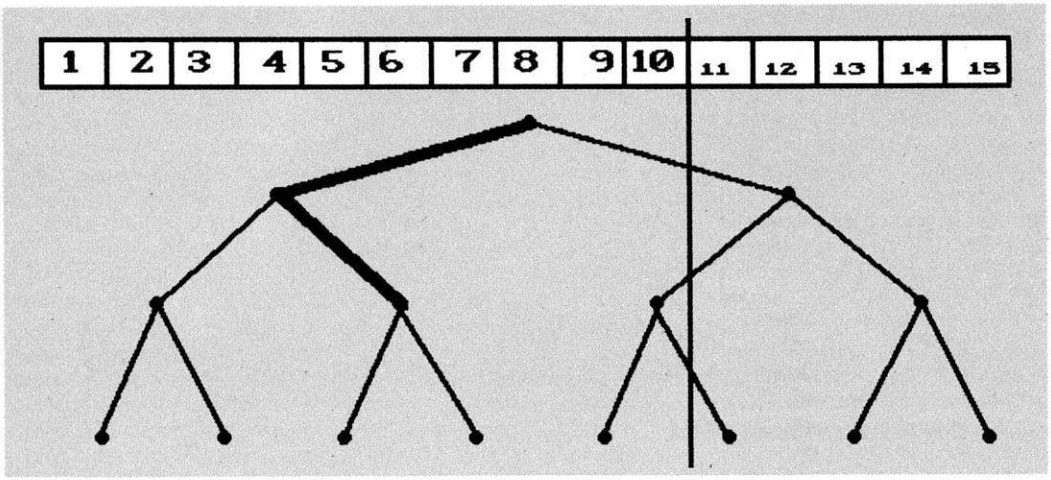


Bild 2. Mit Hilfe der Logarithmus-Funktionen wird berechnet, wieviele Vergleiche im ungünstigsten Fall durchzuführen sind. Hier ein Beispiel mit zehn Elementen.

Element gefunden wird, das größer als x ist. Anschließend wird das Array von rechts beginnend durchsucht, bis ein Element kleiner als x gefunden wird. Die beiden so gefundenen Elemente werden nun vertauscht und das Durchsuchen von beiden Seiten her wird fortgesetzt, bis man sich irgendwo trifft. Damit ist das Feld nun in zwei Teile zerlegt: Im linken Teil befinden sich nur Elemente, die kleiner als x sind; im rechten Teil sind nur Elemente, die größer als x sind. Nun wird das beschriebene Verfahren auf die beiden erzeugten Bereiche getrennt angewendet. In Sprachen wie C oder Pascal würde man einfach das Sortierprogramm erneut für die beiden Teilbereiche aufrufen. In Basic ist man gezwungen, eine Liste über die noch auszuführenden Zerlegungen zu unterhalten.

Vorteilhaftes Indexing

Einen Nachteil, den alle bisher vorgestellten Methoden haben, sollten wir nicht verschweigen: Der Datentyp, mit dem gearbeitet wird, ist innerhalb des Unterprogrammes festgelegt. So sortieren unsere Unterprogramme nur Felder von einfach genauen Gleitpunktzahlen. Will man etwa doppelt genaue Gleitpunktzahlen oder Strings sortieren, so muß das Unterprogramm an diesen Datentyp angepaßt werden.

Ein weiterer Nachteil ist, daß mit unseren Programmen nur jeweils ein Array sortiert werden kann und das ursprüngliche Array dabei auch noch durch das sortierte Feld ersetzt wird. Um diesen letzten Nachteil zu beheben, wenden wir einen einfachen Trick an (Listing 4): Zusätzlich zum zu sortierenden Feld übergeben wir an das Unterprogramm noch ein gleich großes Ganzzahlfeld. Dieses Ganzzahlfeld wird im Unterprogramm so vorbesetzt, daß das »i-te« Feldelement die Zahl i enthält. Nun wird ein ganz normales Sortierverfahren ausgeführt (in unserem Beispiel Heap-Sort), jedoch mit dem Unterschied, daß nicht mehr die Feldelemente selbst verglichen und ausgetauscht werden, sondern die Elemente des Ganzzahlfeldes. Nach dem Sortieren enthält also das erste Element des Ganzzahlfeldes den Feld-Index des kleinsten Elementes, während das letzte Element des Ganzzahlfeldes den Feld-Index des größten Elementes enthält. Damit finden wir also die Rangfolge unseres zu sortierenden Feldes. Als Feld-Index wird anstatt von i% der Wert index% (i%) unseres Ganzzahlfeldes index% benutzt. Diese Methode bietet zudem den Vorteil, daß es nun auch möglich ist, mehrere Felder »gleichzeitig« zu sortieren. Nehmen wir etwa an, Sie haben sich eine Adreßdatei angelegt. Im Feld »Name$« stehen die Namen ihrer Bekannten, im Feld »Adresse$« die zugehörigen Adressen, dann werden auch noch Felder für den Wohnort und die Telefonnummer benötigt. Die Daten Ihrer Bekannten können Sie nun nach den unterschiedlichsten Kriterien sortieren (Familienname, Wohnort,...). Doch wenn Sie ein Feld sortieren, müssen die zugehörigen Daten entsprechend mitsortiert werden. Mit dem kleinen Umweg über das Index-Feld ist das jedoch kein Problem mehr, da ja die eigentlichen Felder überhaupt nicht verändert werden. Die Daten werden sortiert, indem über das Index-Feld indirekt auf die Array-Elemente zugegriffen wird.

Die Hauptaufgabe des Sortierens ist, Daten leicht wiederfinden zu können. Daher benötigen wir noch ein Unterprogramm (Listing 5), das uns zu einem vorgegebenen Element sagt, ob das Element in unse-

Suchen und Finden

rem Array vorhanden ist und wenn ja, an welcher Stelle. Auch hier liegt es in unserem Interesse, das Element möglichst schnell zu finden, doch ein Blick auf Bild 1 hilft uns weiter: Wir sehen zuerst in der Mitte des Arrays nach und vergleichen, ob das dortige Element größer oder kleiner als unser gesuchtes ist. Im ersten Fall sehen wir uns das mittlere Element der linken Hälfte an, im anderen Fall das mittlere Feld der rechten Hälfte. Mit diesem Element verfahren wir wieder wie gehabt. Auch hier gibt es ein kleines Problem: Feldindizes sind ganze Zahlen, doch es ist wenig wahrscheinlich, daß beim fortwährenden Halbieren der Intervallgröße als Ergebnis immer ganze Zahlen auftreten. Die damit verbundenen Rundungsfehler können dazu führen, daß ein Feldelement übersehen wird. Daher wird in unserem Unterprogramm mit Feldindizes gearbeitet, die sich durch Addition und Subtraktion von Zweierpotenzen ergeben. Da es damit aber möglich wird, über die Feldgrenzen »hinauszuschießen«, müssen noch entsprechende Sicherheitsabfragen eingebaut werden. Nach unserer Vereinbarung sind nur Indizes zwischen 1 und n sinnvoll. Wird daher das angegebene Element im Array nicht gefunden, so wird eine 0 an das aufrufende Programm übergeben. In Listing 5 bleiben noch einige Feinheiten zu erklären: Mit Hilfe der Logarithmus-Funktionen wird berechnet, wie viele Vergleiche im ungünstigsten Fall durchzuführen sind. In Bild 2 ist ein Beispiel mit zehn Elementen wiedergegeben. Um ein beliebiges Element aus diesen zehn herauszufinden, sind höchstens drei

```
SUB LookFor( INDEX% , array(1) )STATIC
  n% = CINT(UBOUND(array)) ' obere Grenze eines geordneten Feldes
k% = INT(LOG(n%)/LOG(2))   ' Maximalanzahl von nötigen Vergleichen
i% = 1                     ' Berechen mittleres Element als 2**R%
FOR idx% = 1 TO k%         ' >>Mitte<< bezieht sich hier auf
  i% = 2 * i%              ' nächstgrößere Zweierpotenz
NEXT idx%
idx% = i%                  ' Index fuer mittleres Element
k% = i%                    ' Beachte: dies ist eine Zweierpotenz
WHILE k% > 0 AND array(idx%) <> in  ' k% zählt die Vergleiche
  k% = k% \ 2              ' Division durch 2 geht hier ohne Rest
  IF array(idx%) < in THEN ' zu suchendes El. in oberer Hälfte
    i% = i% + k%
  ELSE                     ' zu suchendes El. in unterer Hälfte
      i% = i% - k%
    END IF
    IF i% > n% THEN          ' obere Hälfte bezieht s. a. 2er-potenz
      idx% = n%              ' das tatsächliche Feld kann kl. sein
    ELSE                     ' In diesem Fall Vergl. einfach mit dem
      idx% = i%              ' größten Element
    END IF
  WEND
  IF array(idx%) = in THEN   ' Falls gefunden, liefere Position
    index% = idx%
  ELSE                       ' sonst liefere 0
    index% = 0
  END IF
END SUB
```

Listing 5. Das Unterprogramm »LookFor« teilt mit, ob das Element in unserem Array vorhanden ist

Vergleiche nötig, wobei diese Zahl bis zu 15 Elementen gültig bleibt. Ab der nächsten Zweierpotenz (also 16) sind dann bis zu vier Vergleiche nötig.

Um Daten wiederzufinden, gibt es neben dem Sortieren noch einen anderen Zugang: Die Schlüsseltransformation, oder in englischer Bezeichnung »Hashing«. Der Vorteil von Hashing ist, daß es schneller als Sortieren ist. Der Nachteil dabei ist, daß das benutzte Feld eine festgelegte Größe

Daten finden mit »Hashing«

hat, die nur verändert werden kann, indem der Datensatz in ein neues, größeres Feld kopiert wird. Es ist also ein Feld mit bestimmter Länge vorgegeben, wobei die Länge aus später noch zu erläuternden Gründen eine Primzahl sein sollte. In dieses Feld sollen nun Daten so eingetragen werden, daß sie möglichst schnell wieder gefunden werden. Dazu denken wir uns ein (möglichst einfaches) Verfahren aus, um die Position zu berechnen, an der das Element im Feld gespeichert werden soll. Für Strings bietet es sich etwa an, die ASCII-Werte der einzelnen Buchstaben aufzusummieren. Die so erhaltene Zahl ist in den meisten Fällen größer als der maximal erlaubte Feldindex. Daher benutzen wir noch die Modulo-Funktion (MOD), um einen gültigen Feldindex zu erhalten. Das Verfahren ist also in der Tat sehr einfach: Berechne mit Hilfe einer fest vorgegebenen (Schlüssel-) Funktion die Position im Array und speichere das Element dort ab. Um es wiederzufinden ist einfach erneut der Schlüssel zu berechnen, schon haben wir unser Element.

Doch ein Haar finden wir noch in dieser Suppe: die beiden Strings »MEIER« und »EIMER« bestehen aus denselben Buchstaben, daher ist auch die Summe der ASCII-Werte gleich. Das heißt aber auch, daß diese Strings in unserem Feld an der gleichen Position abgelegt werden. Eine derartige Situation wird beim Hashing als Kollision bezeichnet. Eine einfache Lösung wäre, einfach den nächsten freien Platz zu benutzen, wobei man sich das Feld als einen Kreis vorstellt, das heißt auf die letzte Position folgt wieder die erste. Das Problem ist nur, daß sich beim Aufsummieren die ASCII-Werte in einer bestimmten Gegend häufen werden. Damit wird es auch sehr wahrscheinlich, daß der nächste Platz ebenfalls nicht frei ist. Dies würde dazu führen, daß mit der Suche nach einem freien Platz viel Zeit verschwendet wird. Diese »lineares Sondieren« genannte Methode ist also nicht unbedingt zu empfehlen. Als nächst komplizierteres Verfahren bietet sich das »quadratische Sondieren« an. Hier wird nach der i-ten Kollision versucht, das Element i*i Positionen hinter der ursprünglich berechneten Position unterzubringen. Aber auch das quadratische Sondieren stellt einen Kompromiß dar. Denn während wir beim linearen Sondieren beim Einfügen von Elementen sichergehen können, einen vorhandenen freien Platz auch zu finden, ist es beim quadratischen Sondieren möglich, keinen freien Platz mehr zu finden — obwohl noch freie Plätze vorhanden sind. Es läßt sich jedoch folgendes mathematisch beweisen: Benutzt man als Array-Länge eine Primzahl, so wird sicher mindestens der halbe Array nach einer freien Stelle zum Einfügen durchsucht.

In der Praxis wirkt sich diese Beschränkung auf die halbe Tabelle nicht sonderlich aus,

```
main:
  n% = 100
  HashDim n%
  PRINT n%
  DIM table$(n%)
  INPUT s$
  WHILE s$ <> ""
    HashWrite m% , s$ , table$()
    PRINT m%,table$(m%),
    HashRead m% , s$ , table$()
    PRINT m%,table$(m%)
    INPUT s$
  WEND
  INPUT s$
  HashRead m% , s$ , table$()
  PRINT m%,
  IF m% <> 0 THEN PRINT table$(m%)
END
```

```
SUB HashDim( n% ) STATIC     ' Bestimme kleinste Primzahl > n%
  DIM Prim%(2*n%)            ' da DIM d. Hash-Arr. Primzahl sein muss
  FOR i% = 2 TO n%           ' Sieb d. Eratosthenes: Markiere alle
    IF Prim%(i%) = 0 THEN    ' Zahlen, die keine Primzahlen sind.
      j% = i% + i%
      WHILE j% <= n%
        Prim%(j%) = 1        ' Alle Vielfachen v. Primzahlen sind
        j% = j% + i%         ' keine Primzahl
      WEND
    END IF
  NEXT i%                    ' Alle Primzahlen bis 2*n% markiert, nun
  i% = n%                    ' Durchsuche Array ab n% und gib erste
  WHILE Prim%(i%) = 1 AND i% < 2*n%  ' gefundene Primzahl aus
    i% = i% + 1
  WEND
  ERASE Prim%                ' Loesche Array, Speicherplatz sparen
  n% = i%
END SUB

SUB HashWrite( message% , s$ , strarray$() ) STATIC  ' schreibe
                                                     's$ in feld
  IF s$ = "" THEN   ' Leerstring braucht nicht eingeordnet werden
    message% = 0
    EXIT SUB
  END IF
  n% = CINT(UBOUND(strarray$))  ' bestimme ob. Feldgrenze, untere=1
  ok = 0                        ' wenn = 1, dann fertig
  h% = 0                        ' Wert der Hashfunktion
  d% = 1                        ' quadratisches Sondieren
  FOR i& = SADD(s$) TO SADD(s$) + LEN(s$)  ' Bestimme Wert der
    h% = h% + PEEK(i&)                     ' Hash-Funktion fuer
  NEXT i&                                  'uebergebenen String
  h% = (h% MOD n%) + 1          ' Reduziere auf Feldgrenzen 1 ... n%
  k% = h%                       ' Merken der errechneten Position
  REPEAT.HashWrite:             ' Solange s$ noch nicht eingetragen
    IF strarray$(h%) = "" THEN  ' freier Platz gefunden
      strarray$(h%) = s$        ' dann trage s4 dort ein
      ok = 1
    ELSE                        ' Feld schon besetzt
      h% = ((h% + d%) MOD n%) + 1  ' Versuche nächstes, durch
      d% = d% + 2                  ' quadratisches Sondieren
                                   ' berechnetes Feldelement
    END IF          ' Beachte: (i+1)**2 = i**2 + 2*i + 1
  IF ok = 0 AND h% <> k% GOTO REPEAT.HashWrite
  IF ok = 0 THEN          ' >>ganzen<< Array durchsucht
    message% = 0          ' aber kein freier Platz gefunden
  ELSE
    message% = h%         ' s$ an Position h% gespeichert
  END IF
END SUB

SUB HashRead( message% , s$ , strarray$() ) STATIC
  IF s$ = "" THEN   ' Leerstring braucht nicht gesucht zu werden
    message% = 0
    EXIT SUB
  END IF
  n% = CINT(UBOUND(strarray$))  ' obere Feldgrenze (=Primzahl)
  ok = 0
  h% = 0
  d% = 1
  FOR i& = SADD(s$) TO SADD(s$) + LEN(s$)  ' Hashfunktion=Summe der
    h% = h% + PEEK(i&)                     ' ASCII-Werte d. Strings
  NEXT i&
  h% = (h% MOD n%) + 1          ' Reduziere auf Feldgrenzen 1 ... n%
  k% = h%                       ' Merken der errechneten Position
  REPEAT.HashRead:              ' Solange nicht gefunden
    IF strarray$(h%) = s$ THEN  ' gefunden
      message% = h%             ' an Position h%
      ok = 1
    ELSEIF strarray$(h%) = "" THEN  ' Leerstring gefunden, also
      message% = 0                  ' kein passender Eintr. vorh.
      ok = 1
    ELSE                            ' versuche nächstes, durch
      h% = ((h% + d%) MOD n%) + 1   ' quadratisches Sondieren
      d% = d% + 2                   ' bestimmtes Feldelement
    END IF
  IF ok = 0 AND k% <> h% GOTO REPEAT.HashRead  ' wei-
ter, bis gefund.
  IF ok = 0 THEN message% = 0          ' oder ganzes Feld durchsucht
END SUB
```

Listing 6. Mit diesem Unterprogramm stehen Ihnen drei Hashing-Funktionen zur Verfügung

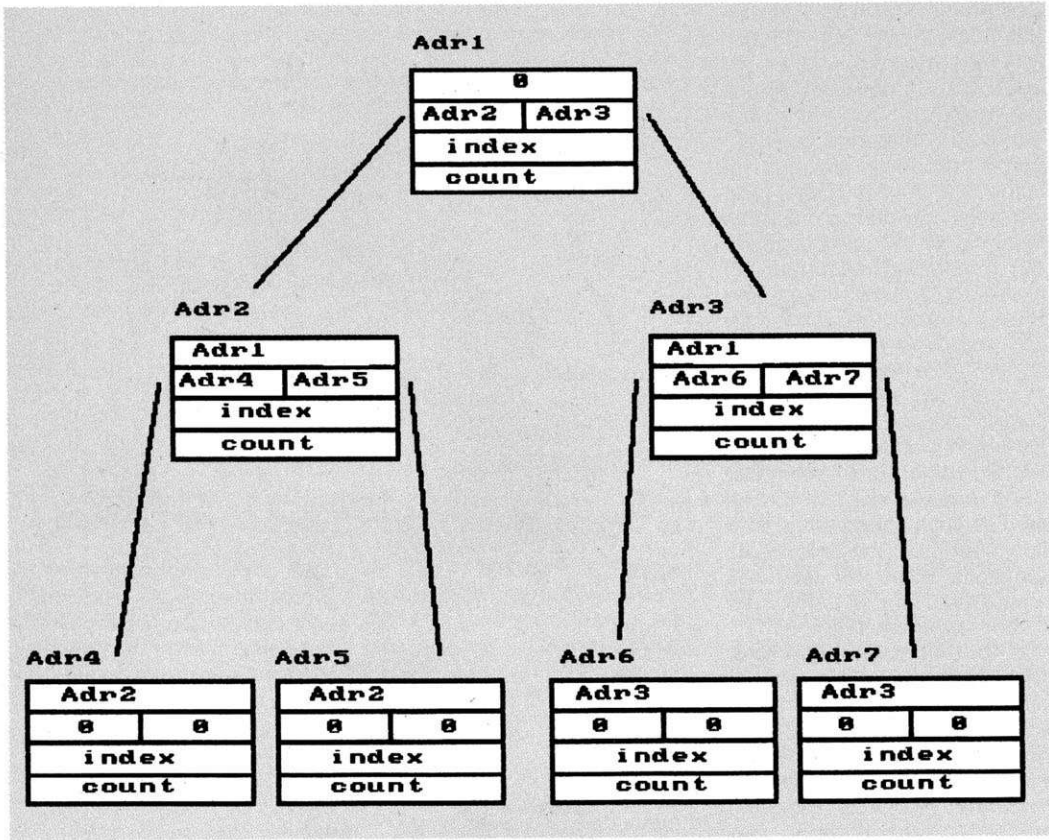


Bild 3. Ein Binärbaum mit dynamischer Speicherbelegung

denn so viele Kollisionen kommen nur sehr selten dann vor, wenn die Tabelle schon fast voll ist. Daher ist es aber auch geraten, die Hashing-Tabelle höchstens zu drei Vierteln zu füllen und ansonsten die Daten in eine größere Tabelle umzukopieren. Als Beispiel aus der Praxis für die Anwendung der Hashing-Methode kann der Amiga dienen: Das Betriebssystem verwaltet die Directory-Einträge durch Hashing. Mit Listing 6 stehen Ihnen drei Hashing-Funktionen zur Verfügung. Das Unterprogramm »HashDim« erwartet als Argument eine Ganzzahlvariable. Zu dem Wert dieser Variablen wird dann die nächst größere Primzahl berechnet und in der Variablen abgelegt. Somit läßt sich das Hashing-Feld leicht dimensionieren. Die zweite Funktion übernimmt den String »s$« und versucht ihn im an »strarray$« übergebenen Stringfeld abzuspeichern. Falls dies gelingt, wird die Feldposition, an der die Zeichenkette abgespeichert wurde, übergeben. Falls der String nicht abgespeichert werden konnte, wird eine Null an das rufende Programm übergeben.

Das dritte Unterprogramm arbeitet umgekehrt: es wird versucht, den »String s$« im Feld »strarray$« zu finden. Falls er gefunden wird, so wird die Position im Feld als Ergebnis geliefert, falls die Zeichenkette nicht gefunden werden konnte, wird eine 0 übergeben. Es bleibt nur noch eine Anmerkung zum quadratischen Sondieren: Nach der i-ten Kollision wird versucht, den String i2 Positionen nach der mit der Hash-Funktion errechneten Position h abzuspeichern (alles Modulo die Array-Größe). Die Berechnung dieser Position wurde in unseren Programmen etwas effizienter gestaltet: Anstatt jeweils aufs neue nach jeder Kollision ein Quadrat zu berechnen, wird die neue Position h+(i+1)2 aus der aktuellen Position h+i2 ermittelt, indem der binomische Lehrsatz

$$(i + 1)^2 = i^2 + 2*i + 1$$

benutzt wird. Die Variable d wird mit 1 vorbesetzt und nach jeder Kollision um 2 erhöht, was somit dem Term 2*i + 1

Dynamische Bäume

entspricht, der nur noch auf die aktuelle Position, die ja h+i2 entspricht, aufaddiert werden muß.

Bis jetzt ist schon mehrmals der Begriff »Binärbaum« gefallen. Es ist langsam an der Zeit, sich mit diesen etwas näher zu beschäftigen. Da Binärbäume ihrer Konzeption nach rekursive Datenstrukturen sind, ist ihre Behandlung in Basic etwas umständlich. Daher beschränken wir uns hier nur auf die einfachsten Fälle. Der wesentliche Vorteil von Binärbäumen ist ihre dynamische Speicherverwaltung. Wir müssen auf die Speicherverwaltungsfunktionen des Betriebssystems zurückgreifen. In unserem Beispielprogramm »search« (Listing 7) wird eine Ganzzahl in das Unterprogramm übernommen. Nun wird versucht, diese Zahl in dem bereits vorhandenen Binärbaum, der etwa wie Bild 3 aussieht, zu finden. Wird die Zahl gefunden, so wird einfach mitgezählt, das wievielte Mal sie auftrat. Wesentlich interessanter wird es dagegen, wenn die Zahl im Baum noch nicht vorhanden ist. Wollten wir die Zahl in einem gewöhnlichen Array einfügen, so müßten wir alle nachfolgenden, größeren Elemente um eine Position nach rechts verschieben, wobei wir auch noch beachten müßten, ob das Array nicht etwa zu klein wird. Ganz anders steht die Sache bei dynamischen Binärbäumen. Hier reservieren wir uns mit Hilfe der »AllocMem«-Funktion des Betriebssystems einfach den entsprechenden Datenbereich und schreiben unsere Daten in diesen Speicher. Nun müssen im Binärbaum nur noch die Verweise auf die Vorgänger und Nachfolger entsprechend angepaßt werden. Dabei ist folgendes zu beachten: Die Ver-

```
CLEAR,100000&,8000
DECLARE FUNCTION AllocMem&( size& , type& ) LIBRARY
LIBRARY "dh0:Basic/BMAP/exec.library"


main:
  WIDTH 75
  id = 1
  OPEN "ram:BinTree" FOR INPUT AS id
  ByteCount id , r&
  CLOSE id
  FreeTree r&
  LIBRARY CLOSE
END
```

```
SUB ByteCount( file , root& ) STATIC
  length& = CLNG(LOF(file))
  lgth% = length& \ 32767
  lgrst% = length& AND 32767
  MakeRoot root&
PRINT root&
  FOR i% = 1 TO lgth%
    f$ = INPUT$( 32767, file )
    fptr& = SADD(f$)
    FOR j% = 0 TO 32766
      search PEEK(fptr&+j%) , root&
    NEXT j%
  NEXT i%
  f$ = INPUT$( lgrst% , file )
  fptr& = SADD(f$)
  FOR j% = 0 TO lgrst%-1
    PRINT PEEK(fptr&+j%);
    search PEEK(fptr&+j%) , root&
  NEXT j%
  PRINT
END SUB
```

weise auf die Vorgänger werden beim Suchen eigentlich nicht benötigt.

Da Basic jedoch keine Rekursion erlaubt, erweist es sich für das spätere Freigeben des Speichers als nützlich, die Referenzen auf die Vorgänger zu implementieren. Hatte der Vorgänger überhaupt keine Nachfolger, so muß in diesen nur eingetragen werden, daß nun das neue Element sein Nachfolger ist. Das neue Element selbst hat dann keine Nachfolger, für sie werden daher Nullen eingetragen. Anders dagegen, wenn das neue Element zwischen zwei Elemente »eingeklinkt« werden soll. Jetzt ist nicht nur die entsprechende Referenz im Vorgänger abzuändern, sondern auch die Referenz im jetzigen Nachfolger des neuen Elements (siehe Bild 4).

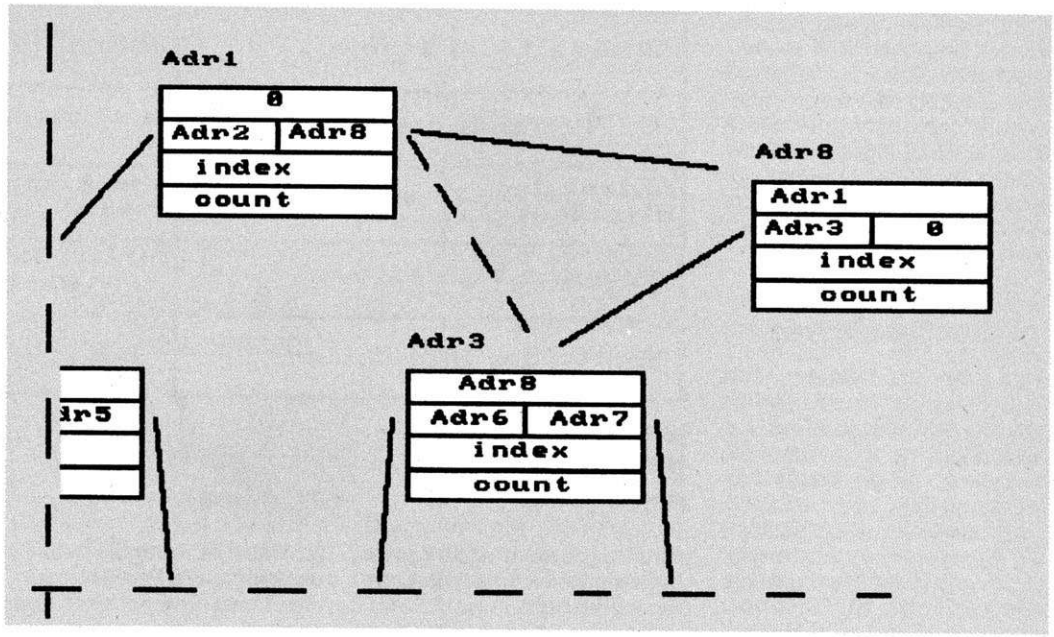


Bild 4. Das Einfügen in einen dynamischen Binärbaum

```
SUB search( x% , root& ) STATIC
  l%  = 0         ' C-Datenstruktur: struct TreeNode{ TreeNode* l ;
  r%  = 4         '                                   TreeNode* r ;
  v%  = 8         '                                   TreeNode* v ;
  index% = 12     '                                   short index ;
  count% = 14     '                                   short count ;
  size&  = 16     '                                 } ;
  p2& = root&          ' Start
  p1& = PEEKL(p2&+r%) ' rechter Nachfolger
  d%  = 1
  WHILE p1& <> 0& AND d% <> 0
    p2& = p1&
    IF x% < PEEKW(p1& + index%) THEN
      p1& = PEEKL(p1& + l%)
      d% = -1
    ELSEIF x% > PEEKW(p1& + index%) THEN
      p1& = PEEKL(p1& + r%)
      d% = 1
    ELSE
      d% = 0
    END IF
  WEND
  IF d% = 0 THEN
    POKEW (p1& + count%), PEEKW(p1& + count%) + 1
  ELSE
    p1& = AllocMem&( size& , 0 )
    IF p1& = 0 THEN ERROR 14
    POKEW (p1& + index%),x%
    POKEL (p1& + l%),0
    POKEL (p1& + r%),0
    POKEL (p1& + v%),p2&
    POKEW (p1& + count%),1
    IF d% < 0 THEN
      POKEL (p2& + l%),p1&
    ELSE
      POKEL (p2& + r%),p1&
    END IF
  END IF
END SUB
```

Listing 7. Die Routine »search« sucht eine Zahl in einem vorhandenen Binärbaum. Wird die Zahl gefunden, so wird gezählt, das wievielte Mal sie auftrat.

```
SUB FreeTree( root& ) STATIC
  l% = 0
  r% = 4
  v% = 8
  index% = 12
  count% = 14
  size& = 16
  p& = root&
  h& = p&
  WHILE PEEKL(p& + v%) <> 0 OR
        PEEKL(p& + r%) <> 0 OR PEEKL(p& + l%) <> 0
    WHILE PEEKL(p& + l%) <> 0
      p& = PEEKL(p& + l%)
    WEND
    WHILE PEEKL(p& + r%) <> 0 AND PEEKL(p& + l%) = 0
      p& = PEEKL(p& + r%)
    WEND
    IF PEEKL(p& + l%) = 0 THEN
      h& = PEEKL(p& + v%)
PRINT PEEKW(p& + index%),PEEKW(p& + count%)
      FreeMem p& , size&
      IF PEEKL(h& + l%) = p& THEN
        POKEL (h& + l%),0
      ELSE
        POKEL (h& + r%),0
      END IF
      p& = h&
    END IF
  WEND
  FreeMem p& , size&
PRINT p&
END SUB
```

Listing 8. mit »FreeTree« wird der von »search« reservierte Speicher wieder an das Betriebssystem zurückgegeben. Dabei kann in unserem Beispiel nur der gesamte Speicher auf einmal zurückgegeben werden.

```
SUB MakeRoot( root& ) STATIC
  l% = 0
  r% = 4
  v% = 8
  index% = 12
  count% = 14
  size&  = 16
  root& = AllocMem&( size& , 0 )
  IF root& = 0 THEN ERROR 14
  POKEL (root& + l%), 0
  POKEL (root& + r%), 0
  POKEL (root& + v%), 0
  POKEW (root& + index%), 0
  POKEW (root& + count%), 0
END SUB
```

Listing 9. Das dynamische Baumprogramm »MakeRoot« können Sie einsetzen, um beispielsweise die Häufigkeit von ASCII-Zeichen in einer Datei zu bestimmen

Im zweiten Unterprogramm (Listing 8) wird der von »search« reservierte Speicher wieder an das Betriebssystem zurückgegeben. Dabei kann in unserem Beispiel nur der gesamte Speicher auf einmal zurückgegeben werden. Es ist also nicht möglich, gezielt Einträge aus

Freiheit für die Speicher

dem Baum zu entfernen. Das liegt daran, daß das Entfernen aus dem Binärbaum nur bei Elementen einfach ist, die höchstens einen Nachfolger besitzen. Bei zwei Nachfolgern, die ja bei Binärbäumen der Normalfall sind, ist hierfür ein ziemlicher Aufwand nötig, falls man — wie alle Anwender von Amiga-Basic — auf Rekursion verzichten muß. Um nun den gesamten Speicher freizugeben, verfolgen wir ein einfaches Konzept: Solange ein linker Nachfolger vorhanden ist, gehe zu diesem. Solange kein linker Nachfolger vorhanden ist, gehe zum rechten Nachfolger. Ist auch dieser nicht vorhanden, so sind wir in der linken unteren Ecke des Baumes bei einem Element angekommen, das keine Nachfolger besitzt. Ein derartiges Element heißt »Blatt«, im Gegensatz zu Elementen mit Nachfolger, die »Knoten« heißen. Ein Blatt läßt sich nun verhältnismäßig leicht vom Baum abschütteln, indem einfach der zugehörige Speicher freigegeben wird und im Vorgänger die Referenz auf das Blatt durch 0 ersetzt wird. Danach gehen wir zurück zum Vorgänger und setzen das Verfahren fort. So wird der Baum von links unten nach rechts oben entfernt, bis am Ende auch die Wurzel (das Element ohne Vorgänger) ausgerissen wird. Die in unserem Programm auftretenden PRINT-Anweisungen ermöglichen es Ihnen, die Arbeitsweise von »FreeTree« zu verfolgen. Sie können diese Befehle ohne Gefahr entfernen.

Unser dynamisches Baumprogramm können wir nun einsetzen, um zum Beispiel die Häufigkeit von ASCII-Zeichen in einer Datei zu bestimmen (Listing 9). Unser Beispiel behandelt nur 1-Byte-Zeichen, daher ließe es sich durch einen 255 Einträge langen Array leichter realisieren, doch schon bei Worten zeigt unser Vorgehen Vorteile: Es wird nur Speicherplatz für Worte benötigt, die auch tatsächlich auftreten. Die Datenstrukturen,

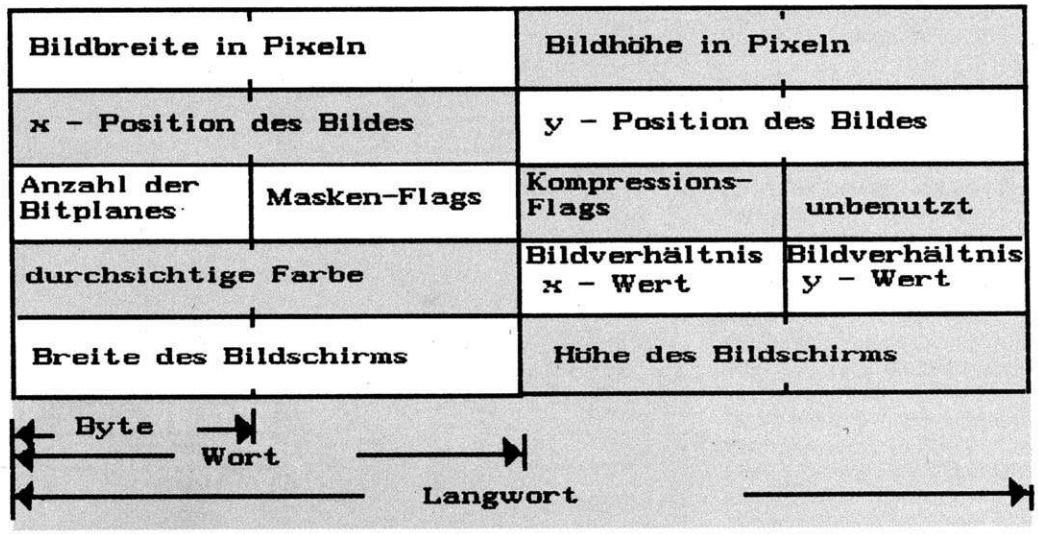


Bild 5a (oben) und 5b (unten). Der BMHD-Eintrag enthält die Informationen über die Größe des Bildes, die Position auf dem Bildschirm, die Anzahl der Bitplanes und anderes mehr

Farb-Offset	
Daten - Offset	
Anzahl der Bitplanes	
Breite des BOB's in Pixeln	
Höhe des BOBs in Pixeln	
Flags	Bitplanes, in die BOB gezeichnet werden soll
Schatten-Auswahl	<Daten der 1. Bitplane>
.....	<nächste Bitplane> <muß bei gerader Adr. beg.>
.....	

die in unseren Beispielprogrammen behandelt werden, enthalten der Einfachheit halber nur ganze Zahlen. Wollen Sie auch Strings behandeln, müssen Sie sich entweder auf eine maximale Stringlänge beschränken, oder aber das Ende eines Strings kenntlich machen, etwa nach C-Konvention durch eine abschließende

Wer zählt die Worte

Null, oder aber Sie speichern die Stringlänge mit ab. All diese Verfahren bedeuten aber, daß die unterschiedlichen Elemente des Binärbaumes auch eine unterschiedliche Länge besitzen. Aus diesem Grund wurden die Referenzen auf die Nachfolger und Vorgänger, die ja als Adressen eine feste Länge von 4 Byte haben, an den Anfang der Datenstruktur gestellt. Statt des »index« genannten Elementes könnte hier die Länge des Strings stehen, gefolgt vom String selbst, der mit Hilfe der CopyMem-Funktion aus der »exec.library« und der SADD-Funktion übertragen werden kann:

```
CopyMem( SADD(s$),
(startadr& + offset%),
LEN(s$) )
```

Allerdings sind bei diesem Vorgehen auch noch die Vergleichsfunktionen zu implementieren, die entscheiden ob ein String größer oder kleiner als ein anderer ist. Wie Sie sehen, bietet sich Ihnen hier ein reiches Betätigungsfeld für eigene Experimente.

Auf die Bestimmung der Häufigkeiten von ASCII-Zeichen werden wir später bei der Behandlung der Datenverschlüsselung noch einmal zurückkommen, doch nun ist zur Erholung erst einmal ein Themenwechsel angesagt.

Vielleicht haben Sie sich auch schon darüber geärgert, daß Amiga-Basic so gar nichts vom IFF-Datenstandard wissen will. Da hat man nun die schönsten Brushes mit einem Zeichenprogramm, wie etwa Deluxe-Paint erstellt, doch die Basic-Animationsobjekte zeigen nur ihre kalte Schulter. Daher ist es höchste Zeit, diesen Zustand zu ändern. Dazu müssen wir uns aber erst einmal näher damit beschäftigen, in welchem Format Basic die Informationen für einen BOB (Blitter Object) erwartet und in welcher Form diese Informationen in einem IFF-Datenfile vorliegen. Dabei werden wir allerdings nicht bis zu den letzten

Feinheiten des IFF-Datenstandards vordringen. Wir sind schon zufrieden, wenn es uns gelingt, einen »Brush« (Pinsel) als BOB einzulesen.

IFF-Daten sind »Langwort-orientiert«, das heißt die wichtigen Schlüsseldaten nehmen jeweils eine Länge von 4 Byte ein. Nachdem also eine Datei geöffnet wurde, die vermutlich einen Pinsel enthält, läßt sich sehr leicht überprüfen, ob dies auch tatsächlich der Fall ist.

Allen IFF-Dateien gemeinsam ist, daß das erste Langwort aus den vier Buchstaben »FORM« besteht, gefolgt von der Anzahl der Bytes in dieser FORM. (Dies ist eigentlich nicht ganz richtig, es kann auch IFF-Dateien geben, die aus mehreren FORMS bestehen, diese Dateien beginnen mit dem Schlüsselwort »LIST«). »SMUS« für musikalische Daten und natürlich »ILBM« für Grafiken. Da wir an Brushes interessiert sind, muß auch unsere Datei dieses Kennwort enthalten. Da dieses Kennwort nur dazu dient, die Dateiart zu erkennen, folgt ihm keine Längenangabe, sondern unmittelbar das nächste Kennwort, welches im Normalfall BMHD ist. Diesem folgt nun wieder ein Langwort, das angibt aus wieviel Bytes dieser Eintrag besteht. Eine Bedingung für Kennworte ist, daß sie immer bei einer geraden Byte-Anzahl in der Datei stehen müssen. Ist daher die Länge eines Eintrages eine ungerade Byte-Anzahl, so muß nach diesem Eintrag noch ein zusätzliches Null-Byte folgen. Der BMHD-Eintrag enthält die Informationen über die Größe des Bildes, an welcher Stelle des Bildschirmes es erscheinen soll, die Anzahl der Bitplanes und so fort (Bild 5). Einige dieser Informationen können von unserem Leseprogramm ignoriert werden, wie Bildposition, Bildschirmgröße oder auch das Bildverhältnis. Sicher benötigt werden dagegen die Informationen über die Bildgröße, die Anzahl der Bit-Ebenen und natürlich auch, ob die Bilddaten in komprimierter Form oder unkomprimiert vorliegen.

Erholsame Unterbrechung

Falls wir die Farbinformationen, die im separaten Eintrag CMAP abgespeichert sind, benutzen wollen, müssen wir auch wissen, welche Farbe als durchsichtig zu gelten hat.

Ein CMAP-Eintrag hat folgendes Format: Nach dem Kennwort folgt wie üblich die Länge in Byte. Die CMAP enthält jeweils 1 Byte für den Rot-, Grün- und Blauanteil der einem Farbregister zugeordneten Farben. Da der Amiga 32 Farbregister besitzt, kann eine CMAP also bis zu 96 Byte lang sein. Jedes Farbregister des Amiga kann Werte zwischen 0 und 15 für jede Farbe annehmen, während in einem Byte 256 verschiedene Werte dargestellt werden können. Es ist nun zu beachten, daß in einer CMAP die Farbinformation jeweils in den oberen 4 Bit eines Byte abgespeichert sind. Dies ermöglicht die Kompatibilität zu späteren Rechnergenerationen, die wesentlich mehr Farben darstellen können.

Eine ILBM-IFF-Datei kann noch weitere Kennworte enthalten, die für unsere Zwecke jedoch unwichtig sind, selbstverständlich mit Ausnahme des noch ausstehenden »BODY«-Eintrages, denn dieser enthält die eigentlichen Bildinformationen (Bild 6). Gehen wir erst einmal davon aus, daß die Bildinformationen in unkomprimierter Form vorliegen. Die Anzahl der einzulesenden Bytes konnten wir wieder dem BODY folgenden Langwort entnehmen und nun kann es also munter ans Einlesen gehen: Die Bildbreite kann natürlich relativ beliebig sein, doch es werden immer ganze Worte

Wie sieht ein CMAP-Eintrag aus?

eingelesen. Die Anzahl der Worte pro Bildzeile und pro Bitplane ergibt sich daher einfach zu:

```
w% = (Breite% + 15) \ 16
```

Im BODY-Eintrag liegt nun jeweils eine Zeile der ersten Bitplane, gefolgt von einer Zeile der zweiten Bitplane, und so fort, bis zur letzten Bitplane. Dann kann noch, abhängig von einem Flag im BMHD-Eintrag, eine Zeile einer Masken-Bitplane folgen, die von uns jedoch ignoriert wird — was aber nicht heißen soll, daß sie nicht eingelesen werden muß, falls sie vorhanden ist.

Nun kann mit der nächsten Zeile der ersten Bitplane fortgefahren werden. Dieses Spiel setzt sich so lange fort, bis sämtliche Daten des BODY-Eintrages eingelesen sind (Listing 10).

Wie Sie sich leicht vorstellen können, steigt bei größeren Bildern die Anzahl der zu lesenden Daten sehr rasch an, daher gibt es für die BODY-Einträge noch die Möglichkeit, daß sie in komprimierter Form vorliegen. Dies kann das Einlesen wesentlich beschleunigen. Dafür müssen die Daten nach dem Einlesen aber erst wieder »ausgepackt« werden, was natürlich auch heißt, daß man wissen muß, wie sie »ein-

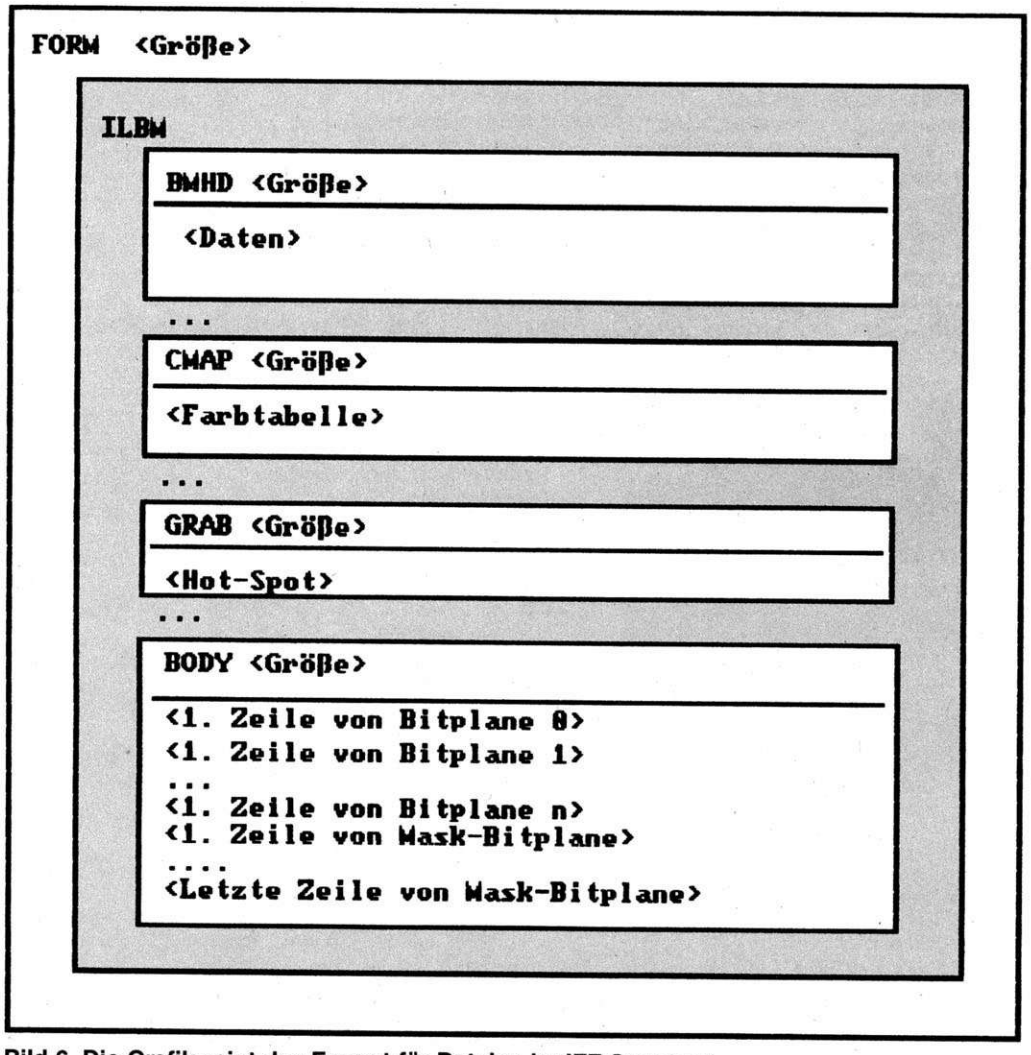


Bild 6. Die Grafik zeigt das Format für Dateien im IFF-Standard

gepackt« wurden. Das von Deluxe-Paint benutzte Einpack-Verfahren erfordert folgende Maßnahmen zum Auspacken:
1. Lies ein Byte ein.
2. Falls dieses Byte eine Zahl n zwischen 0 und einschließlich 127 darstellt, können die nächsten N+1 Bytes unverändert eingelesen werden. Liegt die Zahl n jedoch zwischen 129 und 255, so ziehe von dieser Zahl 1 ab und bilde dann die exklusive oder-Verknüpfung mit 255 (Programmier-Profis würden diese Operation als Bilden des Zweier-Komplements bezeichnen). Nennen wir diese Zahl n', so wissen wir nun, daß das nächste Byte n'+1 mal zu nehmen ist. Einen Sonderfall bildet die noch fehlende Zahl 128, sie besagt, daß nichts zu tun ist.

Bei dieser Art der Datendekompression ist noch zu beachten, daß immer nur separate Zeilen einer Bitplane komprimiert wurden, das heißt eine Datenkompression erstreckt sich nicht über mehrere Zeilen oder Bitplanes (Listing 11).

Nachdem wir nun das IFF-Datenformat kennengelernt haben, müssen wir noch herausfinden, in welcher Form Basic nun die Daten eines BOBs benötigt. Auch hier können wir für unsere einfachen Anwendungen einige Daten ignorieren, indem wir sie etwa mit Null vorbesetzen.

So zum Beispiel gleich die beiden ersten Langworte. Wichtig werden nun die nächsten Langworte, die (in dieser Reihenfolge) die Anzahl der Bitplanes, die Breite und Höhe des Objektes in Pixeln enthalten. Nun folgen weitere 2 Byte, die für Flags reserviert sind. Hier sind für uns die Bits 4 und 5 interessant, die bestimmen, ob die Farbe 0 durchsichtig oder schwarz ist, und ob der Bildhintergrund gesichert werden soll, bevor der BOB gezeichnet wird. Da wir diese beiden Fragen bejahen, schreiben wir hierhin also immer die Zahl 24. Das nächste Wort entscheidet, in welchen Bit-Ebenen der BOB gezeichnet werden soll, hier wählen wir immer alle aus, wobei sich der benötigte Zahlenwert zu

```
2^(Bitebenen) - 1
```

berechnet. Das nächste Wort wird von uns wieder ignoriert (auf Null gesetzt).

Nun folgen die einzelnen Bitebenen, im Gegensatz zu IFF-Dateien jedoch Bitebene nach Bitebene. Damit können wir nun Brushes aus Deluxe-Paint in Basic-BOB's umwandeln, wobei jedoch die folgende Einschränkung gilt:

Da in unserem Unterprogramm »BrushToBOB« die Daten in einen String eingelesen werden, dürfen die BOB-Daten 32767 Byte nicht übersteigen. Die Verwendung derart großer BOBs ist allerdings auch nicht sinnvoll, da diese dann unangenehm stark flackern.

Datenverschlüsselung

Ein interessantes Gebiet der Informatik ist die Datenverschlüsselung. Besonders Banken, Versicherungen und andere Unternehmen in Bereichen mit vertraulichen Daten haben ein großes Interesse daran, daß ihre Daten nicht einfach von jedermann gelesen werden können. Um einen kleinen Einblick in dieses Gebiet zu erhalten, sollen hier einige einfache Algorithmen für die Datenverschlüsselung besprochen werden.

Eine der ersten Methoden der Verschlüsselung war, das Alphabet um eine bestimmte Anzahl von Buchstaben zu verschieben. Als Beispiel kann etwa Tabelle 1 dienen. Diese Art der Chiffrierung wurde bereits von Julius Cäsar benutzt, doch ist diese Methode in unseren heutigen Zeiten viel zu unsicher. Die nächst kompliziertere Methode ist, die Buchstaben

Cäsar-Chiffre

```
abcdefghijklmnopqrstuvwxyz
yzabcdefghijklmnopqrstuvwx
```

Vignère-Quadrat

```
abcdefghijklmnopqrstuvwxyz
zabcdefghijklmnopqrstuvwxy
yzabcdefghijklmnopqrstuvwx
xyzabcdefghijklmnopqrstuvw
wxyzabcdefghijklmnopqrstuv
vwxyzabcdefghijklmnopqrstu
uvwxyzabcdefghijklmnopqrst
tuvwxyzabcdefghijklmnopqrs
stuvwxyzabcdefghijklmnopqr
rstuvwxyzabcdefghijklmnopq
qrstuvwxyzabcdefghijklmnop
pqrstuvwxyzabcdefghijklmno
opqrstuvwxyzabcdefghijklmn
nopqrstuvwxyzabcdefghijklm
mnopqrstuvwxyzabcdefghijkl
lmnopqrstuvwxyzabcdefghijk
klmnopqrstuvwxyzabcdefghij
jklmnopqrstuvwxyzabcdefghi
ijklmnopqrstuvwxyzabcdefgh
hijklmnopqrstuvwxyzabcdefg
ghijklmnopqrstuvwxyzabcdef
fghijklmnopqrstuvwxyzabcde
efghijklmnopqrstuvwxyzabcd
defghijklmnopqrstuvwxyzabc
cdegghijklmnopqrstuvwxyzab
bcdefghijklmnopqrstuvwxyza
```

Tabelle 1.
Eine der ersten Methoden der Verschlüsselung war, das Alphabet zu verschieben

```
main:
  WIDTH 75
  id = 1
  INPUT "Geben Sie eine Brush-Datei an: ", FileName$
  SCREEN 2,320,255,5,1
  WINDOW 2,,,,2
  cl% = 1
  OPEN "dh0:DeluxePaint/Brush/"+FileName$ FOR INPUT AS id
  BrushToBOB id , Bob$ , cl%
  CLOSE id
  OBJECT.SHAPE 1,Bob$
  OBJECT.X 1,50
  OBJECT.Y 1,50
  OBJECT.ON
  OBJECT.VX 1,20
  OBJECT.VY 1,10
  OBJECT.START 1
  WHILE MOUSE(0) = 0 : SLEEP : WEND
  OBJECT.OFF
  OBJECT.CLOSE
  WINDOW CLOSE 2
  SCREEN CLOSE 2
END

SUB GetPalette( s$ , l& , t% ) STATIC
  FOR i% = 0 TO l&\3-1
    r = PEEK(SADD(s$)+3*i%)   / 255
    g = PEEK(SADD(s$)+3*i%+1) / 255
    b = PEEK(SADD(s$)+3*i%+2) / 255
    PALETTE i%,r,g,b
  NEXT i%
  r = PEEK(SADD(s$)) /255
  g = PEEK(SADD(s$)+1) / 255
  b = PEEK(SADD(s$)+2) / 255
  PALETTE t%,r,g,b
END SUB
```

Listing 10. »GetPalette« holt die vorhandene Farbpalette

```
SUB BrushToBOB( FileID , Bob$ , Clr% ) STATIC
  a$ = "" ' Puffer zum Einlesen der unterschiedlichen Kennungen
  l& = 0  ' Länge der jeweiligen Datenabschnitte
  d$ = "" ' Datenabschnitte
  Flen& = 0 ' verbleibende Länge der FORM
  w% = 0  ' Breite des Bobs in Pixeln
  h% = 0  ' Höhe des Bobs in Pixeln
  b% = 0  ' Anzahl der Bitebenen
  m% = 0  ' Mask definiert
  c% = 0  ' Datenkompression
  BDim& = 0 ' Anzahl der Worte (2 Byte) fuer eine Bitplane

  a$ = INPUT$(4,FileID)
  IF a$ <> "FORM" THEN ERROR 54
  Flen& = CVL(INPUT$(4,FileID))
  a$ = INPUT$(4,FileID)
  Flen& = Flen& - 4
  IF a$ <> "ILBM" THEN ERROR 54
  WHILE a$ <> "BMHD" AND Flen& > 0
    a$ = INPUT$(4,FileID)
    l1& = CVL(INPUT$(4,FileID))
    l& = l1& + (l1& AND 1&)
    d$ = INPUT$(l&,FileID)
    Flen& = Flen& - l& - 8
  WEND
  IF a$ <> "BMHD" THEN ERROR 54
  w% = CVI(MID$(d$,1,2))
  ww% = 2*((w%+15)\16)
  h% = CVI(MID$(d$,3,2))
  b% = CVI(MID$(d$,9,2))
  m% = b% AND &HFF
  b% = b% \ &H100
  c% = CVI(MID$(d$,11,2)) \ &H100
  t% = CVI(MID$(d$,13,2))
  BDim& = h% * ww%
  Bob$ = MKL$( 0& ) + MKL$( 0& ) + MKL$(CLNG(b%))
  Bob$ = Bob$ + MKL$(CLNG(w%)) + MKL$(CLNG(h%))
```

```
Bob$ = Bob$ + MKI$( 32+24 ) + MKI$( CINT(2**b% - 1) ) + MKI$( 0 )
Bob$ = Bob$ + SPACE$(b%*BDim&)
WHILE a$ <> "BODY" AND Flen& > 0
  a$ = INPUT$(4,FileID)
  ll& = CVL(INPUT$(4,FileID))
  l& = ll& + (ll& AND 1&)
  d$ = INPUT$(l&,FileID)
  IF (a$ = "CMAP") AND (Clr% = 1) THEN
    CALL GetPalette( d$ , ll& ,t% )
  END IF
  Flen& = Flen& - l& - 8
WEND
IF a$ <> "BODY" THEN ERROR 54
IF c% = 0 THEN
  o& = 1
  FOR j% = 0 TO h%-1
    FOR i% = 0 TO b%-1
      MID$(Bob$,i%*BDim&+j%*ww%+27,ww%) = MID$(d$,o&,ww%)
      o& = o& + ww%
      NEXT i%
      IF m% = 1 THEN
        o& = o& + ww%
      END IF
    NEXT j%
  ELSE
    Cvrt% = 0
    FOR j% = 0 TO h%-1
      FOR i% = 0 TO b%-1
        UnPack d$ , x$ , ww% , Cvrt%
        MID$(Bob$,i%*BDim&+j%*ww%+27,ww%) = x$
      NEXT i%
      IF m% = 1 THEN
        UnPack d$ , x$ , ww% , Cvrt%
      END IF
    NEXT j%
  END IF
END SUB

SUB UnPack( d$ , x$ , w% , cv% ) STATIC
  i% = 1
  x$ = SPACE$(w%)
  WHILE i% <= w%
    p% = PEEK(SADD(d$) + cv%)
    cv% = cv% + 1
    IF p% < 128 THEN
      p% = p% + 1
      MID$(x$,i%,w%) = MID$(d$,cv%+1,p%)
      cv% = cv% + p%
    ELSEIF p% <> 128 THEN
      j% = PEEK(SADD(d$)+cv%)
      cv% = cv% + 1
      p% = ((p%-1) XOR &HFF) + 1
      MID$(x$,i%,w%) = STRING$(p%,j%)
    ELSE
      p% = 0
    END IF
    i% = i% + p%
  WEND
END SUB
```

Listing 11. »BrushToBOB« und »Unpack« helfen bei der Grafikwandlung

nicht einfach zu verschieben, sondern beliebig durcheinander zu würfeln. Jedoch haben diese Methoden einen entscheidenden Nachteil: Jeder Buchstabe des Alphabets wird durch genau einen Buchstaben des Codes ersetzt. Nun treten aber die einzelnen Buchstaben in unserer Sprache unterschiedlich häufig auf (Tabelle 2). Daher müssen wir einen derart verschlüsselten Text nur in den Computerspeicher einlesen, dabei (etwa mit unserem Programm »Byte-Count«) die verschiedenen Buchstaben mitzählen und dann die durch die Gesamtzahl der Buchstaben dividierten Häufigkeiten mit unserer Tabelle vergleichen — schon ist der Text entschlüsselt. Nebenbei bemerkt läßt sich mit dieser Methode der Häufigkeitsbestimmung auch automatisch bestimmen, ob ein vorliegender (unverschlüsselter) Text in englisch, deutsch, Basic, C oder Assembler geschrieben ist. An der Verschlüsselungsmethode von Cäsar können Sie jedoch bereits eine wichtige Eigenschaft von Codierungen erkennen: Es kann klar unterschieden werden zwischen dem Codieralgorithmus (Verschiebe das Alphabet nach links) und dem Codierschlüssel (Verschiebe um drei Buchstaben). Der Codieralgorithmus ist wesentlich schwieriger geheimzuhalten als etwa der Schlüssel. Bei den meisten gängigen Codierverfahren sind die zugehörigen Algorithmen öffentlich bekannt. Damit beruht also die Sicherheit des ganzen Systems auf der Geheimhaltung des Schlüssels.

Nachdem wir erkannt haben, daß das Verschlüsseln nach den oben beschriebenen Verfahren viel zu unsicher ist, gehen wir zum nächst komplizierteren über. Das Verfahren von Vignère benutzt mehrere Verschiebe-Codierungen gleichzeitig: Dazu werden die 26 möglichen Codierungen (abcd..., bcde..., cdef..., usw.) untereinander geschrieben. Das dabei entstehende Quadrat heißt Vignère-Quadrat. Nun denken wir uns ein Schlüsselwort aus und schreiben es über unseren zu verschlüsselnden Text, so daß immer ein Buchstabe unseres Schlüsselwortes über einem Buchstaben des Textes steht. Ist das Schlüsselwort zu Ende, so beginnen wir es wieder von vorne. Nehmen wir als Beispiel das Schlüsselwort AMIGA und als zu verschlüsselnden Text MAGAZIN. Die Vignère-Verschlüsselung funktioniert nun wie folgt: Der Buchstabe M erhält die Verschlüsselung, die unter dem Buchstaben M in dem Alphabet steht, das mit A beginnt, also den Buchstaben M.

Der nächste Buchstabe A erhält den Buchstaben, der in dem Alphabet, das mit M beginnt, unter dem A steht, also ebenfalls ein M. Schließlich ist am Ende das Wort MAGAZIN als MMGIZIZ verschlüsselt. Dem Entschlüsseln stellen sich hier schon etwas größere Probleme, da jetzt durchaus verschiedene Buchstaben des Ursprungstextes mit demselben Buchstaben verschlüsselt sein können, damit läuft aber unsere Analyse der Buchstabenhäufigkeiten ins Leere. Jedoch ist es immer noch möglich diese Texte zu entschlüsseln, falls man eine genügend lange Probe des verschlüsselten Textes zu Verfügung hat.

Hat man die Länge N des Schlüsselwortes ermittelt, so liegen nur noch N monoalphabetische Chiffrierungen vor, die wieder unserer Häufigkeitsanalyse zugänglich sind. Da die Schwachstelle dieser Verschlüsselung also die Anzahl der Buchstaben im Schlüsselwort ist, kann man versuchen diese zu umgehen, indem man ein unendlich langes Schlüsselwort wählt. Ein solches Schlüsselwort entsteht, wenn man etwa die einzelnen Buchstaben des Wortes zufällig wählt. Ein derartiges Schlüsselwort heißt entsprechend auch »Buchstabenwurm«. Ein derart verschlüsselter Text ist auch »theoretisch sicher«. Es gibt keine bekannte Methode, den Text zu entschlüsseln, ohne im Besitz des Buchstabenwurmes zu sein.

Der Buchstabenwurm hat aber auch Nachteile: Da der Schlüssel nun die gleiche Länge wie der Text selbst hat, und er dem Empfänger der verschlüsselten Nachricht ja ebenfalls übersandt werden muß (sonst kann dieser die Nachricht nicht entschlüsseln), verlagert sich das Risiko nur etwas, bleibt aber von der Größenordnung her gleich. Daher hat man in letzter Zeit damit begonnen, sich neue Verschlüsselungs-Algorithmen zu überlegen. Einer der bekanntesten hiervon ist der RSA-Algorithmus. Der Vorteil des RSA-Algorithmus ist, daß das Schlüsselwort, das zum Verschlüsseln benutzt wurde, öffentlich bekannt ist; daher kann es dem Empfänger auch gefahrlos übermittelt werden. Der Trick an der Sache ist nur, daß mit Hilfe dieses Schlüssel-

Buchstaben	Häufigkeit in %
a	6.51
b	1.89
c	3.06
d	5.08
e	17.40
f	1.66
g	3.01
h	4.76
i	7.55
j	0.27
k	1.21
l	3.44
m	2.53
n	9.78
o	2.51
p	0.79
q	0.02
r	7.00
s	7.27
t	6.15
u	4.35
v	0.67
w	1.89
x	0.03
y	0.04
z	1.13

Tabelle 2. Die Buchstabenhäufigkeiten der deutschen Sprache

wortes der Text nicht wieder entschlüsselt werden kann, sondern dazu wird ein anderes Schlüsselwort benötigt, das nur dem Empfänger der Nachricht bekannt ist, nicht einmal unbedingt dem, der die Nachricht verschlüsselt hat. Der einzige Grund, weshalb dieses Verfahren noch nicht in breitem Maße angewendet wird, ist, daß die momentan verfügbare Computer-Hardware noch zu langsam ist, von Software-Lösungen ganz zu schweigen — aber das wird sich vermutlich in naher Zukunft ändern. Zumindest gibt es einige Firmen, die sich mit der Entwicklung derartiger Verschlüsselungs-Chips beschäftigen.

Da der RSA-Algorithmus noch zu wenig praktikabel ist, greifen wir wieder auf die Methode mit dem Buchstabenwurm zurück. Allerdings wählen wir nicht eine wirklich zufällig gewonnene Buchstabenfolge, sondern eine mit Hilfe eines Zufallszahlengenerators erstellte. Dies macht zwar das Verschlüsselungssystem wieder unsicherer, aber wir haben ja schließlich keine Bankgeheimnisse zu hüten. Unser Basic-Unterprogramm (Listing 12) liest Daten aus dem File mit der Kennummer »inID« und schreibt die verschlüsselten Daten in das File mit der Kennummer »outID«. Das Schlüsselwort kann beliebig lang sein, jedoch werden nur die ersten 16 Buchstaben benutzt, ist das Wort kürzer, so wird es wiederholt, bis 16 Buchstaben erreicht sind. Es werden jeweils ganze Zufallszahlen im Bereich von

$0 \text{ bis } 2^{16}-1$

erzeugt. Auf diese Zahlen wird dann zusammen mit dem zu verschlüsselnden Text die XOR-Funktion angewendet. Dies hat den Vorteil, daß bei dieser Funktion das Verschlüsseln und das Entschlüsseln genau gleich sind.

Wendet man also die Verschlüsselungsfunktion (mit dem richtigen Paßwort und dem richtigen Zufallszahlengenerator) auf den verschlüsselten Text an, so entsteht wieder der ursprüngliche Text.

Der Zufallszahlen-Generator kann ebenfalls an das Unterprogramm übergeben werden. In unserem Listing wurde ein eigener Zufallszahlengenerator implementiert, da der eingebaute Generator von Amiga-Basic anscheinend einige Schwierigkeiten mit der Initialisierung hat. Zumindest traten nach dem Initialisieren mit RANDOMIZE mit den gleichen Zahlen nicht immer die gleichen Zufallsfolgen auf. Das hat natürlich zur Folge, daß ein einmal verschlüsselter Text nicht wieder entschlüsselt werden kann. Unser in Maschinensprache implementierter Zufallsgenerator erwartet als Argument die Adresse einer kurzen Ganzzahlvariable. Diese Variable wird mit der Zahl 65549 multipliziert und das niederwertige Wort des Ergebnisses wird in die übergebene Variable zurückgeschrieben.

Zum Abschluß nun wie »angedroht« einige kurze Worte zur Mathematik auf dem Amiga. Denn leider muß gesagt werden, daß Mathematik auf dem Papier und Mathematik auf dem Computer nicht immer das gleiche sind. Dies liegt an der internen Zahlendarstellung der Computer. So könnte man zum Beispiel auf den ersten Blick annehmen, daß das folgende Programmfragment korrekt ist:

```
i = 1
x = 1
dx = 0.1
WHILE x <> 0
  PRINT i
  i = i + 1
  x = x - dx
WEND
```

Und tatsächlich, wenn Sie dieses Programm einem »Schreibtischtest« unterziehen, so werden die Zahlen 1 bis 10 ausgedruckt und dann bricht das Programm ab. Starten Sie das Programm dagegen auf dem Amiga, so werden Sie feststellen, daß Sie eine Endlos-Schleife programmiert haben. Der Grund dafür ist, daß in unserem Dezimalsystem die Zahl 0.1 ein schöner, glatter Bruch ist. In der Computer-internen Binärdarstellung ist diese Zahl jedoch ein unendlicher Dezimalbruch. Da der Computer aber nur mit endlich vielen Stellen rechnen kann, macht er automatisch Rundungsfehler. Diese führen dazu, daß x in unserem Beispiel den Wert 0 niemals annimmt. Die endliche Rechengenauigkeit von Computern hat noch andere Auswirkungen. So kann das Ergebnis von mehreren Additionen durchaus davon abhängen, in welcher Reihenfolge man die Zahlen aufsummiert. Hat man viele Gleitpunktzahlen aufzusummieren, sollte man immer mit den kleinsten beginnen und sich dann zu den größeren Zahlen hin vorarbeiten. Dabei sollten Sie auch beachten, daß Additionen auf dem Computer eine größere Rechengenauigkeit besitzen als Subtraktionen. Dies hat seine Ursache darin, daß sich bei der Subtraktion von beinahe gleich großen Zahlen die führenden Dezimalstellen gegenseitig aufheben können. Die so erreichten Rechenfehler können einige Prozent betragen, in Extremfällen erhalten Sie sogar vollkommen unsinnige Ergebnisse.

Mit diesen eher theoretischen Betrachtungen wollen wir unseren Kurs über fortgeschrittene Basic-Programmiertechniken beenden. Selbstverständlich konnten wir nur eine sehr kleine, eng begrenzte Auswahl an Algorithmen vorstellen, doch hoffen wir, Ihr Interesse an der theoretischen Seite der Programmierung geweckt zu haben. Ohne die Kenntnis von gängigen Datenstrukturen und Algorithmen ist die Erstellung von effizienten und leistungsfähigen Programmen nicht möglich.

Wenn Sie das Prinzip der modularen Programmierung, das heißt, der Aufteilung Ihrer Programme in ein neues Hauptprogramm und zahlreiche bestehende Unterroutinen zudem berücksichtigen, sollte dem Erfolg nichts mehr im Weg stehen.

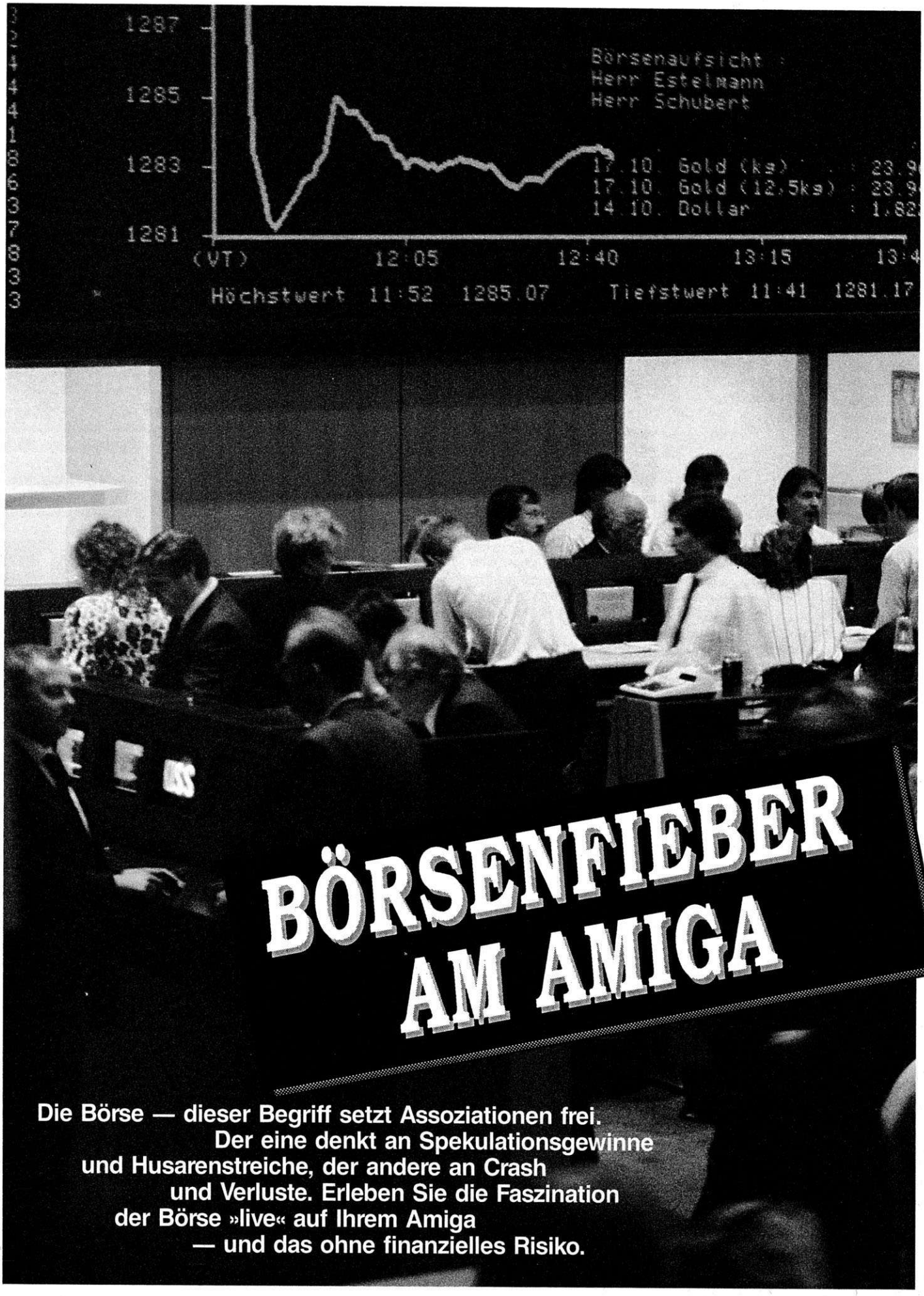
(Jürgen Singer/rs)

```
main:
  LoadRandom Generator$
  inID = 1
  outID = 2
  INPUT "Geben Sie einen Filenamen an: ",FileName$
  OPEN FileName$ FOR INPUT AS inID
  OPEN "ram:Crypt.dat" FOR OUTPUT AS outID
  Encrypt inID, outID, "Amiga", Generator$
  OPEN "ram:Crypt.dat" FOR INPUT AS inID
  OPEN "ram:Decrypt.dat" FOR OUTPUT AS outID
  Encrypt inID, outID, "Amiga", Generator$
END

SUB Encrypt( inID , outID , Password$ , Ranf$ ) STATIC
  DIM pw%(8)
  IF Password$ = "" THEN EXIT SUB
  WHILE LEN(Password$) < 16
    Password$ = Password$ + Password$
  WEND
  FOR i = 1 TO 8
    pw%(i) = CVI(MID$(Password$,2*i,2))
  NEXT i
  n = LOF(inID)
  WHILE n > 1
    i = 1
    WHILE n > 1 AND i <= 8
      x% = CVI(INPUT$(2,inID))
      Random& = SADD(Ranf$)
      CALL Random&( VARPTR(pw%(i)) )
      PRINT #outID,MKI$(x% XOR pw%(i));
      n = n - 2
      i = i + 1
    WEND
  WEND
  IF n = 1 THEN
    x% = ASC(INPUT$(1,inID))
    PRINT #outID,CHR$(x%);
  END IF
  CLOSE inID,outID
  ERASE pw%
END SUB

SUB LoadRandom( Ranf$ ) STATIC
  Ranf$ =          MKI$(&H48E7) + MKI$(&HC080) ' MOVEM.L ..
                                               ' .. A0/D0-D1,-(SP)
  Ranf$ = Ranf$ + MKI$(&H206F) + MKI$(&H10)    ' MOVEA   16(SP),A0
  Ranf$ = Ranf$ + MKI$(&H3010)                 ' MOVE.W  (A0),D0
  Ranf$ = Ranf$ + MKI$(&HB381)                 ' EOR     D1,D1
  Ranf$ = Ranf$ + MKI$(&H3200)                 ' MOVE.W  D0,D1
  Ranf$ = Ranf$ + MKI$(&H4841)                 ' SWAP    D1
  Ranf$ = Ranf$ + MKI$(&HC1FC) + MKI$(&HD)     ' MULU    #000D,D0
  Ranf$ = Ranf$ + MKI$(&HD280)                 ' ADD.L   D0,D1
  Ranf$ = Ranf$ + MKI$(&H3081)                 ' MOVE.W  D1,(A0)
  Ranf$ = Ranf$ + MKI$(&H4CDF) + MKI$(&H103)   ' MOVEM.L ..
                                               ' ..(SP)+,A0/D0-D1
  Ranf$ = Ranf$ + MKI$(&H4E75)                 ' RTS
END SUB
```

Listing 12. Die Verschlüsselungs-Routine »Encrypt« mit dem Maschinenspracheteil »LoadRandom«



BÖRSENFIEBER AM AMIGA

Die Börse — dieser Begriff setzt Assoziationen frei. Der eine denkt an Spekulationsgewinne und Husarenstreiche, der andere an Crash und Verluste. Erleben Sie die Faszination der Börse »live« auf Ihrem Amiga — und das ohne finanzielles Risiko.

Kaufen und Verkaufen, durch geschicktes Manipulieren den Gegner übernahmereif machen. Dieses Thema stammt nicht aus dem Wallstreet-Journal, sondern von einer turbulenten Börsensimulation, die so manchen eingefleischten Broker ins Schwitzen bringen kann. »Broker« (Listing 1) ist ein Spiel für zwei bis vier Teilnehmer. Durch geschickte Aktionen versucht jeder, den Kurs der eigenen Aktien möglichst hoch zu treiben, und den der Mitspieler so tief wie möglich zu drücken.

Wer zuerst eine vorher vereinbarte Summe erreicht, ist Sieger und kann sich beinahe schon als Börsen-Profi fühlen. Sie benötigen allerdings sehr viel planerisches Geschick und ein wenig Glück, um gegen die Konkurrenz zu bestehen.

Alle wichtigen Kommandos werden mit der rechten Maustaste über Pull-Down-Menüs oder wahlweise über Tastenkombinationen (Shortcuts) aufgerufen, die man sich mit dem Menüpunkt **Help** (Bild 1) ansehen kann. Um ein Spiel zu beginnen, wählen Sie bitte **Neues Spiel** an. Nach Eingabe aller Spielernamen werden Sie aufgefordert, das Startkapital und den Betrag festzusetzen, bei dessen Erreichen das Spiel beendet wird. Der Mindestbetrag, der vom Sieger zu erreichen ist, muß mindestens zehnmal so groß sein wie das Startkapital.

Broker's Basar

In jeder neuen Runde haben Sie fünf Handlungspunkte zur Verfügung, die Sie gegen Aktionen an der Börse tauschen können. Je nach Art der Aktion werden unterschiedliche Punktzahlen abgezogen. So kostet jeder Aktienkauf und -verkauf jeweils zwei Punkte, Manipulationen der Kurse (siehe unten) zwischen einem und drei Punkten. Daneben können Sie verschiedene Dienstleistungen in Anspruch nehmen, wobei die meisten bares Geld kosten. So können Sie sich für 60 Mark die Kursentwicklung ausgewählter Aktien ansehen oder für 30 Mark einen Vergleich des Kapitals aller Spieler ausgeben lassen. 40 Mark müssen Sie investieren, wenn Sie Informationen über die Aktien Ihrer Gegner wollen.

Sie sollten dabei bedenken: Jede Information schmälert zwar Ihr Bankkonto, aber eine Fehlinvestition kann weit wertvollere Handlungspunkte kosten. Sie können mit dem Menüpunkt »Börsenaktion« Aktienkurse teilweise gezielt beeinflussen. Voraussetzung hierfür ist, daß Sie die Besitzverhältnisse aller Spieler genau kennen: Wie viele Brauereiaktien hatte doch gleich die Konkurrenz, waren Versicherungsaktien nicht eben billiger geworden?

Wenn Sie keine Handlungspunkte mehr haben, können Sie nur noch Informationen einholen oder an den nächsten Spieler abgeben (mit dem Menüpunkt »Nächster«). Bei jedem Wechsel werden die Schulden, die sich im Verlauf der aktuellen Runde angesammelt haben, mit dem Barkapital verrechnet.

Das Spiel ist zu Ende, wenn das Nettokapital eines Spielers (Bargeld plus in Aktien angelegtes Kapital abzüglich Schulden) unter 100 Mark fällt oder ein Broker den festgelegten Spielendbetrag erreicht. Sind dabei noch Schulden vorhanden, so werden erst diese verrechnet. Sind die Schulden eines Spielers größer als die Hälfte des Startkapitals, wird das Spiel ebenfalls beendet.

Spekulieren Sie mit

Kapital erwirtschaftet der Broker, indem er Aktien zu einem möglichst billigen Kurs erwirbt (Bild 2) und diese zu einem hohen Preis wieder verkauft. Bei Kauf und Verkauf werden jeweils 5 Prozent des Umsatzes als Maklergebühren vom Bargeld abgezogen und zwei Handlungspunkte verbraucht. Damit der Spieler nicht den Überblick verliert, zeigt der Computer beim Verkauf der Aktien den Ankaufskurs und den entsprechenden Gewinn oder Verlust an.

Beim Menüpunkt Börsenaktion geht es richtig los. Hier spielen sich alle Kursveränderungen ab. Allerdings haben Sie nicht unbegrenzt Einfluß auf die Kurse. Beim Aufruf einer Börsenaktion bestimmt der Zufall, welche Manipulationen ermöglicht werden. Die ganze Handlungspalette zeigt sich bei »Manual Activity II« (Bild 3). Hier können Sie wählen, welche Manipulation ausgeführt werden soll. Dazu bekommen Sie noch eine kurze Erklärung zur Wirkung der einzelnen Punkte.

BROKER - Das Börsenspiel von Sebastian Peetz

Menüpunkt	Taste	Erklärung	DM	HP
Laden	ALT l	Ein Spiel wird von Disk geladen		
Speichern	ALT s	Ein Spiel wird abgespeichert		
Neues Spiel	ALT n	Es wird ein neues Spiel begonnen		
Quit	ALT q	Spielende ohne die Spielstände abzuspeichern		
Börsenaktion	a	Der Spieler kann aktiv die Börsenkurse durch geschicktes Handeln beeinflussen	20	1-3
Börsenkurse	b	Es werden die aktuellen Börsenkurse angezeigt	-	-
Charts	c	Aktienkurse können graphisch dargestellt werden	60	-
Aktienkauf	k	Aktien können zum Tageskurs gekauft werden	5%	2
Aktienverk.	v	Aktien können zum Tageskurs verkauft werden	5%	2
Kapitalliste	l	Eigenes Kapital (Aktienkapital und bar)	-	-
Kapitalvergl.	p	Kapital aller Spieler	30	-
Aktienliste	i	Auflistung der eigenen Aktien (Kurse,Stück)	-	-
Aktienvergl.	g	Aktien aller Spieler (Stück)	40	-
Nächster	n	Der nächste Spieler ist an der Reihe		
Uebersicht	u	Alle wichtigen Daten zum Spiel im überblick		
Info	Help	Diese Seite		

Bitte Taste drücken

Bild 1. Diese Übersicht erreichen Sie mit der »Help«-Taste oder mit Help aus dem Info-Menü

Aktienkauf

Name: Boesky Barkapital: 3704 DM

Welche Aktie möchten Sie kaufen? (0) Keine

Nr.	AG	Vorrätig	Kurs
(1)	Bauwerte	1732	167
(2)	Textil	1492	140
(3)	Brauerei	1488	278
(4)	Kosmetik	1250	282
(5)	Elektrizität	1250	428
(6)	Chemie	1000	428
(7)	Pharmazie	1000	604
(8)	Maschinenbau	750	574
(9)	Kaufhaus	750	728
(10)	Computer	500	760
(11)	Automobil	500	878
(12)	Banken	250	878
(13)	Versicherung	250	700

Bild 2. Das Schaufenster für den Aktieneinkauf

Broker

Broker sind Leute, deren Job es ist, an der Börse mit Aktien und Wertpapieren zu handeln. Dabei werden Kursdifferenzen ausgenutzt, um Gewinne zu erzielen. Mit verschiedenen Techniken ist es möglich, Kurse zu beeinflussen. So steigt der Kurs von Aktien häufig auf das Gerücht hin, die Firma würde von einer anderen übernommen. Solche Tendenzen geschickt auszunutzen, ist Aufgabe eines guten Brokers.

```
Börsenaktion

Brokers Manual Activity II
--------------------------

(1)  Dividende / Aufwertung
(2)  Hausse - Baisse !!  12¡
(3)  Hausse - Baisse !¡  12¡¡
(4)  Hausse - Baisse !!! 12¡
(5)  Hausse - Baisse !¡¡ 12¡¡
(6)  Multiple Action Flood
(7)  Hausse - Baisse 12!  1¡
(8)  Hausse - Baisse 12!  1¡¡
(9)  Hausse - Baisse 12!! 1¡
(10) Hausse - Baisse 12!! 1¡¡
(11) Manual Activity I
(12) Finanzamt

Börsenaktion-Nummer? |


Erklärung:  != normal steigend
           !!= stark  steigend
            ¡= normal fallend
           ¡¡= stark  fallend
```

Bild 3. »Manual Activity II« zeigt Ihnen die ganze Trickkiste der Manipulation

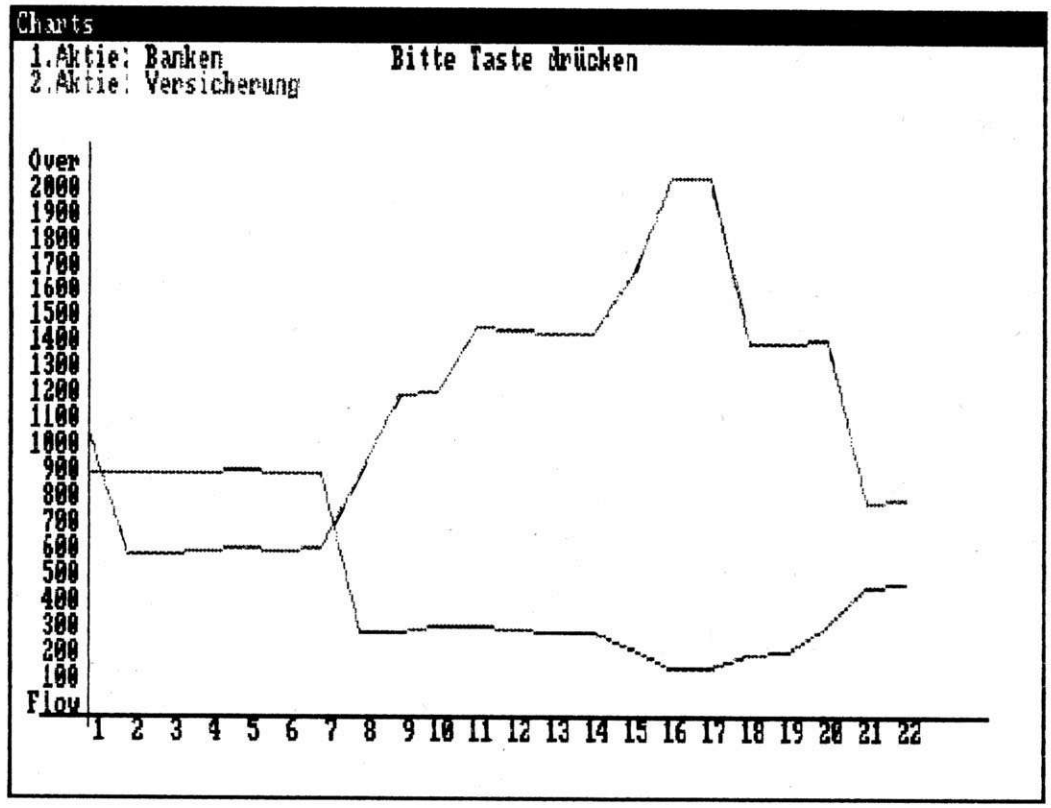


Bild 4. Ausgewählte Aktien können Sie als Grafik ausgeben lassen

So gibt es verschiedene Aktionen, die eine ausgewählte Aktie steigen oder fallen lassen und alle anderen umgekehrt beeinflussen. Wenn Sie Ihren Mitspielern das Leben schwermachen wollen, ist »Manual Activity I« für Sie das Richtige: Hier können Sie selbst eine Aktie bestimmen, die steigen und eine, die sinken soll. Außerdem sind die Veränderungen hier weit größer als bei den zufallsgesteuerten Varianten.

Ebenfalls sehr große Veränderungen erzielen Sie mit »Multiple Action Flood«. Hier wird jede einzelne Aktie entweder stark auf- oder abgewertet. Fast schon harmlos ist dagegen der gelegentlich auftauchende Fiskus, das Finanzamt zeigt sich ungewohnt milde: Steuern werden — im Gegensatz zur Realität — nur auf das Barvermögen erhoben.

Die Notbremse

Jede der zufallsgesteuerten Aktionen kostet Sie einen Handlungspunkt. Sie können Kursveränderungen allerdings auch noch aufhalten, beispielsweise, wenn Ihre Aktien durch eine zufällige Manipulation geschwächt oder die Ihrer Gegner gestärkt würden. In diesem Fall werden Ihnen statt einem gleich drei Handlungspunkte abgezogen. Sie sollten sich also genau überlegen, ob Sie eine Aktion wirklich rückgängig machen wollen. Für die wertvolleren Aktionen, bei denen Sie direkten Einfluß auf die ausgewählten Aktien haben oder die hohe Kursunterschiede mit sich bringen, werden Ihnen immer drei Punkte abgezogen.

Kapitalvergleich

Name	Bargeld	Aktienwert	Brutto	Schulden	Netto
Ivan	861	6364	7225	0	7225
Boesky	191	17303	17494	130	17364

Bitte Taste drücken

Bild 5. Kapitalvergleich gibt Ihnen Informationen über Ihre Mitspieler

Neben diesen Menüpunkten, die aktive Manipulationen zulassen, gibt es die Möglichkeit, Informationen einzuholen. Der Menüpunkt **Börsenkurse** öffnet ein Fenster, in dem die Kurse aller Aktiengesellschaften gezeigt werden. **Ekurs** bedeutet Einstandskurs und dient zur Bewertung des derzeitigen Aktienstands. Wer eine grafische Auswertung der Kursentwicklung sehen möchte, kommt bei **Charts** (Bild 4) auf seine Kosten. Es werden die letzten 22 Kursbewegungen von maximal zwei Aktien gleichzeitig grafisch dargestellt. Hier sollten Sie aber genau wissen, was Sie wollen, denn jede Grafik kostet 60 Mark.

Daneben können Sie sich über die jeweiligen Besitzverhältnisse informieren. **Kapitalliste** zeigt das aktuelle Kapital des Spielers, der am Zug ist, in Barwert und Aktienwert. **Aktienliste** dagegen zeigt Ihren gesamten Aktienbesitz. Um sich Informationen über andere Spieler zu besorgen, muß man Gebühren bezahlen. **Kapitalvergleich** gibt das Kapital aller Spieler aus (Bild 5), **Aktienvergleich** den Besitzstand in Aktien.

Informationen über Ihren eigenen Besitz erhalten Sie kostenlos. Wünschen Sie dagegen Informationen über die anderen Spieler, so müssen Sie 30 Mark (Kapitalvergleich) beziehungsweise 40 Mark (Aktienvergleich) für Gebühren ausgeben.

Mit dem Menüpunkt **Sichern** können Sie ein Spiel unterbrechen und den Spielstand auf Diskette sichern. Beim erneuten Start wählen Sie statt **Neues Spiel** einfach **Laden** und spielen dort weiter, wo Sie aufgehört haben.

Sie können hierfür zwei Verfahren anwenden: Wenn Sie für den gespeicherten Spielstand das Directory »Brokerdata« verwenden wollen, brauchen Sie im Programm nichts zu verändern. Sie geben einfach im CLI ein »makedir Brokerdata«. Wollen Sie dagegen ein eigenes Directory verwenden, dann müssen Sie den Pfadnamen bei den gekennzeichneten Stellen im Listing auf die richtige Diskette und Schublade umstellen (Zeile 76, 78, 87 sowie 97, 99 und 108).

Mit dem Menüpunkt **Quit** verlassen Sie Broker, ohne den Spielstand zu sichern.

Tippen Sie das Basic-Programm »Broker« ab und lassen Sie sich faszinieren von der Welt der schnellen Gewinne und Verluste, der Spekulation und der Übernahmen.

Erleben Sie das Auf und Ab der Börse, Hausse und Baisse, Erfolge und Crashs — und das alles, ohne Ihr Bankkonto zu belasten.

(Christian Buchner/so)

Eingabehinweise

Tippen Sie Broker mit dem Checksummer ab. Beachten Sie dabei die Anleitung auf Seite 159. Starten Sie jetzt Amiga-Basic. Aktivieren Sie das Ausgabefenster und geben Sie

```
CLEAR ,40000,5000
```

ein. Laden Sie nun mit »Open« Ihr Programm und starten Sie es mit »Start«.

Programmname: Broker

Computer: A500, A1000, A2000 mit Kickstart 1.2

Sprache: Amiga-Basic

```
 1 xO0 ' ************************************************
 2 9S  ' *   BROKER - Das Börsenspiel von Sebastian Peetz   *
 3 up  ' *            (C) 1988 Markt & Technik             *
 4 OR  ' ************************************************

 5 GP  RANDOMIZE TIMER
 6 i7  OPTION BASE 1
 7 ZD  'Variablen und Screenaufbau
 8 6r  DIM sp$(4),Ak$(13),Ak(4),kap(4),kre(4),ka(4,13),x(13),z(13)
       ,v(13)
 9 tw  DIM st$(13),note$(13),w(4,13),gr(2)
10 Sl  p=0:me2=0:Q=1:req=0:endes=0:no=0
11 zn  SCREEN 2,640,255,4,2
12 Sh  WINDOW 2,"BROKER",(0,0)-(630,240),0,2
13 i0  WINDOW OUTPUT 2
14 XU  ON ERROR GOTO Fehler
15 fH  FOR i=0 TO 15
16 5m1  READ a,b,c
17 3M   PALETTE i,a/15,b/15,c/15
18 jz0 NEXT i
19 eK  DATA ,,,15,15,15,15,15,,15,,,,15,,,,15,15,,15,8,8,8,,,15
20 q1  DATA 15,15,15,15,15,,10,5,3,,15,,,,15,15,,15,,15,15
21 Fd  LOCATE 15,25:PRINT "Bitte warten Sie einen Moment"
22 Tv  'Aktiendaten lesen
23 on  RESTORE papiere
24 OF  FOR i=1 TO 13:READ Ak$(i):NEXT i
25 6C  'Menüleiste 1
26 PL  m1:
27 zt1  RESTORE menua
28 xw   READ titel:FOR i=1 TO titel
29 YR   READ unter:FOR j=0 TO unter
30 uf2   READ statu:IF statu>2 THEN statu=1
31 z3    IF j=0 AND statu=2 THEN statu=1
32 Fs    READ a$
33 J6    MENU i,j,statu,a$
34 2T1  NEXT j:NEXT i
35 7D0 CLS
36 SP1  ON MENU GOSUB split
37 yl   GOTO schl
38 NU0 'Menüleiste 2
39 fc  m2:
40 GB1  RESTORE menub
41 A9   READ titel:FOR i=1 TO titel
42 le   READ unter:FOR j=0 TO unter
43 7s2   READ statu:IF statu>2 THEN statu=1
44 CG    IF j=0 AND statu=2 THEN statu=1
45 S5    READ a$
46 WJ    MENU i,j,statu,a$
47 8A1  NEXT j:NEXT i:me2=1
48 KQ0 CLS
49 oK  'Mausabfrage zu Menleiste 2
50 gd1  ON MENU GOSUB split
51 zd   IF p THEN GOTO Kurse
52 D0   GOTO schl
53 tQ0 'Hauptprogramm
54 In  c:
55 RX1  CLS
56 Zn   IF hp>0 THEN GOSUB Hpu ELSE IF p THEN GOSUB NoHpu
57 8G0 schl:
58 Aa1  MENU ON
59 wM   a$=INKEY$:IF a$="" GOTO schl
60 HJ0 split:
61 XR1  nu=MENU(1):MENU OFF
62 9l   ON MENU(0) GOTO Broker,Boerse,Aktien,Kapakt,Info
63 oS0 Broker:
64 xp1  ON nu GOTO lad,sav,neue,Ende
65 Yz0 Boerse:
66 bi1  ON nu GOTO Boersenak,Kurse,Charts
67 X20 Aktien:
68 Ke1  ON nu GOTO Aktkauf,Aktverk
69 HI0 Kapakt:
70 V21  ON nu GOTO Kaplist,Kapver,Aktlist,Aktver
71 eI0 Info:
72 Bw1  ON nu GOTO wech,Uebers,Help
73 8m0 'Menüpunkt Laden
74 H41  lad:
75 g02   WINDOW 4,"Daten laden",(0,10)-(300,180),0,2:COLOR 13,7:CL
         S:LOCATE 2,1
76 bd    FILES "BrokerData"                    'Directorypfad ge
         gebenenfalls ändern!
77 6L    PRINT:LINE INPUT "Dateiname: ";datl$
78 fb    OPEN "Brokerdata/"+datl$ FOR INPUT AS #1  'Dirpfad gegeb
         enenfalls ändern
79 KA    INPUT #1,s,kapital,spielende,p,hp
80 8k    FOR i=1 TO s:INPUT #1,sp$(i):INPUT #1,kap(i):INPUT #1,
         kre(i):NEXT i
81 eF    FOR i=1 TO 13
82 oL3    INPUT #1,x(i):INPUT #1,v(i):INPUT #1,z(i):INPUT #1,s
          t$(i):INPUT #1,note$(i)
83 m22   NEXT i
84 13    FOR j=1 TO s
85 c93    FOR i=1 TO 13:INPUT #1,w(j,i):INPUT #1,ka(j,i):NEXT i
86 r82   NEXT j
87 dQ    CLOSE #1
88 Ec    FOR i=1 TO 13:we=VAL(MID$(st$(i),1,4))
89 P73    IF (we<1000 AND we>99)THEN st$(i)=" "+st$(i)
90 EF     IF we<99 THEN st$(i)="  "+st$(i)
91 2G2   NEXT i:IF me2 THEN GOTO k
92 rB    WINDOW CLOSE 4:WINDOW OUTPUT 2
93 nK    LOCATE 15,25:PRINT "Bitte warten Sie einen Moment":GOTO m
         2
94 z80 'Menüpunkt Speichern
95 OC1  sav:
96 G32   WINDOW 4,"Daten abspeichern",(0,10)-(300,180),0,2:COLOR 1
         3,7:CLS:LOCATE 2,1
97 2Z    FILES "Brokerdata"                    'Dirpfad gegebene
         nfalls ändern!
98 qU    PRINT:LINE INPUT"Dateiname: ";dats$
99 vH    OPEN "BrokerData/"+dats$ FOR OUTPUT AS #1  'siehe oben
100 4r   PRINT #1,s,kapital,spielende,p,hp
101 Hr   FOR i=1 TO s:PRINT #1,sp$(i):PRINT #1,kap(i):PRINT #1,
         kre(i):NEXT i
102 za   FOR i=1 TO 13
103 Dr3   PRINT #1,x(i):PRINT #1,v(i):PRINT #1,z(i):PRINT #1,s
          t$(i):PRINT #1,note$(i)
104 7N2  NEXT i
105 6O   FOR j=1 TO s
106 NF3   FOR i=1 TO 13:PRINT #1,w(j,i):PRINT #1,ka(j,i):NEXT i
107 CT2  NEXT j
108 G1   CLOSE #1:KILL "BrokerData/"+dats$+".info" 'siehe oben
109 VM   WINDOW CLOSE 4:WINDOW OUTPUT 2:GOTO Quit
110 KA0 'Menüpunkt Neues Spiel
111 D01 neue:
112 Ah2  WINDOW 4,"Neues Spiel",(0,10)-(280,140),0,2:COLOR 15,5:CL
         S
113 ip1 wie:
114 en2  LOCATE 2,1:INPUT "Wieviele Mitspieler";s:IF (s<2 OR s>4
         ) THEN wie
115 ib   PRINT:FOR i=1 TO s
116 y81 spe:
117 3Q2  COLOR 1,5
118 A2   LOCATE (2*i+2),1
119 eX   PRINT"Name des"i".Spielers:"
120 yf   COLOR  2,5
121 E8   LINE INPUT sp$(i):l=LEN(sp$(i)):IF (l<1 OR l>10)THEN sp
         e
122 Pf1 NEXT i
123 7A  kapi:
124 UP2  COLOR 15,5:LOCATE (2*i+3),1:INPUT "Startkapital (500-1200
         0)";kapital
125 DJ   IF (kapital<500 OR kapital>12000) THEN kapi
126 FY1 spien:
127 4h2  LOCATE (2*i+4),1:INPUT "Spielende (>=Kapital*10)";spiele
128 Ja   nde IF (spielende<kapital*10 OR spielende>9999999&)THEN spi
         en
129 qI   COLOR 2,5:CLS:LOCATE 2,1
130 Pl   FOR i=1 TO s
131 Us4    PRINT " Spieler";i":  ";sp$(i):PRINT
132 V52  NEXT i:COLOR 5,2
133 gH   PRINT:LOCATE ,2:PRINT"Startkapital :";kapital;" DM"
134 qs   LOCATE ,2:PRINT"Spielende bei:";spielende;"DM"
135 1J   COLOR 15,5:PRINT:PRINT:INPUT" Alles in Ordung";z$:IF (z$=
         "n" OR z$="N") GOTO Neu
136 Zt   WINDOW CLOSE 4:WINDOW OUTPUT 2
137 hY1 anf:
138 QS2  CLS:LOCATE 15,22:PRINT"Bitte warten - AMIGA generiert Dat
         en"
139 vL   FOR i=1 TO 13:st$(i)="":NEXT i
140 bh   FOR ip=1 TO 13:x(ip)=INT(ip/2+.5)*150
141 Lj3   v(ip)=(ABS((ip<2)+(ip<4)+(ip<6)+(ip<8)+(ip<10)+(ip
          <12))+1)*250
142 Nr    FOR io=1 TO 23
```

```
143 zN4     h$="0000"+STR$(x(ip))
144 SD      st$(ip)=st$(ip)+MID$(h$,LEN(h$)-3,4)
145 b83    NEXT io
146 gZ     note$(ip)="BBR":z(ip)=INT(ip/2+.5)*150
147 gE2   NEXT ip
148 TO    FOR i=1 TO s:kap(i)=kapital:kre(i)=0
149 7D3    FOR ip=1 TO 13:w(i,ip)=0:ka(i,ip)=0:NEXT ip
150 r72   NEXT i
151 1I    p=1:hp=5:IF me2 THEN CLS:GOTO Kurse
152 UO    GOTO m2
153 IC0 'Menüpunkt Quit
154 vJ1  Ende:
155 h52   MENU RESET
156 iO    WINDOW CLOSE 2
157 OU    SCREEN CLOSE 2
158 ID    END
159 wk0 'Menüpunkt Börsenaktion
160 MD  Boersenak:
161 QJ1   WINDOW 4,"Börsenaktion",(0,10)-(330,235),0,2:CLS
162 mk    IF hp<3 THEN GOTO Notg
163 eD    kre(p)=kre(p)+20:r1=INT(RND*13)+1
164 GP    ON r1 GOTO Divi,HaBa,HaBa,HaBa,HaBa,Flood,HaBa,HaBa,HaBa,H
        aBa,Ma1,Ma2,Fiskus
165 0b0 'Dividende/Aufwertung
166 Ky1  Divi:
167 sJ2    COLOR 0,15:CLS:PRINT:PRINT" Dividende/Aufwertung"
168 MR     r1=INT(RND*15):IF r1=0 OR r1=14 THEN GOSUB Wahl:GOTO eing
169 Gw     GOTO wei
170 mn1   eing:
171 W82    LOCATE 2,1:INPUT" AG-Nummer Dividende/Aufwertung";r1
172 7P     IF r1<1 OR r1>13 THEN eing
173 yP     COLOR 0,15:CLS:PRINT:PRINT" Dividende/Aufwertung"
174 ls1   wei:
175 F82    IF x(r1)<z(r1)THEN GOTO wei3
176 wv     PRINT:PRINT" Dividende 4%":PRINT
177 xe     PRINT" ----------------------------------------"
178 kn     PRINT" AG              Diff.   Alt    Neu"
179 gD     PRINT:PRINT TAB(2)Ak$(r1)TAB(18)INT(x(r1)/25)TAB(26)x(r1)
        TAB(33)x(r1)-INT(x(r1)/25)
180 Oh     PRINT" ----------------------------------------"
181 cW     PRINT:PRINT" Spieler       Haben    *DM      Gewinn"
182 VS     FOR ip=1 TO s:LOCATE 13+ip,2
183 5V3     PRINT sp$(ip)TAB(15)w(ip,r1)TAB(25)INT(x(r1)/25)TAB(32)w
        (ip,r1)*INT(x(r1)/25)
184 ZK2    NEXT ip:PRINT
185 7T     LOCATE 19,1:INPUT " Akzeptieren Sie";jn$: IF jn$="n" OR j
        n$="N" THEN hp=hp-3:GOTO wei2
186 GF     FOR ip=1 TO s:kap(ip)=kap(ip)+w(ip,r1)*INT(x(r1)/25):NEXT
        ip
187 BC     x(r1)=x(r1)-INT(x(r1)/25):hp=hp-1
188 rA1   wei2:
189 mB2    GOTO kk
190 yI1   wei3:
191 942    PRINT:PRINT" Aufwertung (Kurs=EKurs)":PRINT
192 sm     PRINT" ----------------------------------"
193 Bt     PRINT" AG              Aufw.   Alt    Neu"
194 C3     PRINT:PRINT TAB(2)Ak$(r1)TAB(18)z(r1)-x(r1)TAB(26)x(r1)TA
        B(33)z(r1)
195 vp     PRINT" ----------------------------------"
196 ad     PRINT:PRINT" Spieler       Haben    *DM      Verlust"
197 kh     FOR ip=1 TO s:LOCATE 13+ip,2
198 wk3     PRINT sp$(ip)TAB(15)w(ip,r1)TAB(24)z(r1)-x(r1)TAB(32)w(i
        p,r1)*(z(r1)-x(r1))
199 hF      IF w(ip,r1)=0 THEN wei4
200 9D      IF kap(ip)>(w(ip,r1)*(z(r1)-x(r1)))THEN kap(ip)=kap(ip)
        -(w(ip,r1)*(z(r1)-x(r1))):GOTO wei4
201 GP      LOCATE 23,1:PRINT" Achtung, "+sp$(ip)+" !"
202 5n      PRINT" Verlust Ihrer "+Ak$(r1)+"-Aktien!"
203 Av      kap(ip)=kap(ip)+w(ip,r1)*x(r1):v(r1)=v(r1)+w(ip,r1):w(ip
        ,r1)=0
204 Hc1   wei4:
205 7t2    NEXT ip:x(r1)=z(r1):PRINT:PRINT" Automatische Durchführun
        g!"
206 Kx     hp=hp-3:GOSUB tast:GOTO kk
207 tj0 'Hausse/Baisse
208 kM  HaBa:
209 YM1   COLOR 5,2:CLS
210 YW    IF hp<3 THEN GOTO Notg
211 fN    IF r1=2 OR r1=7 THEN up=1:dw=1 ELSE IF r1=3 OR r1=8 THEN u
        p=1:dw=2
212 Jt    IF r1=4 OR r1=9 THEN up=2:dw=1 ELSE up=2:dw=2
213 Cl    r2=INT(RND*13):IF r2=0 OR r2=12 THEN GOSUB Wahl
214 Nf    IF r1>6 THEN go3
215 23    IF r2<>0 AND r2<>12 THEN go2
```

```
216 H7    go:
217 t02    LOCATE 2,1:INPUT " AG-Nummer Hausse";r2:IF r2<1 OR r2>1
        3 THEN go
218 t11   go2:
219 Wu2    CLS:PRINT:PRINT" Hausse:":PRINT" ----------------------
        ------------"
220 vL     PRINT" AG              Diff.   Alt    Neu":PRINT
221 bH     r1=INT(RND*5)+1
222 2C     PRINT TAB(2)Ak$(r2)TAB(18)r1*up TAB(26)x(r2)TAB(33)x(r2)+
        r1*up
223 NH     PRINT" ----------------------------------"
224 xz     PRINT:PRINT" Baisse:":PRINT" --------------------------
        --------"
225 OQ     PRINT" AG              Diff.   Alt    Neu":PRINT
226 dT     FOR ip=1 TO 13:IF ip=r2 THEN go2a
227 ZB3     LOCATE 12+ip,2
228 sT      PRINT Ak$(ip)TAB(18)-r1*dw TAB(26)x(ip)TAB(33)x(ip)-r1*d
        w
229 jo1   go2a:
230 LN2    NEXT ip:PRINT" ----------------------------------"
231 Np     LOCATE 27,2:INPUT"Akzeptieren Sie";jn$:IF jn$="n" OR jn$=
        "N" THEN hp=hp-3:GOTO kk
232 nH     FOR ip=1 TO 13
233 QW3     IF ip=r2 THEN x(ip)=x(ip)+r1*up ELSE x(ip)=x(ip)-r1*dw
234 Ya2    NEXT ip:hp=hp-1:GOTO kk
235 EN1   go3:
236 Zd2    IF r2<>0 AND r2<>12 THEN go5
237 KU1   go4:
238 OP2    LOCATE 2,1:INPUT " AG-Nummer Baisse";r2:IF r2<1 OR r2>1
        3 THEN go4
239 Qb1   go5:
240 JN2    CLS:PRINT:PRINT" Baisse:":PRINT" ----------------------
        ------------"
241 Gg     PRINT" AG              Diff.   Alt    Neu":PRINT
242 wc     r1=INT(RND*5)+1
243 9T     PRINT TAB(2)Ak$(r2)TAB(18)-r1*dw TAB(26)x(r2)TAB(33)x(r2)
        -r1*dw
244 ic     PRINT" ----------------------------------"
245 AU     PRINT:PRINT" Hausse:":PRINT" --------------------------
        --------"
246 Ll     PRINT" AG              Diff.   Alt    Neu":PRINT
247 yp     FOR ip=1 TO 13:IF ip=r2 THEN go6
248 uW3     LOCATE 12+ip,2
249 3K      PRINT Ak$(ip)TAB(18)r1*up TAB(26)x(ip)TAB(33)x(ip)+r1*up
250 fr1   go6:
251 gi2    NEXT ip:PRINT" ----------------------------------"
252 iA     LOCATE 27,2:INPUT"Akzeptieren Sie";jn$:IF jn$="n" OR jn$=
        "N" THEN hp=hp-3:GOTO kk
253 8c     FOR ip=1 TO 13
254 mF3     IF ip=r2 THEN x(ip)=x(ip)-r1*dw ELSE x(ip)=x(ip)+r1*up
255 tv2    NEXT ip:hp=hp-1:GOTO kk
256 o30 'Multiple Action Flood
257 wM  Flood:
258 5p1   COLOR 3,0:CLS
259 A3    PRINT:PRINT" Multiple Action Flood"
260 Cn    PRINT:PRINT" ----------------------------------"
261 yN    PRINT" AG              Diff.   Alt    Neu"
262 09    FOR ip=1 TO 13:r1=INT(RND*151)-75
263 Lg2    LOCATE 6+ip,2:PRINT Ak$(ip)TAB(18)r1;"%"TAB(27)x(ip)TAB(3
        4)x(ip)+INT(x(ip)*r1/100)
264 Qu     x(ip)=x(ip)+INT(x(ip)*r1/100):FOR t=1 TO 3000:NEXT t
265 4g1   NEXT ip:PRINT" ----------------------------------":PRIN
        T
266 3Z    PRINT:PRINT" Automatische Durchführung!":hp=hp-3:GOSUB tas
        t:GOTO kk
267 Zm0 'Manual 1
268 7a  Ma1:
269 b81   COLOR 1,14:CLS
270 n2    PRINT:PRINT" Brokers Manual Activity I":GOSUB Wahl
271 Ya    so:
272 Jk    LOCATE 22,1:INPUT" Welche AG-Nummer soll steigen";r1
273 In    IF r1<1 OR r1>13 THEN so
274 BV    so2:
275 pY    LOCATE 23,1:INPUT" Welche AG-Nummer soll fallen ";r2
276 zn    IF r2<1 OR r2>13 OR r1=r2 THEN so2
277 jG    COLOR 1,14:CLS
278 uX    PRINT:PRINT" Brokers Manual Activity I":r3=INT(RND*50)+1
279 LP    PRINT:PRINT" Es steigt"
280 NK    PRINT" ----------------------------------"
281 AW    PRINT" AG              Diff.   Alt    Neu":PRINT
282 BF    PRINT TAB(2)Ak$(r1)TAB(18)r3;"%"TAB(27)x(r1)TAB(34)INT(x(r
```

Listing 1. »Broker« geben Sie mit dem Checksummer (Seite 159) ein

```
      1)+x(r1)*r3/100)
283 Sd  x(r1)=INT(x(r1)+x(r1)*r3/100)
284 RO  PRINT" ------------------------------------"
285 F2  PRINT:PRINT" Es fällt:"
286 TQ  PRINT" ------------------------------------"
287 Gc  PRINT" AG              Diff.    Alt    Neu":PRINT
288 na  PRINT TAB(2)Ak$(r2)TAB(18)-r3;"%"TAB(27)x(r2)TAB(34)INT(x(
        r2)-x(r2)*r3/100)
289 s3  x(r2)=INT(x(r2)-x(r2)*r3/100)
290 XU  PRINT" ------------------------------------"
291 A5  PRINT:PRINT:PRINT" Automatische Durchführung!":hp=hp-3:GOS
        UB tast:GOTO kk
292 3H0 'Manual 2
293 a4  Ma2:
294 OX1 COLOR 1,14:CLS
295 cy  PRINT:PRINT" Brokers Manual Activity II"
296 2t  PRINT" ------------------------":PRINT
297 26  PRINT" <1>  Dividende / Aufwertung":PRINT" <2>  Hausse
        /Baisse (1)>  (1/2)<"
298 jW  PRINT" <3>  Hausse/Baisse (1)>  (1/2)<<":PRINT" <4>
         Hausse/Baisse (1)>> (1/2)<"
299 Nu  PRINT" <5>  Hausse/Baisse (1)>> (1/2)<<":PRINT" <6
        >  Multiple Action Flood"
300 bK  PRINT" <7>  Hausse/Baisse (1/2)>  (1)<":PRINT" <8>
        Hausse/Baisse (1/2)>  (1)<<"
301 hh  PRINT" <9>  Hausse/Baisse (1/2)>> (1)<":PRINT" <10>
         Hausse/Baisse (1/2)>> (1)<<"
302 7o  PRINT" <11> Manual Activity I":PRINT" <12> Finanzamt"
303 Gy  LOCATE 22,1:PRINT" Erklärung:  >= normal steigend"
304 Kn  PRINT"            >>= stark  steigend":PRINT"
          <= normal fallend"
305 nn  PRINT"            <<= stark  fallend"
306 Ft  ai:
307 3V2  LOCATE 18,1:INPUT" Börsenaktion-Nummer";r1:IF r1<1 OR r1
         >12 THEN ai
308 nS   ON r1 GOTO Divi,HaBa,HaBa,HaBa,HaBa,Flood,HaBa,HaBa,HaBa,
         HaBa,Ma1,Fiskus
309 TaO 'Fiskus
310 Kw  Fiskus:
311 q81  max=0:maxi=0
312 01   COLOR 4,0:CLS
313 f1   PRINT:PRINT" Der Fiskus bittet zur Kasse:":PRINT
314 0C   PRINT" Kapitalertragssteuer"
315 uy   PRINT" --------------------":PRINT
316 w5   PRINT" (25% des Barkapitals, sofern dieses "
317 AO   PRINT"  größer ist als der Freibetrag)"
318 rg   PRINT:PRINT" Startkapital:";kapital;"DM"
319 gO   PRINT" Freibetrag:  ";INT(kapital*1.5);"DM"
320 na   PRINT:PRINT:PRINT" Name          Bar DM   Stnetto   Steuer"
         :PRINT
321 sr   FOR ip=1 TO s:LOCATE 15+ip,2
322 WC2   IF kap(ip)>INT(kapital*1.5) THEN max=kap(ip)-INT(kapital
          *1.5):maxi=max/4
323 ZR    PRINT sp$(ip)TAB(14)kap(ip)TAB(23)INT(max)TAB(33)INT(maxi
          )
324 J3    IF kap(ip)>INT(kapital*1.5) THEN kap(ip)=kap(ip)-INT(max
          i):max=0:maxi=0
325 LT1  NEXT ip:PRINT:PRINT
326 oG   PRINT" Automatische Durchführung!":GOSUB tast:GOTO k2
327 i10 'Menüpunkt Börsenkurse
328 Iz  Kurse:
329 cF1  WINDOW 4,"Börsenkurse",(0,10)-(360,175),0,2:COLOR 7,1:CLS
330 XW   PRINT:PRINT " AG                EKurs     Kurs     Stück"
331 Wn   PRINT" ----------------------------------------"
332 gg   FOR i=1 TO 13:LOCATE 3+i,2:PRINT Ak$(i)TAB(20)z(i)TAB(30)x
         (i)TAB(39)v(i):: NEXT i
333 s4   PRINT:PRINT:PRINT" Bitte Taste drücken":GOTO kkk
334 akO 'Menüpunkt Charts
335 tK  Charts:
336 4o1  WINDOW 4,"Charts",(0,10)-(630,240),0,2
337 jc   kre(p)=kre(p)+60
338 Y2   ausw:
339 K22   COLOR 0,1:CLS
340 hM    LOCATE 3,2:PRINT"Bitte wählen Sie:"
341 cI    FOR i=1 TO 13:LOCATE 4+i,2
342 w43    PRINT Ak$(1)TAB(18)"<";i;">"
343 H12   NEXT i:LOCATE 24,1
344 ck    PRINT" Zurück zum Menü:   bei 1. < 0 > eingeben"
345 Xd    PRINT" Einzeldarstellung: bei 2. < 0 > eingeben"
346 ab1   eing1:
347 Gg2    LOCATE 20,2:INPUT"1. AG-Nummer (0-13) ";gr(1):IF gr(1)<0
           OR gr(1)>13 THEN eing
348 zd    IF gr(1)=0 THEN kk
349 jg1   eing2:
350 9w2    LOCATE 21,2:INPUT"2. AG-Nummer (0-13) ";gr(2):IF gr(2)<0
           OR gr(2)>13 THEN eing2
351 8j    IF gr(2)=0 THEN ee=1:gr(2)=22 ELSE ee=2
352 T2    COLOR 1,0:CLS:COLOR 4,0:PRINT TAB (2)"1.Aktie: "+Ak$(gr(1
          ))
353 IL    IF gr(2)<>22 THEN COLOR 3,0:PRINT TAB(2)"2.Aktie: "+Ak$
          (gr(2))
354 sN    COLOR 1,0:d=4
355 tO    LOCATE 5,1:PRINT" Over":PRINT" 2000":PRINT" 1900":PRINT"
          1800":PRINT" 1700":PRINT" 1600"
356 2W    PRINT" 1500":PRINT" 1400":PRINT" 1300":PRINT" 1200":PRINT
          " 1100":PRINT" 1000":PRINT"  900"
357 LU    PRINT"  800":PRINT"  700":PRINT"  600":PRINT"  500":PRINT
          "  400":PRINT"  300":PRINT"  200"
358 g4    PRINT"  100":PRINT" Flow":LINE(45,30)-(45,210)
359 dM    LOCATE 27,7:PRINT"1  2  3  4  5  6  7  8  9 10 11 12 13 1
          4 15 16 17 18 19 20 21 22"
360 JU    LINE(15,207)-(600,207)
361 GO    FOR ip=1 TO ee
362 ut3    ilm=0:iln=0:ilv=0:ilb=0:ila=0:ilc=0
363 4G     ilm=VAL(MID$(st$(gr(ip)),5,4)):iln=VAL(MID$(st$(gr(ip)),
           9,4))
364 86     ilv=207-(ilm/12):ilb=207-(iln/12)
365 OL     LINE(45,ilv)-(69,ilb),d
366 6z2   eing3:
367 3Y3    FOR io=3 TO 22
368 YD4     ila=VAL(MID$(st$(gr(ip)),io*4+1,4)):IF ila<20 THEN ein
            g4
369 XM      ilc=207-(ila/12)
370 JX      LINE -(io*24+21,ilc),d
371 Fm3    NEXT io
372 I71   eing4:
373 hT3    d=d-1
374 Lt2    NEXT ip
375 92     LOCATE 1,30:PRINT "Bitte Taste drücken":GOSUB tast2:GOTO
           ausw
376 rV0 'Menüpunkt Aktienkauf
377 LN  Aktkauf:
378 UC1  WINDOW 4,"Aktienkauf",(0,10)-(425,187),0,2:COLOR 13,1:CLS
379 EB   IF hp<2 THEN GOTO Notg
380 YB   PRINT:PRINT" Name: "+sp$(p)+"    Barkapital:";kap(p);"DM"
381 Of   PRINT:PRINT" Welche Aktie möchten Sie kaufen?          <0
         > Keine"
382 kL   PRINT:PRINT" ------------------------------------------
         -------"
383 Bu   PRINT"  Nr.       AG                    Vorrätig     Kurs"
384 X8   FOR i=1 TO 13
385 Xe2   LOCATE 8+i,2:PRINT TAB(3)"<";i;">"TAB(14)Ak$(i)TAB(34)v
          (i)TAB(47)x(i)
386 fv1  NEXT i
387 86   ein:
388 D12   LOCATE 4,33:INPUT anr:IF anr=0 THEN GOTO ein3 ELSE IF(anr
          <0 OR anr>13)THEN ein
389 I3    COLOR 13,1:CLS
390 gY    PRINT:PRINT" Aktiennummer:";anr:PRINT:PRINT" ----------
          ----------------------------------"
391 1z    PRINT" AG               Brief     Kurs     Vorrätig"
392 HL    LOCATE 7,2:PRINT Ak$(anr)TAB(18)z(anr)TAB(28)x(anr)TAB(37
          )v(anr)
393 dK    PRINT" ------------------------------------------"
394 1L    mini=v(anr):maxi=INT(kap(p)/x(anr))
395 Ao    IF mini>maxi THEN maxim=maxi ELSE maxim=mini
396 aU    PRINT:PRINT" Maximale Ankaufsmenge in Stück:";maxim
397 Us    PRINT:INPUT " Ankaufsmenge in Stück";akts
398 jK    IF akts>v(anr) THEN note$(anr)="BG" ELSE IF akts<1 THEN
           note$(anr)="BBR" ELSE GOTO ein2
399 1h    PRINT:PRINT" Ankaufsmenge zu klein!":GOSUB tast:GOTO Aktk
          auf
400 DN1  ein2:
401 Gx2   IF akts*x(anr)>kap(p)THEN PRINT:PRINT" Ihr Kapital reich
          t nicht aus!":GOSUB tast:GOTO Aktkauf
402 r7    an=anr:bn=x(an)
403 tX    PRINT:PRINT:PRINT" Kaufpreis:     ";akts*bn;"DM"
404 Ik    PRINT" Maklergebühr:  ";INT(akts*x(an)/50);"DM"
405 Er    PRINT" ----------------------":PRINT" Effektivsumme: ";
          akts*bn+INT(akts*x(an)/50);"DM"
406 iG    w(p,an)=w(p,an)+akts:note$(an)="GELD":kap(p)=kap(p)-akts*
          bn:v(an)=v(an)-akts
407 79    IF kap(p)<0 THEN kre(p)=kre(p)+ABS(kap(p)):kap(p)=0
408 kT    ka(p,an)=x(an)
409 4q    PRINT:PRINT" Anweisung ausgeführt! Bitte Taste drücken"
410 U1    hp=hp-2:kre(p)=kre(p)+INT((akts*x(an))/50):GOTO kkk
411 Te1  ein3:
412 Nm2   GOTO kk
```

```
413 dL0 'Menüpunkt Aktienverkauf
414 ML  Aktverk:
415 401  WINDOW 4,"Aktienverkauf",(0,10)-(425,187),0,2:COLOR 3,1:CL
         S
416 pm   IF hp<2 THEN GOTO Notg
417 9m   PRINT:PRINT" Name: "+sp$(p)+"    Barkapital:";kap(p);"DM"
418 38   PRINT:PRINT" Welche Aktie möchten Sie verkaufen?        <
         0> Keine"
419 Lw   PRINT:PRINT" ------------------------------------------
         -------"
420 cT   PRINT"   Nr.       AG                    Besitz       Kurs"
421 8j   FOR i=1 TO 13
422 sZ2   LOCATE 8+i,2:PRINT TAB(3)"<";i;">"TAB(14)Ak$(i)TAB(34)w
          (p,i)TAB(47)x(i)
423 GW1  NEXT i
424 he   eg1:
425 sS2   LOCATE 4,36:INPUT anr:IF anr=0 THEN GOTO ein3 ELSE IF(anr
          <0 OR anr>13)THEN eg1
426 Vs    IF w(p,anr)=0 OR note$(anr)="GELD" THEN notv ELSE GOTO ja
          ver
427 6h1  notv:
428 su2   PRINT" Verkauf nicht möglich! Bitte Taste drücken":GOSUB
          tast2:GOTO kk
429 LD1  javer:
430 wh2   COLOR 3,1:CLS
431 LD    PRINT:PRINT" Aktiennummer:";anr:PRINT:PRINT" ----------
          ----------------------------------"
432 xs    PRINT" AG                Brief     Kurs     Besitz"
433 Ox    LOCATE 7,2:PRINT Ak$(anr)TAB(18)z(anr)TAB(28)x(anr)TAB(37
          )w(p,anr)
434 Iz    PRINT" ------------------------------------------"
435 MM    PRINT:PRINT" Kauf der Aktie bei:";ka(p,anr);"für insgesam
          t";ka(p,anr)*w(p,anr);"DM"
436 p4    PRINT:PRINT" Effektivkaufpreis pro Aktie:";ka(p,anr)+INT(
          x(anr)/50);"DM"
437 Pu    PRINT:PRINT:INPUT" Verkaufsmenge in Stück";aktv
438 WZ    IF aktv<1 OR aktv>w(p,anr)THEN PRINT:PRINT" Verkaufsmen
          ge zu klein/groß!":GOSUB tast:GOTO Aktverk
439 kf    IF aktv*x(anr)<1 THEN PRINT:PRINT" Totales Verlustgeschä
          ft !!!":GOSUB tast:GOTO Aktverk
440 xt    an=anr:w(p,an)=w(p,an)-aktv:note$(an)="BRIEF":kap(p)=kap(
          p)+aktv*x(an):v(an)=v(an)+aktv
441 o7    PRINT:PRINT" Verkaufswert: ";x(an)*aktv;"DM":PRINT" Makle
          rgebühr: ";INT((x(an)*aktv)/50);"DM"
442 pm    kre(p)=kre(p)+INT((x(an)*aktv)/50):PRINT:PRINT " Anweisun
          g ausgeführt!  Bitte Taste drücken"
443 6i    hp=hp-2:GOTO kkk
444 3J0 'Menüpunkt Kapitalliste
445 U9  Kaplist:
446 QP1  WINDOW 4,"Kapitalliste",(0,10)-(200,165),0,2:COLOR 12,5:CL
         S
447 AL   GOSUB Aktka
448 Mn   PRINT:PRINT" Name: ";sp$(p)
449 xu   PRINT:PRINT:PRINT" Bargeld:   ";kap(p);"DM"
450 VE   PRINT:PRINT" Aktienwert:";Ak(p);"DM"
451 H3   PRINT:PRINT" ----------------------":PRINT" BRUTTO:     ";
         kap(p)+Ak(p);"DM"
452 iG   PRINT:PRINT" Schulden:  ";kre(p);"DM"
453 6R   PRINT:PRINT" ----------------------":PRINT" NETTO:      ";
         kap(p)+Ak(p)-kre(p);"DM"
454 p1   PRINT:PRINT:PRINT" Bitte Taste drücken":GOTO kkk
455 iv0 'Menüpunkt Kapitalvergleich
456 ZH  Kapver:
457 wJ1  IF no=0 THEN kre(p)=kre(p)+30:g$="Kapitalvergleich"
458 j6   misp:
459 pd2   IF s=2 THEN sp2 ELSE IF s=3 THEN sp3 ELSE sp4
460 EZ1  sp2:
461 7W2   WINDOW 4,g$,(0,10)-(580,90),0,2:COLOR 5,12:CLS
462 mS    GOTO bewe
463 Lh1  sp3:
464 Bu2   WINDOW 4,g$,(0,10)-(580,107),0,2:COLOR 5,12:CLS
465 pV    GOTO bewe
466 Sp1  sp4:
467 Cv2   WINDOW 4,g$,(0,10)-(580,125),0,2:COLOR 5,12:CLS
468 ij   bewe:
469 4N3    PRINT:PRINT" Name                Bargeld    Aktienwert
           Brutto   Schulden    Netto"
470 ic     PRINT" ------------------------------------------------
           ----------------------"
471 Yj     GOSUB Aktka
472 0d4     FOR ip=1 TO s
473 7t5      PRINT TAB(2)sp$(ip)TAB(20)kap(ip)TAB(31)Ak(ip)TAB(44)k
             ap(ip)+Ak(ip)TAB(53)kre(ip)TAB(65)kap(ip)+Ak(ip)-kre(ip
             )
```

```
474 h64     PRINT:NEXT ip
475 cu3     PRINT:PRINT" Bitte Taste drücken":no=0
476 Fr      GOTO kkk
477 n30 'Menüpunkt Aktienliste
478 RA  Aktlist:
479 eX1   WINDOW 4,"Aktienliste",(0,10)-(420,235),0,2:COLOR 2,0:CLS
480 hs    GOSUB Aktka
481 tK    PRINT:PRINT" Name: ";sp$(p)
482 PM    PRINT:PRINT" Unverbrieftes Kapital: ";kap(p);"DM"
483 TN    PRINT" Verbrieftes Kapital  : ";Ak(p);"DM"
484 Iz    PRINT" ------------------------------------------------"
485 pc    PRINT" AG                    Umsz.      Habe      Kurs":PRINT
486 Qm    FOR i=1 TO 13:LOCATE 8+i,2:PRINT Ak$(i)TAB(22)note$(i)TAB(
          31)w(p,i)TAB(40)x(i)
487 ps    NEXT i:PRINT" ------------------------------------------
          -----"
488 rO    PRINT:PRINT" Gesamtkapital: ";kap(p)+Ak(p);"DM"
489 Oa    PRINT:PRINT:PRINT" Bitte Taste drücken":GOTO kkk
490 Ik0 'Menüpunkt Aktienvergleich
491 YK  Aktver:
492 231   kre(p)=kre(p)+40
493 IR    IF s=2 THEN s2 ELSE IF s=3 THEN s3 ELSE s4
494 CF    s2:
495 Gw2   WINDOW 4,"Aktienvergleich",(0,10)-(360,170),0,2:COLOR 0,2
          :CLS
496 6o    PRINT:PRINT" AG"TAB(21)sp$(1)TAB(35)sp$(2)
497 CT    PRINT" ------------------------------------------"
498 w6    FOR ip=1 TO 13:LOCATE 3+ip,2
499 1s3    PRINT Ak$(ip)TAB(20)w(1,ip)TAB(34)w(2,ip)
500 QV2   NEXT ip:PRINT:PRINT:PRINT" Bitte Taste drücken":GOTO kkk
501 MQ1  s3:
502 SA2   WINDOW 4,"Aktienvergleich",(0,10)-(470,170),0,2:COLOR 0,2
          :CLS
503 kT    PRINT:PRINT" AG"TAB(21)sp$(1)TAB(35)sp$(2)TAB(48)sp$(3)
504 ga    PRINT" ------------------------------------------------
          ------"
505 3D    FOR ip=1 TO 13:LOCATE 3+ip,2
506 e13    PRINT Ak$(ip)TAB(20)w(1,ip)TAB(34)w(2,ip)TAB(47)w(3,ip)
507 Xc2   NEXT ip:PRINT:PRINT:PRINT" Bitte Taste drücken":GOTO kkk
508 Wb1  s4:
509 bK2   WINDOW 4,"Aktienvergleich",(0,10)-(570,170),0,2:COLOR 0,2
          :CLS
510 5p    PRINT:PRINT" AG"TAB(21)sp$(1)TAB(35)sp$(2)TAB(48)sp$(3)TA
          B(61)sp$(4)
511 qK    PRINT" ------------------------------------------------
          ------------------"
512 AK    FOR ip=1 TO 13:LOCATE 3+ip,2
513 xP3    PRINT Ak$(ip)TAB(20)w(1,ip)TAB(34)w(2,ip)TAB(47)w(3,ip)T
           AB(60)w(4,ip)
514 ej2   NEXT ip:PRINT:PRINT:PRINT" Bitte Taste drücken":GOTO kkk
515 3R0 'Menüpunkt Nächster
516 7i  Nexte:
517 rR1   CALL requester(" Spielerwechsel?","Ja","Nein")
518 yx    LOCATE 18,2
519 xy    IF req=2 THEN WINDOW OUTPUT 2:req=0:GOTO c
520 Z8    IF req=1 THEN wech
521 2C    GOTO Nexte
522 DI    wech:
523 fe2   WINDOW 4,"Spielerwechsel",(0,10)-(330,240),0,2:COLOR 1,5:
          CLS
524 L4    PRINT:PRINT" Bitte warten Sie: AMIGA rechnet!":hp=5
525 kP    IF kre(p)>0 AND kap(p)>kre(p) THEN GOSUB Schulden
526 Au    IF kre(p)>0 AND kap(p)<kre(p) THEN GOSUB Zinsen
527 Tr    GOSUB Aktka:krelim=kapital/2
528 rS    FOR i=1 TO 13
529 AM3    IF x(i)<50 THEN GOSUB Low ELSE IF x(i)>2000 THEN GOSUB
           High
530 mu     h$=STR$(x(i)):IF LEN(h$)=4 THEN h$=" "+h$ ELSE IF LEN(h$
           )=3 THEN h$="  "+h$
531 nt     st$(i)=MID$(st$(i),5,88)+MID$(h$,2,4)
532 PU     IF note$(i)="GELD" THEN note$(i)="BZ"
533 2I2   NEXT i
534 OX    IF (kap(p)+Ak(p))>=spielende AND kre(p)=0 THEN endes=1
535 fH    IF ((kap(p)+Ak(p)-kre(p)<100)OR kre(p)>krelim)THEN ende
          s=1
536 c0    p=p+1:IF p>s THEN p=1
537 1y    IF endes THEN Spielend
538 h1    PRINT:PRINT:PRINT" Nächster Spieler ist: ";sp$(p)
539 G0    PRINT:PRINT" Bitte Taste drücken":GOTO kkk
540 Lj1  Spielend:
541 jX2   GOSUB tast:WINDOW CLOSE 4:WINDOW OUTPUT 2
542 gH    g$="Schlußberechnung":CLS:GOTO misp
```

Listing 1. (Fortsetzung)

```
543 BN0 'Menüpunkt Uebersicht
544 W8  Uebers:
545 gw1  WINDOW 4,"Uebersicht",(0,10)-(250,140),0,2:COLOR 5,6:CLS
546 WO   PRINT:FOR i=1 TO s:PRINT" Spieler";i;":   ";sp$(i):PRINT:N
         EXT i
547 h4   PRINT:COLOR 6,5:LOCATE ,2:PRINT "Startkapital :";kapital;"
          DM"
548 wI   LOCATE ,2:PRINT"Spielende bei:";spielende;"DM":COLOR 5,6:P
         RINT
549 IW   GOSUB tast:GOTO kk
550 5t0 'Menüpunkt Help
551 Oz  Help:
552 fD1  WINDOW 4,"BROKER - Das Börsenspiel von Sebastian Peetz",(0
         ,10)-(630,200),0,2
553 TO   COLOR 5,15:CLS
554 Iv   PRINT
555 8S   PRINT " Menüpunkt                                    Erklärung
                          DM] HP"
556 zT   PRINT "-------------------------------------------------
         ------------------------------"
557 Fx   PRINT "Laden                        Ein Spiel wird von Disk gel
         aden                    ]"
558 Bv   PRINT "Speichern                    Ein Spiel wird abgespeicher
         t                       ]"
559 zP   PRINT "Neues Spiel                  Es wird ein neues Spiel beg
         onnen                   ]"
560 xH   PRINT "Quit                         Spielende ohne die Spielstä
         nde abzuspeichern       ]"
561 41   PRINT "Börsenaktion                 Der Spieler kann aktiv die
         Börsenkurse            20] 1-3"
562 uS   PRINT "                             durch geschicktes Handeln b
         eeinflussen             ]"
563 Mr   PRINT "Börsenkurse                  Es werden die aktuellen Bör
         senkurse angezeigt     - ] -"
564 VJ   PRINT "Charts                       Aktienkurse können graphisc
         h dargestellt werden   60] -"
565 Mj   PRINT "Aktienkauf                   Aktien können zum Tageskurs
          gekauft werden        5%] 2"
566 H1   PRINT "Aktienverk.                  Aktien können zum Tageskurs
          verkauft werden       5%] 2"
567 3G   PRINT "Kapitalliste                 Eigenes Kapital (Aktienkapi
         tal und bar)           - ] -"
568 iw   PRINT "Kapitalvergl.                Kapital aller Spieler
                                30] -"
569 m6   PRINT "Aktienliste                  Auflistung der eigenen Akti
         en (Kurse,Stück)       - ] -"
570 WZ   PRINT "Aktienvergl.                 Aktien aller Spieler (Stück
         )                      40] -"
571 aJ   PRINT "Nächster                     Der nächste Spieler ist an
         der Reihe               ]"
572 XQ   PRINT "Uebersicht                   Alle wichtigen Daten zum Sp
         iel im überblick        ]"
573 IT   PRINT "Info                         Diese Seite
                                 ]"
574 rb   PRINT:PRINT "Bitte Taste drücken":GOTO kkk
575 FP0 'Unterprogramme
576 jf  'Subprogramme für Menüpunkt Börsenaktion/Hausse-Baisse
577 WA  Wahl:
578 hW1  LOCATE 4,1:PRINT" ---------------------------------------
         "
579 bn   PRINT"   Nr.    AG                  Habe    Kurs"
580 hI   FOR i=1 TO 13
581 z12   LOCATE 6+i,2:PRINT "<";i;">"TAB(11)Ak$(i)TAB(28)w(p,i)T
          AB(36)x(i)
582 391  NEXT i:PRINT" ---------------------------------------"
583 lN   RETURN
584 h20 'Subprogramme für Menüpunkt Nächster
585 qG  Schulden:
586 WR1  PRINT:PRINT:PRINT" Schulden:":PRINT" ---------"
587 99   PRINT:PRINT " "+sp$(p)+" muß ";kre(p);"DM  bezahlen."
588 EN   kap(p)=kap(p)-kre(p):kre(p)=0
589 rT   RETURN
590 620 Zinsen:
591 tO1  PRINT:PRINT:PRINT"  Schulden: ";kre(p);"DM":PRINT"+ Zinsen
         :   ";CINT(kre(p)/10);"DM"
592 Ki   PRINT" -----------------":PRINT:PRINT"  Gesamt:   ";kre(p
         )+CINT(kre(p)/10);"DM"
593 Io   kre(p)=kre(p)+CINT(kre(p)/10)
594 wY   RETURN
595 YM0 Low:
596 Qj1  PRINT:PRINT:PRINT" Aufwertung:":PRINT" -----------":PRINT
597 gs   PRINT" Kurswert zu niedrig bei: "+Ak$(i)+" !"
598 mY   PRINT" Aufwertung je Aktie    :";(z(i)-x(i));"DM":PRINT
599 Rg   FOR ip=1 TO s
```

```
600 9I2   IF kap(ip)<w(ip,i)*(z(i)-x(i)) THEN verl
601 9n    PRINT TAB(2)sp$(ip)TAB(13)"muß"TAB(17)w(ip,i)*(z(i)-x(i))
          ;"DM zahlen."
602 kD    kap(ip)=kap(ip)-w(ip,i)*(z(i)-x(i))
603 MG    GOTO weiter
604 nA1  verl:
605 sk3     PRINT TAB(2)sp$(ip)TAB(13)"- AKTIENVERLUST -"
606 FX      kap(ip)=kap(ip)+w(ip,i)*x(i):v(i)=v(i)+w(ip,i):w(ip,i)=0
607 OL1  weiter:
608 7f2   NEXT ip
609 1v    x(i)=z(i)
610 07    PRINT:PRINT" - TASTE -":GOSUB tast2
611 Dp1   RETURN
612 b30 High:
613 Jo1   PRINT:PRINT:PRINT" Dividende:":PRINT" ----------":PRINT
614 pP    PRINT" Kurswert zu hoch bei:  "+Ak$(i)+" !"
615 7S    PRINT" Auszahlung je Aktie : ";(x(i)-2000);"DM":PRINT
616 ix    FOR ip=1 TO s
617 OK2   PRINT TAB(2)sp$(ip)TAB(13)"erhält"TAB(20);(w(ip,i)*(x(i)-
          2000));"DM"
618 wy    kap(ip)=kap(ip)+(w(ip,i)*(x(i)-2000))
619 Iq1   NEXT ip
620 9D    x(i)=2000
621 BI    PRINT:PRINT" - TASTE -":GOSUB tast2
622 O0    RETURN
623 k50 'Sonstige Subprogramme
624 Mt  Aktka:
625 r61   FOR ip=1 TO s
626 fQ2    Ak(ip)=0
627 S33     FOR i=1 TO 13
628 h14      Ak(ip)=Ak(ip)+x(i)*w(ip,i)
629 aq3     NEXT i
630 wZ1   NEXT ip:RETURN
631 9z0 Notg:
632 hM1   PRINT:PRINT:PRINT" Sie haben nicht genug Handlungspunkte!"
633 mW    PRINT:PRINT" Bitte Taste drücken":GOTO kkk
634 yh0 Hpu:
635 7f1   LOCATE 11,24:PRINT "Spieler: ";sp$(p)
636 3Y    LINE(175,90)-(460,125),3,BF
637 5U    LOCATE 14,23:COLOR 8,3
638 6V    PRINT" Sie haben noch";hp;"Handlungspunkte! "
639 4M    COLOR 1,0
640 gI    RETURN
641 kd0 NoHpu:
642 Em1   LOCATE 11,24:PRINT "Spieler: ";sp$(p)
643 Go    LINE(175,90)-(490,125),3,BF
644 Cb    LOCATE 14,23:COLOR 8,3
645 EL    PRINT" Sie haben keine Handlungspunkte mehr! "
646 BT    COLOR 1,0
647 nP    RETURN
648 Nm0 Fehler:
649 by1   WINDOW 5,"ERROR - ERROR - ERROR",(310,10)-(610,80),0,2:COL
          OR 1,5:CLS
650 i3    PRINT:PRINT" Es ist folgender Fehler aufgetreten:":PRINT
651 kc    IF ERR=53 THEN PRINT" DATEI NICHT GEFUNDEN !":GOTO dudi
652 4Y    IF ERR=61 THEN PRINT" DISKETTE IST VOLL !":GOTO dudi
653 my    IF ERR=64 THEN PRINT" FALSCHER DATEINAME !":GOTO dudi
654 iJ    IF ERR=67 THEN PRINT" ZUVIELE DATEIEN !":GOTO dudi
655 Vh    IF ERR=70 THEN PRINT" DISKETTE SCHREIBGESCHÜTZT !":GOTO du
          di
656 Wi    IF ERR=74 THEN PRINT" UNBEKANNTE DISKETTE":GOTO dudi
657 Xx    PRINT " UNBEKANNTER FEHLERTYP!"
658 im    dudi:
659 QU2    PRINT:GOSUB tast:WINDOW CLOSE 5:RESUME kk
660 SJ0 'Häufig verwendete Sub-Programme
661 Ly  k:
662 2k1   WINDOW CLOSE 4
663 CU    WINDOW OUTPUT 2
664 Yo    GOTO Kurse
665 hc0 k2:
666 6o1   WINDOW CLOSE 4
667 GY    WINDOW OUTPUT 2
668 VN    no=1:GOTO Kapver
669 WM0 kk:
670 As1   WINDOW CLOSE 4
671 Kc    WINDOW OUTPUT 2
672 gk    GOTO c
673 qt0 kkk:
674 JW1   IF INKEY$="" GOTO kkk
675 Fx    WINDOW CLOSE 4
676 Ph    WINDOW OUTPUT 2
677 Oc    IF endes THEN Ende
678 mq    GOTO c
679 Ss0 tast:
```

```
680 2M1   PRINT:PRINT " Bitte Taste drücken"
681 jJ    WHILE INKEY$=""
682 RF    WEND
683 Nz    RETURN
684 bx0 tast2:
685 nN1   WHILE INKEY$=""
686 VJ    WEND
687 R3    RETURN
688 tv0 'Diverse Daten
689 ER  papiere:
690 Z91   DATA Bauwerte,Textil,Brauerei,Kosmetik,Elektrizität,Chemie
          ,Pharmazie
691 Ic    DATA Maschinenbau,Kaufhaus,Computer,Automobil,Banken,Versi
          cherung
692 0D0 menua:
693 dS1   DATA 5
694 cQ    DATA 4
695 cz    DATA 1,Broker
696 rh    DATA 1,"Laden       "
697 HR    DATA 0,"Speichern   "
698 wQ    DATA 1,"Neues Spiel"
699 6n    DATA 1,"Quit        "
700 gT    DATA 3
701 GI    DATA 0,Börse
702 G5    DATA 0,"Börsenaktion"
703 16    DATA 0,"Börsenkurse "
704 ON    DATA 0,"Charts      "
705 jV    DATA 2
706 I6    DATA 0,Aktien
707 Oh    DATA 0,"Aktienkauf    "
708 Vt    DATA 0,"Aktienverkauf"
709 rf    DATA 4
710 64    DATA 0,Kapital/Aktienwerte
711 FM    DATA 0,"Kapitalliste      "
712 qu    DATA 0,"Kapitalvergleich  "
713 Dt    DATA 0,"Aktienliste       "
714 fU    DATA 0,"Aktienvergleich   "
```

```
715 vi    DATA 3
716 El    DATA 1,Information
717 7H    DATA 0,"Nächster   "
718 HP    DATA 0,"Uebersicht"
719 1k    DATA 1,"Help       "
720 YhO menub:
721 5u1   DATA 5
722 4s    DATA 4
723 4R    DATA 1,Broker
724 J9    DATA 1,"Laden       "
725 1w    DATA 1,"Speichern   "
726 Os    DATA 1,"Neues Spiel"
727 YF    DATA 1,"Quit        "
728 8v    DATA 3
729 kn    DATA 1,Börse
730 ka    DATA 1,"Börsenaktion"
731 Fb    DATA 1,"Börsenkurse "
732 ss    DATA 1,"Charts      "
733 Bx    DATA 2
734 mb    DATA 1,Aktien
735 sC    DATA 1,"Aktienkauf    "
736 zO    DATA 1,"Aktienverkauf"
737 J7    DATA 4
738 aZ    DATA 1,Kapital/Aktienwerte
739 jr    DATA 1,"Kapitalliste      "
740 KP    DATA 1,"Kapitalvergleich  "
741 hO    DATA 1,"Aktienliste       "
742 9z    DATA 1,"Aktienvergleich   "
743 NA    DATA 3
744 gD    DATA 1,Information
745 bm    DATA 1,"Nächster   "
746 1u    DATA 1,"Uebersicht"
747 TC    DATA 1,"Help       "
(C) 1988 M&T
```

Listing 1. (Schluß)

Der pfiffige Fußball-Manager

Fußballfans aufgepaßt: Hier kommt »Anpfiff« — eine starke Simulation. Die Grenzen zwischen Traum und Realität zerfließen: Dieses Spiel zwischen Sieg und Niederlage wird auch Sie begeistern. Führen Sie als Manager das Team Ihrer Wahl zur Meisterschaft — oder in den Abstieg.

Der Anpfiff ertönt im Stadion, das mit 75000 Fans gefüllt ist. Wird in diesem Spiel die Meisterschaft entschieden? Gebannt sitzen Sie auf der Tribüne, verfolgen von oben die Ereignisse auf dem Rasen. Die Weichen für den Erfolg haben Sie mit geschickter Ein- und Verkaufspolitik gestellt. Nun kommt es nur noch auf Ihre Jungs unten auf dem grünen Rasen an. Werden Ihre teuer erkauften Kicker das scheinbar Unmögliche wahr machen?

Die oben beschriebene Situation kennen die Manager der Fußball-Bundesliga ganz genau. Diese gut bezahlten Spezialisten sorgen dafür, daß es Ihrer Mannschaft am Saisonende nicht »an den Kragen geht«. Durch geschickte Handlungsweise sorgen Männer wie Hoeneß, Ribbek oder Grashoff dafür, daß am Ende der Saison ein Plus für ihren Verein herausspringt.

Bild 1. Führen Sie Ihre Lieblingsmannschaft durch eine spannende Bundesliga-Saison

Im Schweiße des Angesichts

Bei »Anpfiff« (Bild 1) schlüpfen Sie in die Rolle eines Bundesliga-Managers. Wählen Sie Ihre Lieblingsmannschaft, lenken Sie die Geschicke dieses Teams über eine komplette Saison. Keine Angst: Bei Mißerfolgen werden Sie nicht entlassen. Der Eintrag in die Liste der besten Manager aller Zeiten bleibt Ihnen in diesem Fall allerdings verwehrt.

Die Möglichkeiten eines Ma-

nagers sind zahlreich. Es kommt darauf an, die gebotenen Instrumente geschickt anzuwenden. Bei Anpfiff stehen folgende Maßnahmen bereit:
1. Mannschaft aufstellen,
2. Spieler kaufen und verkaufen,
3. Toto-Tips zur Erhöhung des Kapitals,
4. Baumaßnahmen zur Vergrößerung des eigenen Stadions.

die Spielernamen verändern. Außerdem sind die Namen der Spieler zu ändern, die für den Kauf bereitstehen und die Teamnamen im UEFA-Cup.

Wenn Sie die Namen der Spieler und der Mannschaften in der Version ändern wollen, die Sie im Heft zum Abtippen finden, manipulieren Sie einfach die entsprechenden DATA-Zeilen im Listing.

Bild 2. Wählen Sie das Team aus, dessen Geschicke Sie eine Saison lang bestimmen wollen

Ziel des Spiels ist es, Ihre Mannschaft zum Meistertitel zu führen. Daneben können Sie sich aber auch noch im UEFA-Cup versuchen (gilt nur für die Version auf der Programmservice-Diskette).

Nach der Eingabe des Listings mit dem Checksummer (Seite 159) speichern Sie das Programm bitte auf Diskette. Laden Sie Amiga-Basic und geben im Direktmodus ein:

```
CLEAR,120000,8000
```

Laden Sie anschließend »Anpfiff« und starten das Programm mit RUN.

Folgen Sie nun den Anweisungen des Programms. Geben Sie also zuerst Ihren Namen ein, anschließend wählen Sie den Bundesliga-Verein, den Sie führen möchten (Bild 2). Folgende Menüs warten auf Aktivitäten:

Fingerspitzen und Härte

Namen ändern

(Dieser Programmpunkt existiert nur bei der Version auf der Programmservice-Diskette.)

In diesem Menü können Sie die Mannschaftsnamen und

Tototip

In diesem Programmteil (Bild 3) können Sie versuchen, Ihr Geld zu vermehren. Tippen Sie die Ergebnisse der Begegnungen in der nächsten Bundesliga-Runde. Die Eingabe erfolgt nach den üblichen Regeln, also eine »1« für einen Heimsieg, eine »0« für Unentschieden und eine »2« für einen Auswärtssieg. Bei fünf

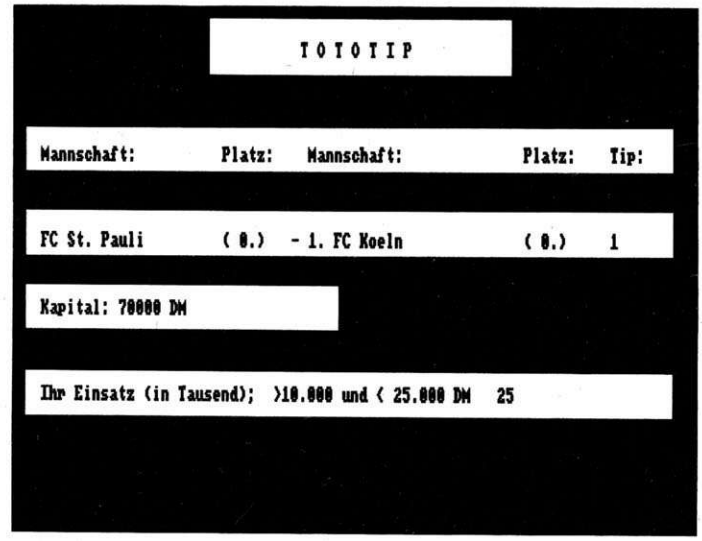


Bild 3. Der Toto-Tip bringt bei Erfolg bares Geld — Manager-Punkte gibt es zusätzlich.

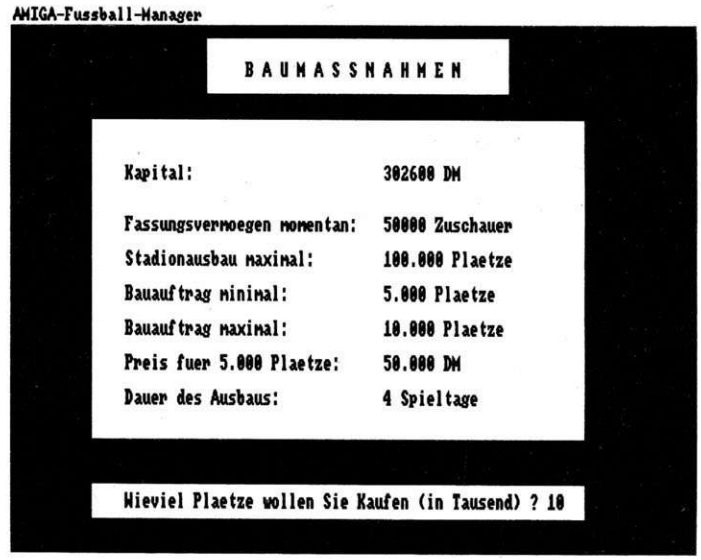


Bild 4. Vergrößern Sie Ihr Stadion möglichst früh in der Saison. Mehr Zuschauer passen in das erweiterte Stadion, die Einnahmen steigen, der Kassierer strahlt.

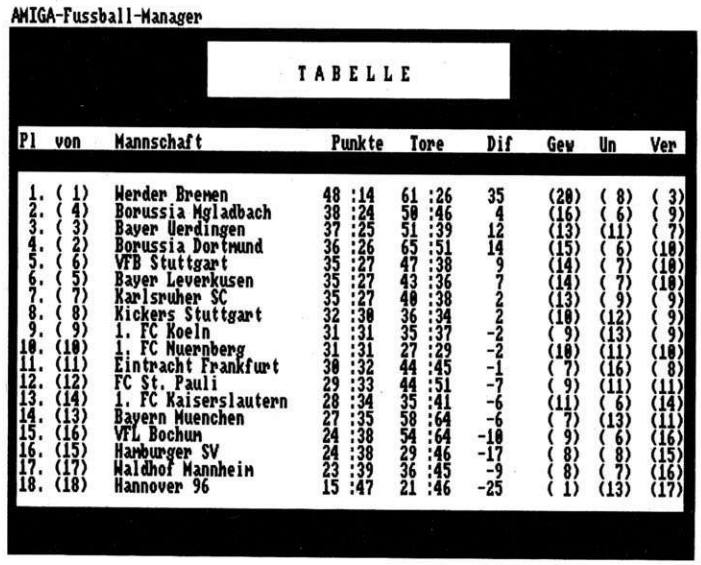


Bild 5. Die Tabelle zeigt den Stand der Liga. Hoffentlich ist Ihr Rückstand auf den Ersten nicht zu groß.

richtig getippten Ergebnissen erhalten Sie Ihr Geld zurück, bei mehr als fünf richtigen Voraussagen erhöht sich der Gewinn entsprechend des von Ihnen eingesetzten Betrages. Maximal ist ein Betrag von 25000 Mark einzusetzen, das Minimum beträgt 10000 Mark.

Transfer

In diesem Menü können Sie Spieler kaufen, verkaufen, oder einen Ihrer Spieler für einen Spieltag verleihen. Ferner können Sie im Punkt »Übersicht« eine Tabelle aufrufen, die den momentanen Stand der zu kaufenden Spieler darstellt. Nach dem ersten Spieltag steht lediglich ein Spieler zum Verkauf. Allerdings kommt nach jedem Spieltag, bis zum dreizehnten einschließlich, ein neuer Spieler zum Angebot hinzu.

Sie können allerdings nur Spieler kaufen, verkaufen oder verleihen, wenn Sie nach einem Spieltag ein entsprechendes Angebot bekommen (Anzeige auf dem Bildschirm).

Weiterhin dürfen Sie in einer Saison (34 Spieltage) höchstens vier Spieler kaufen. Deshalb sollten Sie bei entsprechenden Angeboten und eventuell vorhandenem Kapital nicht einfach drauflosKaufen. Warten Sie besser, bis Ihr Vermögen ausreicht, um einen der »großen Fische« an Land zu ziehen.

Baumaßnahmen

Hier können Sie das Fassungsvermögen Ihres Stadions aufstocken (Bild 4). Näheres dazu ergibt sich im Programmteil selbst. Allerdings wird dieser Part von einem Zufallsgenerator gesteuert, so daß Sie nicht zu jeder Zeit Bauaufträge vergeben können. Sie erhalten dann eine entsprechende Meldung.

Tabelle zeigen

Zeigt die aktuelle Tabellensituation (Bild 5). Vor dem ersten Spieltag ist keine Ausgabe möglich.

Spielerstatus zeigen

In einer Tabelle werden alle Spieler Ihres Teams aufgelistet. Diese Tabelle gibt Aufschluß über die gelben Karten eines Spielers sowie seinen Spielerfaktor, seinen Monatslohn und seinen Verkaufswert. Diese Übersicht ist sehr wichtig, da hier grundlegende Informationen darüber erscheinen, ob Sie einen Spieler eventuell verkaufen sollten, oder ob ein Spieler mit drei geschossenen Toren lieber in das Team eingewechselt werden sollte (siehe unten, Faktorerhöhung).

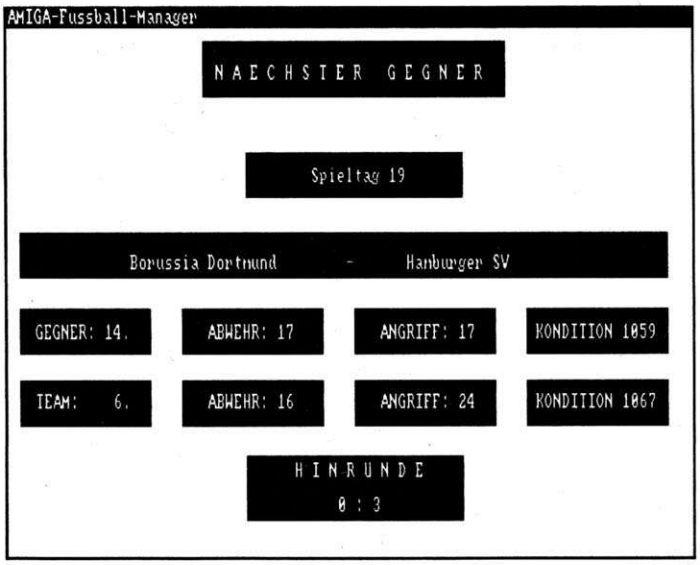


Bild 6. Die Stärken und Schwächen des nächsten Gegners sollten Sie bei der Mannschaftsaufstellung berücksichtigen

Bild 7. Feilen Sie an der Taktik, wechseln Sie Spieler mit schwächerer Kondition aus

Nächsten Gegner zeigen

Einer der informativsten Punkte ist wohl dieses Menü, in dem Sie die Spielstärke Ihres nächsten Gegners erfahren (Bild 6).

Dazu wird neben dem Tabellenplatz, der Angriffs- und Abwehrstärke und der Kondition der gesamten Mannschaft, nach dem 17. Spieltag auch das Ergebnis aus der Hinrunde angezeigt. Nach dieser Hintergrundinformation können Sie Ihre taktischen Überlegungen anstellen. Sollte Ihre Mannschaft vehement stürmen oder besser defensiv spielen (mehr Verteidiger aufstellen).

Mannschaft aufstellen

Hier gestalten Sie die Mannschaft individuell, passen diese dem zu erwartenden Gegner an. Nachdem Sie diesen Menüpunkt gewählt haben, erscheint die Liste Ihrer Spieler (Bild 7). Wenn Sie einen Wechsel vornehmen wollen, geben Sie bitte <j> ein. Wählen Sie anschließend den Spieler, der ausgewechselt werden soll und geben dessen Nummer gefolgt von <RETURN> ein.

Spieltag zeigen

Der nächste Spieltag mit allen anstehenden Begegnungen wird aufgelistet.

Spieltag spielen

Geben Sie den Start frei für die anstehende Auseinandersetzung — wir wünschen Ihnen einen Kantersieg. Das

Optimale Vereinsführung

Spiel dauert zweimal 45 Minuten (keine Echtzeit!). Die Minuten verrinnen, doch da erscheint die Anzeige »TOR« (Bild 8), wer hat zugeschlagen? Kurz darauf atmen Sie auf, das von Ihnen betreute Ensemble hat zugeschlagen (Bild 9). Nach Ablauf der ersten Halbzeit stoppt das Programm, nach kurzer Pause geht's weiter. Nach Ablauf der 90

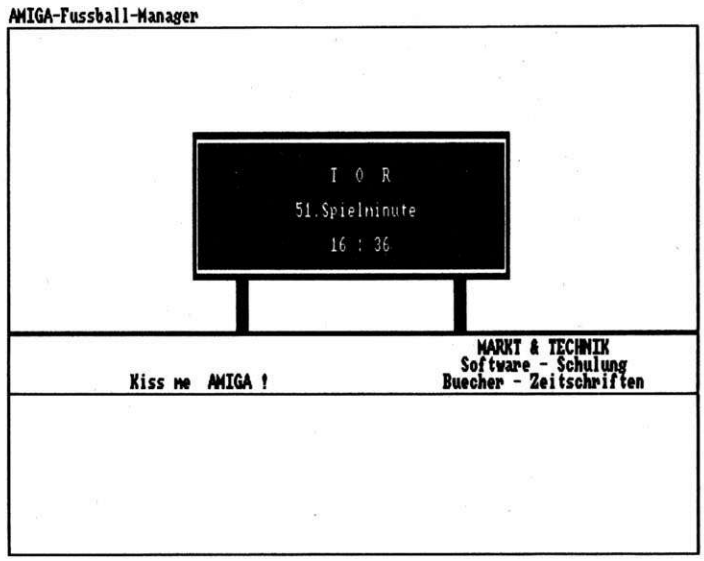


Bild 8. Der Torschrei hallt durchs Stadion, wer hat denn nun getroffen?

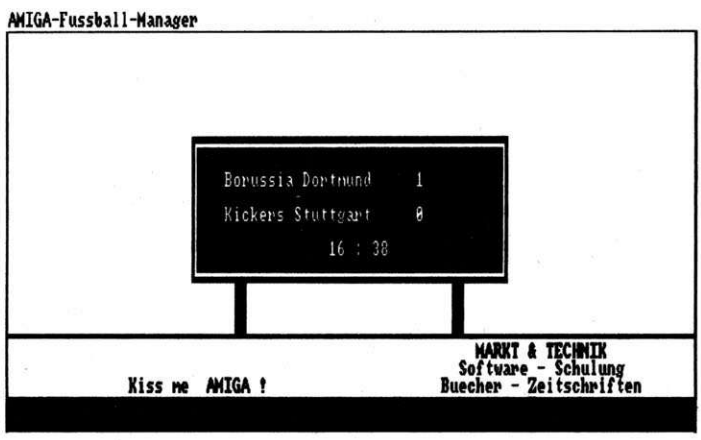


Bild 9. Das 1:0 fiel in diesem Schlagerspiel bereits in der achten Minute für die Dortmunder Borussia

schweißtreibenden Minuten ist der erste doppelte Punktgewinn hoffentlich im Kasten.

Nach jedem Spieltag erhalten Sie zusätzliche Informationen, die beispielsweise Verletzungen, rote Karten und geschossene Tore betreffen (Bild 10). Folgendes gilt es zu beachten:

Nach der 2. gelben Karte ist ein Spieler für ein Spiel gesperrt. Sieht ein Spieler »Rot«, ist er automatisch für die nächsten Begegnungen auf Eis gelegt. Es gibt keine Sportgerichtsverhandlung, das Strafmaß ist festgelegt.

Die Stärke einer Mannschaft ergibt sich aus der Addition aller eingesetzten Einzelspieler. Dabei wird zwischen Abwehr und Angriff unterschieden. Zur Abwehr gehört auch der Torwart; das Mittelfeld wird zum Angriff gezählt.

Der Heimvorteil wird per Zufallsgenerator gesteuert und kann ein Tor für die Heimmannschaft bedeuten. Die Chance dabei beträgt 50 Prozent.

Gleiches gilt für die Kondition, auch das Übergewicht auf diesem Bereich kann ein Tor bedeuten (Wahrscheinlichkeit 50:50). Für die Aufstellung der Mannschaft bedeutet das: Sie sollten versuchen, die Kondition der gegnerischen Mannschaft zu überbieten. In Heimspielen ist, wie auch in der Realität, ein Punktgewinn wahrscheinlicher als bei einem Auswärtsspiel. Das sollten Sie auch bei einem Toto-Tip berücksichtigen. Im UEFA-Cup (nur bei der Version auf der Programmservice-Diskette) gibt es ein Hin- und ein Rückspiel. Das Weiterkommen wird dann durch Addieren der erzielten Punkte und Tore entschieden (Bild 11).

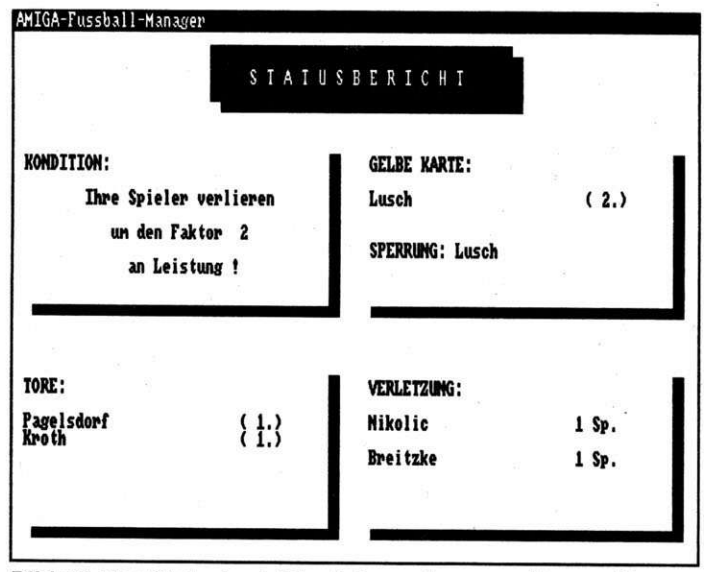


Bild 10. Der Statusbericht zeigt geschossene Tore, gelbe und rote Karten sowie Verletzungen

Faktorerhöhung der Spieler

Wichtig ist, daß Sie die Kondition eines Spielers nie unter 70 Prozent sinken lassen. Ein Spieler, dessen Kondition diesen Wert unterschreitet, ist automatisch für den Rest der Saison verletzt. Er verliert dann erheblich an Verkaufswert.

Der Verkaufswert eines Spielers steigt mit der Anzahl seiner geschossenen Tore, allerdings nur bis zum sechsten erzielten Tor.

Der Faktor eines Spielers steigt nach dem vierten geschossenen Tor um den Wert 1.

Damit ist gewährleistet, daß Ihre Mannschaft auch ohne Neuerwerbungen an Stärke gewinnt. Die Werte der Gegner in der Bundesliga erhöhen sich allerdings ebenfalls im Laufe der Saison.

Wenn Ihr Kapital das dritte Mal die Grenze von 0 Mark unterschritten hat, sind Sie von der Vereinsführung fristlos entlassen. Das Spiel ist in diesem Fall sofort beendet — starten Sie am besten gleich einen neuen Versuch. Zur Kontrolle Ihrer Finanzlage erhalten Sie nach jedem Spieltag eine Abrechnung der Einnahmen und Ausgaben (Bild 12).

Bild 11. Die Ergebnisse im UEFA-Cup. Dieser Wettbewerb ist nur in der Version auf der Diskette enthalten.

Hinweise zu Installation

Geben Sie »Anpfiff« (Listing 1) bitte mit dem Checksummer (Seite 159) ein. Speichern Sie das Programm bitte auf eine Diskette, auf der sich auch Amiga-Basic befindet. Sie finden dieses Programm auf der jedem Amiga beiliegenden »Extras«-Diskette. Vor dem Starten des Programms »Anpfiff« laden Sie bitte Amiga-Basic. Geben Sie dann im Direktmodus folgende Zeile ein:

```
CLEAR,120000,8000
```

Schließen Sie die Zeile mit <RETURN> ab und laden erst dann das Programm »Anpfiff« mit

```
LOAD "ANPFIFF",r
```

Sollten Sie den Wunsch haben, die Spieler- und Vereinsnamen zu wechseln, ändern Sie die entsprechenden DATA-Zeilen im Listing (Zeilen 1687 bis 1692). Besitzer der Version auf der Programmservice-Diskette nehmen diese Änderungen natürlich mit der nur dort enthaltenen Funktion 1, »Namen ändern«, vor.

Programmname:	Anpfiff
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

```
Programmautor: Thorsten Froese
------------------------------

  1 EAO REM Programm von
```

```
 2 zQ  REM Thorsten Froese, 4600 Dortmund 41
 3 Mb  REM fuer Markt & Technik im November 1988
 4 eq  TOPIC:
 5 di  REM Fenstergroesse definieren
 6 VZ  SCREEN CLOSE 1:WINDOW CLOSE 1
 7 zz  CLS:SCREEN 1,640,256,3,2
 8 zU  WINDOW 1,"AMIGA-Fussball-Manager",,0,1
 9 6g  PRINT"Einen Moment bitte..."
10 S2  DIM na$(20),po$(20),zu$(20),TE$(18),tw$(18),E$(18),a(162),
       MN$(18),Vna$(13),Kaufko(13)
11 1j  DIM fa(20),ko(20),pu(18),pq(18),E(18),Lohnmo(18),Verkaufs(
       19),Vfa(13)
12 uI  DIM kn(18),ks(27),ka(27),TKON(18),Tore(18),INSTORE(18),HEI
       MTORE(18),IARSCH(19)
13 4v  DIM TABPUN(18),TOTOPUN(18),TABTORE(18),TABGETORE(18),GEWIN
       N(18),Vpo$(13),STATITO2(20)
14 cP  DIM TOTOTIP(18),Team$(18),TPUN(18),TABGEPU(18),TGE(18),TBT
       (18),TGP(18),STATITO(20),zu(19)
15 J9  DIM Tito1(14),TBD(18),DIFFERENZ(18),TP(18),TPLATZ(18),TABE
       PLA(18),GELBE(19),TORKO(19),AM$(18)
16 1U  DIM STOPPER(19),LOST(18),WIN(18),EQUAL(18),L(18),k(18),H(1
       8)
17 KK  GOSUB U1
18 zU  Anfang:
19 Uz  REM Farben definieren
20 ki  GOSUB Farben
21 O8  CLS:GOSUB MITTIG:LOCATE 14,8:ver=0:neg=0
22 LJ  INPUT"Bitte geben Sie Ihren Namen ein: ",ma$
```

Neben dem Ziel, die Meisterschaft und den Titel im UEFA-Cup zu ergattern, ist es Ihre Aufgabe, viele Managerpunkte zu sammeln.

Diese Managerpunkte werden durch verschiedene Aktionen erzielt. Sie errechnen sich wie folgt:

Abrechnung am Saisonende

Der Kauf eines Spielers bringt eine seinem Faktor entsprechende Punktzahl. Ein Beispiel: Faktor 6 eines Spielers bringt sechs Managerpunkte.

Ein gewonnenes Spiel ergibt zwei Punkte, ein Unentschieden einen Punkt.

Die Faktorerhöhung eines Spielers durch vier geschossene Tore bringt einen Punkt, ein erfolgreicher Tototip (mehr als fünf Richtige) deren drei.

Weiterhin wird am Saisonende eine Abrechnung erstellt (Bild 13). Zu den bereits erzielten Punkten werden zusätzliche für das erwirtschaftete Kapital, für den Stadionausbau, die Plazierung in der Tabelle und das Erreichen der UEFA-Cup-Runden addiert. So können Sie stets den erreichten Stand notieren. Im nächsten Spiel geht es dann daran, den bisher erreichten Punktestand zu übertreffen.

Bei der Version auf der Programmservice-Diskette wird die Gesamtpunktzahl mit Ihrem Namen und dem erreichten Tabellenplatz auf Diskette gespeichert — allerdings nur, wenn Sie sich unter den zehn besten Managern plazieren konnten.

Wir wünschen Ihnen viele spannende Bundesliga-Runden, prickelnde UEFA-Cup-Spiele und eine glückliche Hand bei allen Entscheidungen. Und denken Sie bei Niederlagen an das »Wunder von Mailand«, geben Sie niemals auf. *(rs)*

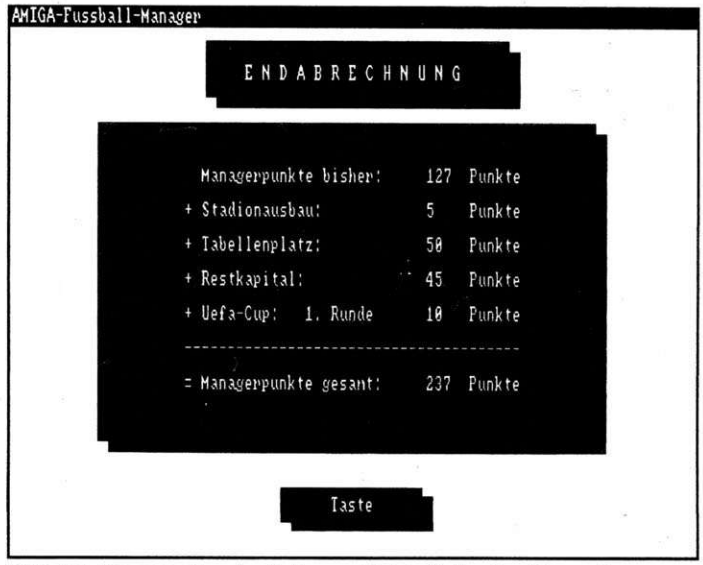


Bild 13. Abgerechnet wird zum Schluß: In der Bewertung sind alle Faktoren eines guten Managements berücksichtigt

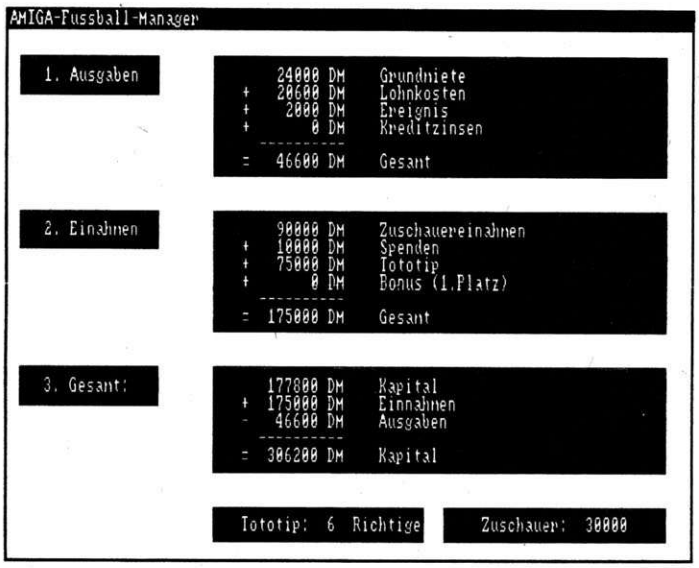


Bild 12. Die Abrechnung aller Kosten und Einnahmen nach einem Spieltag. Der Tototip hat sich gelohnt.

Version auf der Programmservice-Diskette

Das Listing auf den folgenden Seiten ist eine leicht verkürzte Version des Programms »Anpfiff« von Thorsten Froese. In der Originalversion, die Sie auf der Programmservice-Diskette finden, sind einige zusätzliche Features enthalten. Der UEFA-Cup, dessen Runden in Hin- und Rückspielen entschieden werden, ist einer der Zusätze. Außerdem können Sie Namen für die Bundesliga-Teams, die Spielernamen und die Bezeichnungen der Teams im UEFA-Cup festlegen. Diese werden dann auf Diskette gespeichert.

Ebenso sind auf Diskette drei Bilder gespeichert, ein Stadionbild, ein Bild für die erreichte Meisterschaft und eines für den Gewinn des UEFA-Cups.

Der Umfang des Programms mit den zusätzlichen Bildern wäre für den Abdruck im Heft leider erheblich zu groß. Wir haben uns für diesen Kompromiß entschieden, damit alle Leser voll auf ihre Kosten kommen.

```
23 8Q  COLOR 1,0
24 o2  REM FOR i = 1 TO 18
25 xQ  REM  AM$(i)=e$(i)
26 VI  REM NEXT i
27 eB  FOR i = 1 TO 18
28 PF2   zu=100:GOSUB zufallszahl
29 DO0 E(i)=z:NEXT i
30 hE  FOR i = 1 TO 18
31 oH2   FOR j = 1 TO 18
32 aK4     IF E(j) < E(i) THEN SWAP E(i),E(j): SWAP E$(i),E$(j)
33 OH2   NEXT j
34 zF0 NEXT i
35 hE  CLS:GOSUB USCH:LOCATE 3,30
36 64  LINE (190,48)-(470,199),2,bf
37 QG  LINE (180,43)-(460,194),1,bf
38 GX  PRINT "T E A M A U S W A H L"
39 qN  FOR i = 1 TO 18
40 b52   LOCATE 6+i,27
41 Go    PRINT i; TAB(33)E$(i)
42 7N0 NEXT i
43 zG  LINE (230,214)-(430,231),2,bf
44 Ok  LINE (220,209)-(420,226),1,bf
45 Vs  LOCATE 28,36
46 4g  INPUT"IHRE WAHL: ",T:COLOR 1,0
47 EY  IF T > 18 GOTO Anfang
48 Q5  IF T < 1 GOTO Anfang
49 D5  en$(1) = E$(1):E$(1)=E$(T):E$(T)=en$(1)
50 1Y  FOR i = 1 TO 18
```

```
51 EE2   TE$(i)=E$(i)
52 HT    MN$(i)=TE$(i)
53 IY0 NEXT i
54 01  Weitereins:
55 uZ  FOR i = 1 TO 16
56 qb  ko(i)=100:zu$(i)="OK"
57 Mc  NEXT i
58 ZJ  ka=70000&:va=50000&:sp=1:bm=1:bms=1
59 km  z=0:zu = 30:RANDOMIZE TIMER
60 Ef  z=2+INT(RND*(zu)+.5)
61 iP  ROTKAR=z
62 31  GOTO MANAGERMARKT
63 Uc  mann:
64 Nj  Haupt:
65 z0  CLS:COLOR 0,1
66 73  LINE (9,9)-(621,20),2,bf
67 HH  LINE (3,4)-(615,16),1,bf:LOCATE 2,2
68 6h  PRINT"Manschaftskonstellation:    ";TE$(1);:PRINT TAB(61)"
       Spieltag: ";sp
69 T6  PRINT
70 tw  GOSUB Mannschaftaufbau
```

Listing 1. »Anpfiff« ist eine Sportsimulation der Extraklasse. Erleben Sie die Fußball-Bundesliga aus der Sicht des Vereins-Managers. Bitte Installationshinweise beachten.

```
 71 V8  PRINT
 72 3j  Hau2:
 73 KO  pa=0:pb=0:pc=0:pd=0:pd=0:pe=0:pf=0:pg=0:ph=0:kg=0
 74 ji  FOR i = 1 TO 11
 75 jO  IF po$(i)="Torwart" AND zu$(i)="OK" THEN pa=pa+fa(i):kg=kg
        +ko(i)
 76 bj  IF po$(i)="Abwehr" AND zu$(i)="OK" THEN pb=pb+fa(i):kg=kg+
        ko(i)
 77 d0  IF po$(i)="Mittelfeld" AND zu$(i)="OK" THEN pc=pc+fa(i):kg
        =kg+ko(i)
 78 Um  IF po$(i)="Angriff" AND zu$(i)="OK" THEN pd=pd+fa(i):kg=kg
        +ko(i)
 79 iy  NEXT i
 80 Tw  pg=pa+pb:ph=pc+pd
 81 xu  ka(1)=pg:ks(1)=ph:kn(1)=kg
 82 rZ  IF SPPA1=1 THEN SPPA1=0:GOTO Hau3
 83 Cu  COLOR 0,1:SPIELFREI=1
 84 8k  LINE (9,226)-(153,236),2,bf
 85 2X  LINE (3,223)-(147,232),5,bf
 86 hC  LINE (247,226)-(401,236),2,bf
 87 hD  LINE (241,223)-(395,232),4,bf
 88 uU  LINE (482,226)-(621,236),2,bf
 89 1b  LINE (476,223)-(615,232),3,bf
 90 zL  LOCATE 29,2:COLOR 2,5:PRINT"Abwehrstärke:";pg
 91 8e  LOCATE 29,32:COLOR 2,4:PRINT"Angriffstärke:";ph
 92 8h  LOCATE 29,61:COLOR 2,3:PRINT"Kondition:";kg
 93 wx  LOCATE 27,27:COLOR 0,1
 94 er  LINE (205,210)-(430,220),2,bf
 95 U6  LINE (199,207)-(424,216),1,bf
 96 CE  PRINT" Wollen Sie wechseln (J/N)";Bes$
 97 Kc  COLOR 1,0
 98 fy  SPPA=1
 99 of  JaNein3:
100 Pq  ver=3:neg=3
101 xr  GOSUB Abfrageschleife
102 rH  GOTO JaNein3
103 vd  Noger3:
104 Yn  ver=0:neg=0:MITTELAB=0
105 ED  FOR i = 1 TO 11
106 GL  IF po$(i)="Mittelfeld" AND zu$(i)="OK" THEN MITTELAB=MITTE
        LAB+1
107 gk  IF MITTELAB=2 THEN GOTO MANAGERMARKT
108 BR  NEXT i
109 2y  FOR i = 12 TO 18
110 Ji  IF po$(i)="Mittelfeld" AND zu$(i)="OK" THEN
111 Ik3    CLS:GOSUB MITTIG:LOCATE 14,9
112 xI     PRINT"Sie muessen mindestens zwei Mittelfeldspieler auf
           bieten":GOSUB warten
113 Ho     COLOR 1,0:GOTO Haupt
114 mf0 END IF
115 IY  NEXT i
116 vt  GOTO MANAGERMARKT
117 H3  Roger3:
118 e3  CLS:ma=1:GOSUB Mannschaftaufbau:ver=0:neg=0
119 pA  LINE (64,213)-(164,224),4,bf:LOCATE 28,10:COLOR 2,4:PRINT"
        AUSWECHSELN"
120 Zh  LINE (450,213)-(550,224),1,bf:LOCATE 28,58:COLOR 0,1:PRINT
        "EINWECHSELN"
121 zR  LINE (213,207)-(255,220),7,bf
122 G0  LINE (357,207)-(399,220),7,bf
123 d6  COLOR 2,7:LOCATE 27,29:INPUT"",sr
124 q0  IF sr <1 OR sr >18 OR sr >11 THEN BEEP:GOTO Roger3
125 L0  LINE (64,213) - (164,226),1,bf
126 8t  LOCATE 28,10:COLOR 0,1:PRINT"AUSWECHSELN"
127 wk  LINE (450,213) - (550,224),5,bf
128 pB  LOCATE 28,58:COLOR 2,5:PRINT"EINWECHSELN"
129 5S  LOCATE 27,47:COLOR 2,7:INPUT"",ss
130 n0  IF ss >18 OR ss <1 OR ss <12 THEN BEEP:GOTO Roger3
131 13  IF po$(sr)="Torwart" AND po$(ss)<>"Torwart" THEN BEEP:GO
        TO Roger3
132 fe  FOR i = 1 TO 11
133 Bz  IF po$(sr)<>"Torwart" AND po$(ss)="Torwart" THEN BEEP:GO
        TO Roger3
134 br  NEXT i
135 1f  na$(19)=na$(sr):fa(19)=fa(sr):po$(19)=po$(sr):ko(19)=ko(sr
        ):zu$(19)=zu$(sr):GELBE(19)=GELBE(sr):TORKO(19)=TORKO(sr):S
        TOPPER(19)=STOPPER(sr):IARSCH(19)=IARSCH(sr):zu(19)=zu(sr)
136 cF  na$(sr)=na$(ss):fa(sr)=fa(ss):po$(sr)=po$(ss):ko(sr)=ko(ss
        ):zu$(sr)=zu$(ss):GELBE(sr)=GELBE(ss):TORKO(sr)=TORKO(ss):S
        TOPPER(sr)=STOPPER(ss):IARSCH(sr)=IARSCH(ss):zu(sr)=zu(ss)
137 lE  na$(ss)=na$(19):fa(ss)=fa(19):po$(ss)=po$(19):ko(ss)=ko(19
        ):zu$(ss)=zu$(19):GELBE(ss)=GELBE(19):TORKO(ss)=TORKO(19):S
        TOPPER(ss)=STOPPER(19):IARSCH(ss)=IARSCH(19):zu(ss)=zu(19)
```

```
138 zH  COLOR 1,0
139 QL  GOTO Haupt
140 AJ  Mannschaftaufbau:
141 Bq  IF ma=1 THEN
142 EF3    CLS:COLOR 0,1
143 j6     LOCATE 2,33
144 HY     LINE (205,6)-(425,20),2,bf
145 pV     LINE (200,1)-(420,15),1,bf
146 bq     PRINT"W E C H S E L"
147 JC0 END IF
148 BT  COLOR 0,1
149 tA  LINE (9,28)-(621,204),2,bf
150 9o  LINE (3,23)-(615,200),1,bf:LOCATE 4,2
151 hD  PRINT"Spielername und Nummer:"TAB(30)"Faktor:"TAB(39)"Kond
        ition:"TAB(51)"Zustand:"TAB(61)"Position:":PRINT
152 zy  FOR i = 1 TO 11
153 sy  LOCATE 5+i,2
154 32  PRINT na$(i) TAB(17)"("i")"TAB(31) fa(i) TAB(40) ko(i) TAB
        (51) zu$(i) TAB(61) po$(i)
155 wC  NEXT i
156 87  LOCATE 18,2
157 CG  PRINT"Auswechselspieler:"
158 pl  FOR i = 12 TO 18
159 fJ  IF na$(i)="" THEN ko(i)=0
160 7F  LOCATE 7+i,2
161 A9  PRINT na$(i) TAB(17)"("i")"TAB(31) fa(i) TAB(40) ko(i) TAB
        (51) zu$(i) TAB(61) po$(i)
162 b5  NEXT i:COLOR 1,0
163 Ha  IF ma=0 THEN RETURN
164 Rj  COLOR 0,1
165 KL  LINE (266,207) - (346,235),5,bf
166 p0  LINE (266,207) - (306,235),7,b
167 tj  LINE (306,207) - (346,235),7,b
168 bP  LINE (266,212) - (280,230),7,b
169 27  LINE (332,212) - (346,230),7,b
170 p7  CIRCLE (306,221),12,7
171 OR  LINE (212,206) - (256,221),7,bf
172 pf  LINE (212,206) - (256,221),2,b
173 eZ  LINE (233,221) - (235,235),7,bf
174 YO  LINE (356,206) - (400,221),7,bf
175 zE  LINE (356,206) - (400,221),2,b
176 vo  LINE (377,221) - (379,235),7,bf
177 W5  LINE (69,218) - (169,229),2,bf
178 wc  LINE (455,218) - (555,229),2,bf
179 gh  ma=0
180 Gs  RETURN
181 RT  Zufaelligkeit:
182 oq  kd=0
183 Yk  IF sp<=2 THEN RETURN
184 LL  IF ka<=0 THEN RETURN
185 76  zu=3:GOSUB zufallszahl
186 nE  ON z GOTO Pech,Glueck,Pech
187 YO  Pech:
188 j5  IF ka <=100000& THEN zu=10:GOTO Ermit
189 7O  IF ka >=200000& THEN zu=30:GOTO Ermit
190 Uf  IF ka >=400000& THEN zu=50:GOTO Ermit
191 4P  Ermit:
192 IM  GOSUB zufallszahl
193 ZC  kd=z*1000
194 jB  IF kd>ka THEN kd=ka
195 V7  RETURN
196 Jx  Glueck:
197 X9  RETURN
198 Fe  MANAGERMARKT:
199 9A  CLS:COLOR 0,1
200 xC  neg=0:ver=0:x=0:FOR i = 1 TO 5
201 MK  LINE (20,15+x) - (310,35+x),2,bf
202 aQ  LINE (10,10+x) - (300,30+x),1,bf
203 pA  LINE (330,15+x) - (620,35+x),2,bf
204 vX  LINE (320,10+x) - (610,30+x),1,bf
205 X5  x=x+32
206 11  NEXT i
207 0z  LINE (20,180) - (620,237),2,bf
208 U8  LINE (10,175) -(610,232),1,bf
209 R4  LOCATE 3,8:PRINT"M A N A G E R M A R K T"
210 f8  LOCATE 3,44:PRINT"-1-  TOTOTIP"
211 qa  LOCATE 7,5:PRINT"-2-  TRANSFERMARKT"
212 XC  LOCATE 7,44:PRINT"-3-  BAUMASSNAHMEN"
213 XU  LOCATE 11,5:PRINT"-4-  TABELLE ZEIGEN"
214 bD  LOCATE 11,44:PRINT"-5-  SPIELERSTATUS ZEIGEN"
215 Cb  LOCATE 15,5:PRINT"-6-  NAECHSTEN GEGNER ZEIGEN"
216 VE  LOCATE 15,44:PRINT"-7-  MANNSCHAFT AUFSTELLEN"
217 j1  LOCATE 19,5:PRINT"-8-  NAECHSTEN SPIELTAG ZEIGEN"
```

```
218 QW  LOCATE 19,44:PRINT"-9- NAECHSTEN SPIELTAG SPIELEN"
219 1U  LOCATE 24,5:PRINT"KAPITAL:        ";ka;:PRINT"DM"
220 ja  LOCATE 24,44:PRINT"SPIELTAG:    ";sp
221 G8  LOCATE 26,5:PRINT"STADIONPLATZE: ";va
222 9J  LOCATE 26,44:PRINT"MANAGERNAME: ";ma$
223 xN  LOCATE 28,5:PRINT"MANAGERFAKTOR: ";mf
224 Iv  LOCATE 28,44:PRINT"IHRE WAHL:"
225 JS  LOCATE 28,57:INPUT"",a
226 Ph  COLOR 1,0
227 rc  IF a > 9 THEN GOTO MANAGERMARKT
228 It  IF a < 1 THEN GOTO MANAGERMARKT
229 pm  ON a GOTO TOTO,Transfer,Bau,Tabell,SPstatus,Gegner,mann,Ta
        g,Spielen
230 rx  Transfer:
231 Nt  CLS:
232 gD  GOSUB USCH
233 2B  x=0
234 5R  FOR i=1 TO 5
235 G2  LINE (230,62+x) - (430,82+x),2,bf
236 hE  LINE (220,57+x) - (420,77+x),1,bf
237 3b  x=x+32
238 HX  NEXT i
239 7f  LOCATE 3,28:PRINT"T R A N S F E R M A R K T"
240 2Y  LOCATE 9,31:PRINT"-1- Spieler kaufen"
241 Dx  LOCATE 13,31:PRINT"-2- Spieler verkaufen"
242 M1  LOCATE 17,31:PRINT"-3- Spieler verleihen"
243 iM  LOCATE 21,31:PRINT"-4- Zurueck"
244 OC  LOCATE 25,31:PRINT"Ihre Wahl:";:INPUT"",a
245 Za  IF a>4 GOTO Transfer
246 IO  IF a<1 GOTO Transfer
247 k2  COLOR 1,0
248 V6  ON a GOTO Spielerkauf,Spielerverkauf,Spielerverleih,MANAGE
        RMARKT
249 BR  Spielerkauf:
250 kr  IF kaufzaehler>=4 THEN
251 pr3    CLS:GOSUB MITTIG
252 SQ     LOCATE 14,5
253 63     PRINT"Tut mir Leid, aber sie koennen nur 4 Auslaender i
           n Ihr Team aufnehmen"
254 r9     COLOR 1,0
255 zT     GOSUB warten:GOTO MANAGERMARKT
256 4x0 END IF
257 uW  IF KaufSp=0 THEN
258 wy3    CLS:GOSUB MITTIG
259 FQ     LOCATE 14,20
260 Yi     PRINT"Zur Zeit koennen Sie keinen Spieler kaufen !"
261 yG     COLOR 1,0
262 6a     GOSUB warten:GOTO MANAGERMARKT
263 B40 END IF
264 T0  FOR i = 1 TO 18
265 Mr  IF na$(i)="" THEN GOTO Spielerkauf1
266 jz  NEXT i
267 rx  CLS
268 y0  GOSUB MITTIG
269 Ym  LOCATE 14,23
270 zL  PRINT"Ihr Team besteht bereits aus 18 Spielern !":COLOR 1,
        0:GOSUB warten:GOTO MANAGERMARKT
271 mJ  Spielerkauf1:
272 cU  CLS:GOSUB USCH
273 lz  LOCATE 3,35:PRINT"K A U F E N"
274 s6  LINE (20,60) - (620,70),2,bf
275 xQ  LINE (10,55) - (610,65),1,bf
276 yW  LOCATE 8,10
277 Ef  PRINT"Name:" TAB(23) "Faktor:" TAB(34) "Position:" TAB(49)
         "Kaufkosten:"
278 xc  Allright=sp
279 7Q  IF Allright>13 THEN Allright=13
280 o4  Liva=Allright*9
281 WH  LINE (20,81) - (620,90+Liva),2,bf
282 nL  LINE (10,76) - (610,85+Liva-5),1,bf
283 de  FOR i = 1 TO Allright
284 4A  GOSUB Verkaufsauswertung
285 BW  LOCATE 10+i,5
286 Ou  PRINT USING"# # ";i;:PRINT TAB(10) Vna$(i) TAB(27) Vfa(i)
        TAB(39) Vpo$(i) TAB(53) Kaufko(i)
287 4K  NEXT i
288 rQ  LINE (20,227) - (235,237),2,bf
289 3P  LINE (10,222) - (225,232),1,bf
290 PW  LINE (285,227) - (615,237),2,bf
291 Om  LINE (275,222) - (605,232),1,bf
292 Ew  LOCATE 29,5:PRINT"Kapital: ";ka;:PRINT"DM"
293 GG  LOCATE 29,37:PRINT"Welchen Spieler wollen Sie kaufen ?"
294 NV  LOCATE 29,73:INPUT"",a
295 Wo  COLOR 1,0
296 3L  IF a < 1 THEN GOTO Spielerkauf1
297 V9  IF a > Allright THEN GOTO Spielerkauf1
298 BE  IF Kaufko(a)>ka THEN
299 NT2    CLS
300 UW     GOSUB MITTIG
301 JJ     LOCATE 14,7
302 1z     PRINT"Sie haben nicht genug Kapital,um den Spieler kaufe
           n zu koennen !"
303 LM     COLOR 1,0:GOSUB warten:GOTO Transfer
304 qJ0 END IF
305 8f  FOR i = 1 TO 18
306 Qh  IF na$(i)="" THEN GOTO Spuebergeben
307 Oe  NEXT i
308 9Y  GOSUB warten:GOTO Transfer
309 Lu  Spuebergeben:
310 MY  na$(i)=Vna$(a):po$(i)=Vpo$(a):fa(i)=Vfa(a):zu$(i)="OK":ko(
        i)=100:STOPPER(i)=0
311 OZ  FOR i = 2 TO 6
312 Qn  IF Vfa(a)=i THEN mf=mf+i
313 Uk  NEXT i
314 jN  ka=ka-Kaufko(a):Vna$(a)="Verkauft":Vpo$(a)="":Vfa(a)=0:Kau
        fko(a)=0:KaufSp=0
315 a4  CLS:GOSUB MITTIG:LOCATE 14,8:kaufzaehler=kaufzaehler+1
316 dt  PRINT"Sie haben nun den ";:PRINT USING"#.";kaufzaehler;:P
        RINT" von 4 moeglichen Auslaendern in Ihrem Team !"
317 lp  GOSUB warten:COLOR 1,0:GOTO MANAGERMARKT
318 4k  Verkaufsauswertung:
319 L1  IF Vfa(i) = 2 THEN Kaufko(i)=80000&
320 OH  IF Vfa(i) = 3 THEN Kaufko(i)=200000&
321 BP  IF Vfa(i) = 4 THEN Kaufko(i)=300000&
322 Wm  IF Vfa(i) = 5 THEN Kaufko(i)=450000&
323 dh  IF Vfa(i) = 6 THEN Kaufko(i)=600000&
324 aC  RETURN
325 q4  Spielerverkauf:
326 wj  IF VerkaufSp=0 THEN
327 353    CLS:GOSUB MITTIG
328 MX     LOCATE 14,20
329 OM     PRINT"Zur Zeit koennen Sie keinen Spieler verkaufen !":
           COLOR 1,0
330 Cg     GOSUB warten:GOTO MANAGERMARKT
331 HA0 END IF
332 aS  CLS:GOSUB USCH
333 pC  LOCATE 3,32
334 wH  PRINT"V E R K A U F E N"
335 Ca  LINE (18,50) - (620,60),2,bf
336 1L  LINE (8,45) - (610,55),1,bf
337 lm  LOCATE 7,5
338 Kj  PRINT"Name" TAB(21) "F" TAB(27) "Position" TAB(42) "Kon" T
        AB(55) "Preis" TAB(69) "Lohn"
339 79  GOSUB MITTIG
340 Vr  LOCATE 14,31:PRINT"EINEN MOMENT BITTE..."
341 kj  GOSUB LWBerechnung
342 Xd  LINE (18,71) - (620,236),2,bf
343 tD  LINE (8,66) - (610,231),1,bf
344 1I  FOR i = 1 TO 18
345 EO  LOCATE 9+i,2
346 x8  PRINT USING"# # ";i;:PRINT TAB(5) na$(i) TAB(20) fa(i) TAB
        (27) po$(i) TAB(41) ko(i) TAB(54) Verkaufs(i) TAB(68) Lohnm
        o(i)
347 2I  NEXT i
348 S9  PRINT:COLOR 2,1:PRINT TAB(2):PRINT"Kapital:";ka;"DM";:PRIN
        T TAB(28):INPUT"Welchen Spieler moechten Sie verkaufen ?",a
349 Og  COLOR 1,0
350 8Z  IF a < 1 THEN GOTO Spielerverkauf
351 Xi  IF a > 18 THEN GOTO Spielerverkauf
352 tQ  FOR i = 1 TO 18
353 9Z  IF po$(i)="Mittelfeld" THEN Pruefsumme=Pruefsumme+1
354 Dm  IF po$(i)="Torwart" THEN ABCHECK=ABCHECK+1
355 AQ  NEXT i
356 Kn  IF po$(a)="Mittelfeld" AND Pruefsumme <=2 THEN
357 4T3    CLS:GOSUB MITTIG:LOCATE 14,6
358 ML     PRINT"Tut mir Leid, aber Sie muessen mindestens 2 Mitte
           lfeldspieler haben !":GOSUB warten:COLOR 1,0:Pruefsumme=
           0:GOTO Spielerverkauf
359 jc0 END IF
360 10  IF po$(a)="Torwart" AND ABCHECK <=2 THEN
361 Km3    CLS:GOSUB MITTIG:LOCATE 14,9
362 a9     PRINT"Tut mir Leid, aber Sie muessen mindestens 2 Torhu
           eter haben !":COLOR 1,0:GOSUB warten:ABCHECK=0:GOTO Spie
           lerverkauf
```

Listing 1. »Anpfiff« (Fortsetzung)

```
363 ng0 END IF
364 Zg  IF zu$(a)="verliehen" THEN
365 P23     CLS:GOSUB MITTIG:LOCATE 14,13
366 yX      PRINT"Sie koennen keinen verliehenen Spieler verkaufen"
            :COLOR 1,0:GOSUB warten:GOTO Spielerverkauf
367 rk0 END IF
368 gM  CLS:GOSUB MITTIG:LOCATE 14,25
369 gW  PRINT"Moechten Sie wirklich verkaufen";Bes$
370 j1  COLOR 1,0
371 IB  JaNein5:
372 30  ver=5:neg=5
373 LF  GOSUB Abfrageschleife
374 Ln  GOTO JaNein5
375 VJ  Roger5:
376 mM  neg=0:ver=0
377 of  FOR i = 1 TO 13
378 B1  IF Vna$(i)="Verkauft" THEN Vna$(i)=na$(a):Vpo$(i)=po$(a):V
        fa(i)=fa(a):GOTO nureiner
379 Yo  NEXT i
380 v0  nureiner:
381 7X  na$(a)="":po$(a)="":fa(a)=0:zu$(a)="":ko(a)=0:TORKO(a)=0:G
        ELBE(a)=0
382 xU  ka=ka+Verkaufs(a):VerkaufSp=0:Verkaufs(a)=0:Lohnmo(a)=0
383 At  CLS:GOSUB MITTIG:LOCATE 14,28
384 Gx  PRINT"OK, Spieler ist verkauft":COLOR 1,0:GOSUB warten:GOT
        O MANAGERMARKT
385 QZ  Spielerverleih:
386 Ww  Zzahl=0
387 S0  IF VerleihSp=0 THEN
388 6n3     CLS:GOSUB MITTIG:LOCATE 14,17
389 Bp      PRINT"Zur Zeit koennen Sie keinen Spieler verleihen"
390 RX      COLOR 1,0:GOSUB warten:GOTO MANAGERMARKT
391 F80 END IF
392 X4  FOR i = 1 TO 18
393 ip  IF na$(i)="" THEN Zzahl=i:GOTO Spverleih2
394 n3  NEXT i
395 p5  Spverleih2:
396 Y5  IF Zzahl=0 THEN Zzahl=18
397 Fj  zu=Zzahl:GOSUB zufallszahl
398 NN  IF na$(z)="" THEN GOTO Spverleih2
399 fF  IF zu$(z)="OK" THEN AB=z:GOTO Spverleih3
400 kZ  GOTO Spverleih2
401 1D  Spverleih3:
402 eJ  CLS:GOSUB USCH:LOCATE 3,24
403 US  PRINT"S P I E L E R   V E R L E I H E N"
404 W2  LINE (230,62) - (430,82),2,bf
405 qR  LINE (220,57) - (420,77),1,bf
406 Qu  LOCATE 9,33
407 RC  PRINT"Kapital:";ka;:PRINT"DM":zu=17
408 dZ  Teauslo:
409 kE  GOSUB Teauslosung
410 SN  IF TE$(z)=TE$(1) THEN GOTO Teauslo
411 rJ  LINE (92,109)-(540,195),2,bf
412 Gb  LINE (83,104)-(530,190),1,bf
413 yD  LOCATE 15,14
414 hr  LINE (92,217)-(540,232),2,bf
415 Yk  LINE (83,212)-(530,227),1,bf
416 JP  PRINT"T E L E X :":LOCATE 17,22
417 sr  PRINT TE$(z):LOCATE 19,30
418 lr  PRINT"bietet fuer den Spieler"
419 x9  LOCATE 21,38:PRINT na$(AB)
420 7P  IF fa(AB)=2 THEN Verleihko=60000&
421 Mi  IF fa(AB)=3 THEN Verleihko=90000&
422 Go  IF fa(AB)=4 THEN Verleihko=130000&
423 iM  IF fa(AB)=5 THEN Verleihko=180000&
424 Lv  IF fa(AB)=6 THEN Verleihko=220000&
425 Ty  LOCATE 23,46:PRINT Verleihko;:PRINT"DM"
426 i6  LOCATE 28,28
427 rG  PRINT"Nehmen Sie dieses Angebot an ?";Bes$
428 fx  COLOR 1,0
429 HB  JaNein6:
430 7P  ver=6:neg=6
431 HB  GOSUB Abfrageschleife
432 Kn  GOTO JaNein6
433 TI  Roger6:
434 Rh  neg=0:ver=0:zu$(AB)="verliehen":ka=ka+Verleihko:AB=0:Verle
        ihSp=0
435 cG  CLS:GOSUB MITTIG:LOCATE 14,14
436 FN  PRINT"O.K., der Spieler wird fuer einen Spieltag verliehen
        "
437 CI  COLOR 1,0:GOSUB warten:GOTO MANAGERMARKT
438 kV  Teauslosung:
439 aI  z=0:RANDOMIZE TIMER
440 Gl  z=2+INT(RND*(zu)):RETURN
441 jF  TOTO:
442 7n  IF TOTOTIPPEN=1 THEN
443 8r3     CLS:GOSUB MITTIG:LOCATE 14,28
444 AG      PRINT"Sie haben bereits getippt":COLOR 1,0:GOSUB warten
            :GOTO MANAGERMARKT
445 700 END IF
446 Nl  IF ka < 10000 THEN
447 Y93     CLS:GOSUB MITTIG:LOCATE 14,20
448 d8      PRINT"Sie haben nicht genuegend Geld zum Tippen":COLOR
            1,0:GOSUB warten:GOTO MANAGERMARKT
449 B40 END IF
450 W7  CLS:GOSUB USCH:LOCATE 3,34
451 nb  PRINT"T O T O T I P"
452 YQ  LINE (20,134) - (310,154),2,bf
453 wt  LINE (10,129) - (300,149),1,bf
454 o3  LINE (20,60) - (620,80),2,bf
455 tN  LINE (10,55) - (610,75),1,bf
456 ip  LOCATE 9,4
457 u0  PRINT"Mannschaft:" TAB(25) "Platz:" TAB(35) "Mannschaft:"
        TAB(60) "Platz:" TAB(70) "Tip:"
458 45  LOCATE 18,4
459 Hd  PRINT"Kapital:";ka;:PRINT"DM"
460 ru  FOR i = 1 TO 17 STEP 2
461 kD  FOR j = 1 TO 18
462 Uj  IF Team$(j)=TE$(a(i)) THEN Plaz1=j
463 10  IF Team$(j)=TE$(a(i+1)) THEN Plaz2=j
464 xE  NEXT j
465 Bz  AGAIN:
466 ug  GOSUB MITTIG:LOCATE 14,4
467 N4  PRINT TE$(a(i)) TAB(25) "(";:PRINT USING"##.";Plaz1;:PRI
        NT")" TAB(33)"-"TAB(35) TE$(a(i+1)) TAB(60):PRINT"(";:PRINT
         USING"##.";Plaz2;:PRINT")" TAB(70): INPUT"",TOTOTIP(a(i)
        )
468 55  IF TOTOTIP(a(i))>2 THEN BEEP:GOTO AGAIN
469 so  IF TOTOTIP(a(i))<0 THEN BEEP:GOTO AGAIN
470 1H  NEXT i
471 FD  TOTOFR:
472 Xb  LINE (20,174) - (620,194),2,bf
473 v4  LINE (10,169) - (610,189),1,bf
474 zw  LOCATE 23,4
475 gz  INPUT"Ihr Einsatz (in Tausend);  >10.000 und < 25.000 DM
        ",MON
476 aa  IF MON>25 OR MON<10 THEN BEEP:GOTO TOTOFR
477 qA  kapi=ka:MONETE=MON*1000:PRu=kapi-MONETE
478 63  IF PRu<0 THEN BEEP:GOTO TOTOFR
479 t7  ka=ka-MONETE
480 Vn  COLOR 1,0
481 fN  CLS:GOSUB MITTIG:LOCATE 14,27
482 Rt  PRINT"O.K. Ihr Einsatz wird notiert !":GOSUB warten:COLOR
        1,0:TOTOTIPPEN=1:GOTO MANAGERMARKT
483 Ik  Tag:
484 Ca  CLS:LOCATE 3,33:GOSUB USCH
485 1A  PRINT"NAECHSTER SPIELTAG"
486 78  LINE (47,48)-(596,199),2,bf
487 eS  LINE (37,43)-(586,194),1,bf
488 JM  FOR i = 1 TO 17 STEP 2
489 Cf  FOR j = 1 TO 18
490 wB  IF Team$(j)=TE$(a(i)) THEN Plaz1=j
491 Tq  IF Team$(j)=TE$(a(i+1)) THEN Plaz2=j
492 Pg  NEXT j
493 Tq  LOCATE 6+i,11
494 Xn  PRINT TE$(a(i)) TAB(33) "(";:PRINT USING"##.";Plaz1;:PRI
        NT")" TAB(39)"-"TAB(41) TE$(a(i+1)) TAB(65):PRINT"(";:PRINT
         USING"##.";Plaz2;:PRINT")"
495 Qg  NEXT i
496 IZ  LINE (230,214)-(430,231),2,bf
497 h3  LINE (220,209)-(420,226),1,bf
498 uJ  LOCATE 28,38
499 kp  PRINT"Taste":COLOR 1,0:GOSUB warten:GOTO MANAGERMARKT
500 7Y  Gegner:
501 Hx  CLS:GOSUB USCH:LOCATE 3,25
502 4h  PRINT"N A E C H S T E R   G E G N E R "
503 7d  LINE (230,62) - (430,82),2,bf
504 R2  LINE (220,57) - (420,77),1,bf
505 7o  Hau3:
506 wE  IF SPPA=0 THEN SPPA=1:SPPA1=1:GOTO Hau2
507 9g  LOCATE 9,36
508 Uh  PRINT"Spieltag";sp
509 Zk  PLKJ:
510 su  GOSUB MITTIG
511 ZV  LOCATE 14,3
512 hk  FOR i = 1 TO 17 STEP 2
```

```
513 LS  IF TE$(a(i)) = TE$(1) THEN PRINT TAB(15) TE$(1) TAB(40)"-"
        TAB(47) TE$(a(i+1)):NF=a(i+1)
514 jI  IF TE$(a(i+1))= TE$(1) THEN PRINT TAB(15) TE$(a(i)) TAB(40
        )"-"TAB(47) TE$(1):NF=a(i)
515 kO  NEXT i
516 X4  FOR i = 1 TO 18
517 G8  IF Team$(i)=TE$(NF) THEN TPla=i
518 KO  IF Team$(i)=TE$(1) THEN TPla1=i
519 o4  NEXT i
520 8e  GOSUB GEGTAF
521 56  LOCATE 18,4
522 Jd  COLOR 2,1
523 yc  PRINT"GEGNER:";:PRINT TAB (12):PRINT USING"# # ";TPla;:PRI
        NT".";:PRINT TAB(24):COLOR 2,5:PRINT"ABWEHR:";ka(NF);:PRINT
         TAB(44):COLOR 2,4:PRINT"ANGRIFF:";ks(NF);:PRINT TAB(62):CO
         LOR 2,3
524 5U  PRINT"KONDITION";kn(NF)
525 Oa  LOCATE 22,4:COLOR 2,1
526 OD  PRINT"TEAM:";:PRINT TAB (12):PRINT USING"# # ";TPla1;:PRIN
        T".";:PRINT TAB(24):COLOR 2,5:PRINT"ABWEHR:";ka(1);:PRINT T
        AB(44):COLOR 2,4:PRINT"ANGRIFF:";ks(1);:PRINT TAB(62):COLOR
         2,3
527 fi  PRINT"KONDITION";kn( )
528 Jb  COLOR 0,1
529 Ou  WHILE sp>17
530 Fw  FOR i = 18 TO 34 STEP 2
531 50  LOCATE 26,34
532 yS  IF sp=i THEN PRINT"H I N R U N D E":LOCATE 28,38:PRINT STA
        TITO2(sp-17);:PRINT TAB(41):PRINT":";:PRINT TAB(42):PRINT S
        TATITO(sp-17)
533 2I  NEXT i
534 Nz  FOR i = 19 TO 33 STEP 2
535 XT  IF sp=i THEN PRINT"H I N R U N D E":LOCATE 28,38:PRINT STA
        TITO(sp-17);:PRINT TAB(41):PRINT":";:PRINT TAB(42):PRINT ST
        ATITO2(sp-17)
536 5L  NEXT i
537 4t  GOTO DOEDEL
538 7v  WEND
539 rw  LOCATE 27,37:PRINT"T A S T E"
540 hx  DOEDEL:
541 sN  NF=0:COLOR 1,0:GOSUB warten:GOTO MANAGERMARKT
542 Bm  GEGTAF:
543 81  LINE (20,134) - (140,154),2,bf
544 WU  LINE (10,129) - (130,149),1,bf
545 qv  LINE (20,167) - (140,187),2,bf
546 2u  LINE (10,162) - (130,182),1,bf
547 AG  LINE (170,134) - (300,154),2,bf
548 m2  LINE (160,129) - (290,149),5,bf
549 47  LINE (170,167) - (300,187),2,bf
550 Cv  LINE (160,162) - (290,182),5,bf
551 jQ  LINE (330,134) - (460,154),2,bf
552 Ot  LINE (320,129) - (450,149),4,bf
553 dH  LINE (330,167) - (460,187),2,bf
554 om  LINE (320,162) - (450,182),4,bf
555 po  LINE (230,202) - (430,231),2,bf
556 kd  LINE (220,197) - (420,226),1,bf
557 28  LINE (490,134) - (620,154),2,bf
558 dW  LINE (480,129) - (610,149),3,bf
559 wz  LINE (490,167) - (620,187),2,bf
560 3P  LINE (480,162) - (610,182),3,bf
561 P1  RETURN
562 61  LWBerechnung:
563 Ip  FOR i = 1 TO 18
564 O9  IF TORKO(i) >=0 AND fa(i)=0 THEN Lohnmo(i)=0:Verkaufs(i)=
        0
565 2z  IF TORKO(i) >=0 AND fa(i)=2 THEN Lohnmo(i)=1000:Verkaufs(
        i)=90000&
566 zy  IF TORKO(i) >=0 AND fa(i)=3 THEN Lohnmo(i)=1300:Verkaufs(
        i)=150000&
567 T8  IF TORKO(i) >=0 AND fa(i)=4 THEN Lohnmo(i)=1800:Verkaufs(
        i)=230000&
568 HA  IF TORKO(i) >=0 AND fa(i)=5 THEN Lohnmo(i)=2500:Verkaufs(
        i)=500000&
569 3K  IF TORKO(i) >=0 AND fa(i)=6 THEN Lohnmo(i)=4000:Verkaufs(
        i)=600000&
570 dt  NEXT i
571 Qx  FOR i = 1 TO 18
572 WH  IF TORKO(i) >=0 AND fa(i)=0 THEN Lohnmo(i)=0:Verkaufs(i)=
        0
573 Or  IF TORKO(i) >=2 AND fa(i)=2 THEN Lohnmo(i)=1200:Verkaufs(
        i)=120000&
574 U1  IF TORKO(i) >=2 AND fa(i)=3 THEN Lohnmo(i)=1700:Verkaufs(
        i)=200000&
575 H7  IF TORKO(i) >=2 AND fa(i)=4 THEN Lohnmo(i)=2300:Verkaufs(
        i)=300000&
576 EL  IF TORKO(i) >=2 AND fa(i)=5 THEN Lohnmo(i)=3000:Verkaufs(
        i)=600000&
577 tj  IF TORKO(i) >=2 AND fa(i)=6 THEN Lohnmo(i)=4500:Verkaufs(
        i)=700000&
578 l1  NEXT i
579 Y5  FOR i = 1 TO 18
580 eP  IF TORKO(i) >=0 AND fa(i)=0 THEN Lohnmo(i)=0:Verkaufs(i)=
        0
581 K9  IF TORKO(i) >=4 AND fa(i)=3 THEN Lohnmo(i)=2100:Verkaufs(
        i)=300000&
582 CX  IF TORKO(i) >=4 AND fa(i)=4 THEN Lohnmo(i)=2900:Verkaufs(
        i)=400000&
583 Ex  IF TORKO(i) >=4 AND fa(i)=5 THEN Lohnmo(i)=3500:Verkaufs(
        i)=550000&
584 rw  IF TORKO(i) >=4 AND fa(i)=6 THEN Lohnmo(i)=5000:Verkaufs(
        i)=900000&
585 B7  IF TORKO(i) >=4 AND fa(i)=7 THEN Lohnmo(i)=7500:Verkaufs(
        i)=1000000&
586 t9  NEXT i
587 gD  FOR i = 1 TO 18
588 mX  IF TORKO(i) >=0 AND fa(i)=0 THEN Lohnmo(i)=0:Verkaufs(i)=
        0
589 sj  IF TORKO(i) >=6 AND fa(i)=3 THEN Lohnmo(i)=3000:Verkaufs(
        i)=350000&
590 C9  IF TORKO(i) >=6 AND fa(i)=4 THEN Lohnmo(i)=4000:Verkaufs(
        i)=480000&
591 kO  IF TORKO(i) >=6 AND fa(i)=5 THEN Lohnmo(i)=5500:Verkaufs(
        i)=650000&
592 aQ  IF TORKO(i) >=6 AND fa(i)=6 THEN Lohnmo(i)=7500:Verkaufs(
        i)=1300000&
593 Qf  IF TORKO(i) >=6 AND fa(i)=7 THEN Lohnmo(i)=9000:Verkaufs(
        i)=1500000&
594 1H  NEXT i
595 oL  FOR i = 1 TO 18
596 PQ  IF zu$(i)="Spital" AND IARSCH(1)=1 THEN Verkaufs(i)=Verkau
        fs(i)/2:mf=mf-10
597 4K  NEXT i
598 Oc  RETURN
599 Ku  SPstatus:
600 An  CLS:GOSUB MITTIG:LOCATE 14,31
601 NR  PRINT"EINEN MOMENT BITTE...":GOSUB LWBerechnung:COLOR 1,0
602 Oj  CLS:GOSUB USCH:LOCATE 3,28
603 OA  PRINT"S P I E L E R S T A T U S"
604 Xv  LINE (18,50) - (620,60),2,bf
605 Mg  LINE (8,45) - (610,55),1,bf
606 u3  LOCATE 7,3
607 Yd  PRINT"Name:" TAB(15) "Gelbe:" TAB(25) "Tore:" TAB(35) "Fak
        tor:" TAB(45) "Lohnkosten:" TAB(60) "Verkaufswert:"
608 pv  LINE (18,71) - (620,236),2,bf
609 BV  LINE (8,66) - (610,231),1,bf
610 3a  FOR i = 1 TO 18
611 Zk  LOCATE 9+1,3
612 eC  PRINT na$(i) TAB(16) GELBE(i) TAB(26) TORKO(i) TAB(36) fa(
        i) TAB(46) Lohnmo(i) TAB(61) Verkaufs(i)
613 Ka  NEXT i
614 39  COLOR 1,0:GOSUB warten:GOTO MANAGERMARKT
615 dR  Tabell:
616 mZ  IF sp=1 THEN
617 It8         CLS:GOSUB MITTIG:LOCATE 14,20
618 6K          PRINT"Es hat noch kein Spieltag stattgefunden"
619 k2          COLOR 1,0
620 sM          GOSUB warten:GOTO MANAGERMARKT
621 xq0 END IF
622 OY  Tabaus=1:sp=sp-1:GOSUB TABELLE:sp=sp+1:Tabaus=0
623 64  GOTO MANAGERMARKT
624 tH  Bau:
625 Sa  IF sp=1 THEN GOTO Verw
626 jM  IF bmst=1 THEN
627 rXA           CLS:GOSUB MITTIG:LOCATE 14,25
628 Sb           PRINT"Im Stadion wird momentan noch ausgebaut"
629 uC           COLOR 1,0
630 2W           GOSUB warten:GOTO MANAGERMARKT
631 700 END IF
632 1h  IF ka<5000 THEN
633 U4A           CLS:GOSUB MITTIG:LOCATE 14,10
634 G2           PRINT"Sie bringen fuer einen Ausbau nicht genueg
                 end Kapital auf "
635 OI           COLOR 1,0
```

Listing 1. »Anpfiff« (Fortsetzung)

```
636 8c            GOSUB warten:GOTO MANAGERMARKT
637 D60 END IF
638 aj  IF bms=0 THEN GOTO RAUSWERFEN
639 Db  IF bm=1 THEN GOTO baum
640 jG  RAUSWERFEN:
641 i8  CLS:GOSUB MITTIG:LOCATE 14,7
642 iT  PRINT"Sie koennen an jedem Spieltag nur einmal den Baumark
        t besuchen"
643 Wc  COLOR 1,0:GOSUB warten:GOTO MANAGERMARKT
644 sv  baum:
645 cE  IF va>=100000& THEN
646 1f2   CLS:GOSUB MITTIG:LOCATE 14,14
647 FZ    PRINT"Ihr Stadion kann nicht mehr weiter ausgebaut werde
          n"
648 bh    COLOR 1,0:GOSUB warten:GOTO MANAGERMARKT
649 PI0 END IF
650 hh  zu=4:GOSUB zufallszahl
651 lA  ON z GOTO Bam,Verw,Bam,Bam
652 p5  Bam:
653 pY  CLS:GOSUB USCH:LOCATE 3,28
654 tW  PRINT"B A U M A S S N A H M E N"
655 8t  LINE (82,48)-(540,195),2,bf
656 jI  LINE (73,43)-(530,190),1,bf
657 Jw  LOCATE 9,14
658 aj  LINE (82,217)-(540,232),2,bf
659 Rc  LINE (73,212)-(530,227),1,bf
660 pn  PRINT"Kapital:                    ";ka"DM"
661 gX  LOCATE 12,14:PRINT"Fassungsvermoegen momentan:  ";va"Zusch
        auer"
662 LZ  LOCATE 14,14:PRINT"Stadionausbau maximal:        100.000 P
        laetze"
663 MQ  LOCATE 16,14:PRINT"Bauauftrag minimal:           5.000 Pla
        etze"
664 Gp  LOCATE 18,14:PRINT"Bauauftrag maximal:           10.000 Pl
        aetze"
665 Ve  LOCATE 20,14:PRINT"Preis fuer 5.000 Plaetze:     50.000 DM
        "
666 tI  LOCATE 22,14:PRINT"Dauer des Ausbaus:            4 Spielta
        ge"
667 x1  VEla:
668 Oh  LOCATE 28,14
669 ys  INPUT"Wieviel Plaetze wollen Sie Kaufen (in Tausend) ";P
670 jv  IF P>10 THEN GOTO VEla
671 xF  WQ=P:WQ=WQ*10000
672 eR  IF WQ>ka THEN BEEP:GOTO VEla
673 qs  IF P < 5 OR P > 10 THEN BEEP:GOTO VEla
674 3o  P=P*1000:vb=va+P
675 ee  IF vb <=100000& THEN GOTO Bauaus
676 w3  IF vb >=100000& THEN BEEP:GOTO VEla
677 3L  Bauaus:
678 bQ  bmst=1:FERTIG=sp+4:PP=P:ka=ka-WQ:bms=0:PRINT:vb=0
679 es  COLOR 1,0:CLS:GOSUB MITTIG:LOCATE 14,21
680 LT  PRINT"In Ordnung, Ihr Stadium wird umgehend ausgebaut"
681 8E  COLOR 1,0:GOSUB warten:GOTO MANAGERMARKT
682 uw  Verw:
683 CZ  CLS:GOSUB MITTIG:LOCATE 14,4
684 HX  PRINT"Bauauftraege lassen es zur Zeit nicht zu, das Sie Ih
        r Stadion ausbauen"
685 2m  bms=0:COLOR 1,0
686 wQ  GOSUB warten:GOTO MANAGERMARKT
687 3u  UEEINSCHUB:
688 Mx  CLS:GOSUB USCH:LOCATE 3,34
689 Jx  PRINT"U E F A - C U P"
690 8e  LINE (230,62) - (430,82),2,bf
691 S3  LINE (220,57) - (420,77),1,bf
692 1B  IF sp=11 OR sp=14 THEN GOTO maleue1
693 6C  IF sp=20 OR sp=23 THEN GOTO maleue2
694 mx  IF sp=29 OR sp=32 THEN GOTO maleue3
695 LN  LINE (47,99)-(596,234),2,bf
696 69  LINE (37,94)-(586,229),1,bf
697 X7  GOTO springback
698 JG  maleue1:
699 aH  LINE (47,99)-(596,170),2,bf
700 L3  LINE (37,94)-(586,165),1,bf
701 bB  GOTO springback
702 QO  maleue2:
703 SV  LINE (47,99)-(596,136),2,bf
704 zp  LINE (37,94)-(586,131),1,bf
705 fF  GOTO springback
706 XW  maleue3:
707 IL  LINE (47,99)-(596,122),2,bf
708 37  LINE (37,94)-(586,117),1,bf
709 tu  springback:
```

```
710 oQ  RETURN
711 Jh  Bank:
712 Jb  Spielen:
713 KU  IF SPIELFREI=0 THEN
714 pE3   CLS:GOSUB MITTIG:LOCATE 14,6
715 Ij    PRINT"Vor einem Spieltag muessen Sie einmal nach Ihrer
          Mannschaft sehen"
716 Jb    COLOR 1,0
717 Rv    GOSUB warten:GOTO MANAGERMARKT
718 WP0 END IF
719 RE  IF sp=1 THEN
720 6j3   CLS:GOSUB MITTIG:LOCATE 14,31
721 WX    PRINT"EINEN MOMENT BITTE...":COLOR 1,0
722 aT0 END IF
723 zv  Bundesliga:
724 tQ  FOR i = 1 TO 18
725 CY  MN$(i)=Team$(i)
726 9P  NEXT i
727 ia  FOR i = 1 TO 18  STEP 2
728 9C  r=(a(i)):o=(a(i+1))
729 ku  TKON(r)=0:TKON(o)=0:HEIMTORE(r)=0
730 Py  zu=1:GOSUB zufalleins
731 RH  IF z=0 THEN GOTO Spielwertung
732 OH  IF z=1 THEN GOTO KONDITIONSTOR
733 wb  KONDITIONSTOR:
734 oe  IF kn(r) > kn(o) THEN TKON(r)=1
735 35  IF kn(r)=kn(o) THEN TKON(r)=1
736 nL  IF kn(o) > kn(r) THEN TKON(o)=1
737 68  Spielwertung:
738 LY  TOR=ks(r)-ka(o)
739 ze  WHILE TOR > 0
740 XL  IF TOR > 9 THEN TOR=9
741 XZ  GOSUB plusto
742 ON  Tore(r)=plto
743 uu  GOTO weitermachen
744 RF  WEND
745 3h  WHILE TOR = 0
746 fE  zu=1:GOSUB zufalleins
747 fA  IF     z = 0 THEN Tore(r)=1
748 gB  IF     z = 1 THEN Tore(r)=0
749 00  GOTO weitermachen
750 XL  WEND
751 7k  WHILE TOR < 0
752 JL  GOSUB minusto
753 2A  Tore(r)=mito
754 55  GOTO weitermachen
755 cQ  WEND
756 7Y  weitermachen:
757 hu  TOR=ks(o)-ka(r)
758 Ix  WHILE TOR > 0
759 qe  IF TOR > 9 THEN TOR=9
760 qs  GOSUB plusto
761 bX  Tore(o)=plto
762 mm  GOTO weiterimtext
763 kY  WEND
764 M0  WHILE TOR = 0
765 yX  zu=1:GOSUB zufalleins
766 jB  IF     z = 0 THEN Tore(o)=1
767 kC  IF     z = 1 THEN Tore(o)=0
768 ss  GOTO weiterimtext
769 qe  WEND
770 Q3  WHILE TOR < 0
771 ce  GOSUB minusto
772 FK  Tore(o)=mito
773 xx  GOTO weiterimtext
774 vj  WEND
775 zQ  weiterimtext:
776 9i  zu=1:GOSUB zufalleins
777 tA  IF z=1 THEN HEIMTORE(r)=1
778 qC  IF z=2 THEN HEIMTORE(r)=0
779 Dj  WHILE sp=1
780 Dm  zu=1:GOSUB zufalleins
781 Pz  IF z=1 THEN Tore(r)=Tore(r)+1
782 Af  IF z=2 THEN Tore(o)=Tore(o)+1
783 XH  GOTO auswertenSp
784 5t  WEND
785 9K  auswertenSp:
786 Dg  INSTORE(r)=Tore(r)+TKON(r)+HEIMTORE(r)
787 4W  INSTORE(o)=Tore(o)+TKON(o)
788 BV  WHILE TE$(r)=E$(1)
789 3A  IF INSTORE(o)=0 THEN GOTO SDF
790 Na  IF INSTORE(r) < 3 GOTO SDF
791 Ox  zu=1:GOSUB zufalleins
```

```
792 6R  IF z=1 THEN INSTORE(r)=INSTORE(r)-1:INSTORE(o)=INSTORE(o)-
        1
793 JJ  GOTO SDF
794 F3  WEND
795 1C  SDF:
796 DU  WHILE TE$(o)=E$(1)
797 Sq  IF INSTORE(r)=0 THEN GOTO SDT
798 oC  IF INSTORE(o) < 3 GOTO SDT
799 W5  zu=1:GOSUB zufalleins
800 NT  IF z=1 THEN INSTORE(o)=INSTORE(o)-1:INSTORE(r)=INSTORE(r)-
        1
801 LZ  GOTO SDT
802 NB  WEND
803 nS  SDT:
804 Sq  IF TE$(r)=E$(1) THEN GESCHOSS=INSTORE(r)
805 2K  IF TE$(o)=E$(1) THEN GESCHOSS=INSTORE(o)
806 1T  WHILE INSTORE(r) > INSTORE(o)
807 xX  IF TE$(r)=E$(1) THEN mf=mf+2
808 gg  WIN(r)=WIN(r)+1
809 3A  LOST(o)=LOST(o)+1
810 EA  TABPUN(r)=TABPUN(r)+2
811 Wp  TOTOPUN(r)=1
812 6W  TABTORE(r)=TABTORE(r)+INSTORE(r)
813 Uq  TABGEPU(o)=TABGEPU(o)+2
814 V1  TABTORE(o)=TABTORE(o)+INSTORE(o)
815 v6  TABGEPU(r)=TABGEPU(r)+0
816 yb  AB=INSTORE(o)
817 J1  TABGETORE(r)=TABGETORE(r)+AB
818 c1  abc=TABGETORE(r)
819 nW  ac=INSTORE(r)
820 h3  TABGETORE(o)=TABGETORE(o)+ac
821 U7  acd=TABGETORE(o)
822 Hp  DIFFERENZ(o)=TABTORE(o)-acd
823 S4  DIFFERENZ(r)=TABTORE(r)-abc
824 ow  GOTO weitergehen
825 kY  WEND
826 Fw  WHILE INSTORE(o) > INSTORE(r)
827 5c  IF TE$(o)=E$(1) THEN mf=mf+2
828 Zi  WIN(o)=WIN(o)+1
829 iv  LOST(r)=LOST(r)+1
830 A0  TABPUN(o)=TABPUN(o)+2
831 tD  TOTOPUN(r)=2
832 Qq  TABTORE(r)=TABTORE(r)+INSTORE(r)
833 g0  TABGEPU(o)=TABGEPU(o)+0
834 MZ  TABGEPU(r)=TABGEPU(r)+2
835 qM  TABTORE(o)=TABTORE(o)+INSTORE(o)
836 4n  ac=INSTORE(r)
837 yK  TABGETORE(o)=TABGETORE(o)+ac
838 1O  acd=TABGETORE(o)
839 Ly  AB=INSTORE(o)
840 g5  TABGETORE(r)=TABGETORE(r)+AB
841 z0  abc=TABGETORE(r)
842 1N  DIFFERENZ(r)=TABTORE(r)-abc
843 cA  DIFFERENZ(o)=TABTORE(o)-acd
844 8G  GOTO weitergehen
845 4s  WEND
846 b2  WHILE INSTORE(r) = INSTORE(o)
847 X6  IF TE$(r)=E$(1) THEN mf=mf+1
848 4y  EQUAL(r)=EQUAL(r)+1
849 qe  EQUAL(o)=EQUAL(o)+1
850 q1  TABPUN(r)=TABPUN(r)+1
851 TI  TABPUN(o)=TABPUN(o)+1
852 8Q  TOTOPUN(r)=0
853 bn  TABGEPU(r)=TABGEPU(r)+1
854 5Q  TABGEPU(o)=TABGEPU(o)+1
855 nD  TABTORE(r)=TABTORE(r)+INSTORE(r)
856 Kw  TABGETORE(r)=TABGETORE(r)+(INSTORE(o))
857 C1  TABTORE(o)=TABTORE(o)+INSTORE(o)
858 7g  TABGETORE(o)=TABGETORE(o)+(INSTORE(r))
859 br  DIFFERENZ(o)=TABTORE(o)-(TABGETORE(o))
860 xM  DIFFERENZ(r)=TABTORE(r)-(TABGETORE(r))
861 PX  GOTO weitergehen
862 L9  WEND
863 1a  weitergehen:
864 uy  WHILE TOTOTIPPEN=1
865 M9  IF   TOTOPUN(r)=TOTOTIP(r) THEN GEWINN = GEWINN +1
866 Mm  GOTO weitergehen2
867 QE  WEND
868 OF  weitergehen2:
869 Mj  IF TE$(r)=E$(1) THEN Anzeige$(1)=E$(1):Anzeigeto(1)=INSTOR
        E(r):Anzeige$(2)=TE$(o):Anzeigeto(2)=INSTORE(o)
870 32  IF TE$(o)=E$(1) THEN Anzeige$(1)=TE$(r):Anzeigeto(1)=INSTO
        RE(r):Anzeige$(2)=E$(1):Anzeigeto(2)=INSTORE(o)
871 OH  WHILE sp<18
872 aE  IF TE$(r)=E$(1) THEN STATITO(sp)=INSTORE(r):STATITO2(sp)=I
        NSTORE(o)
873 Mx  IF TE$(o)=E$(1) THEN STATITO(sp)=INSTORE(o):STATITO2(sp)=I
        NSTORE(r)
874 yp  GOTO DODOIT
875 YM  WEND
876 as  DODOIT:
877 aq  NEXT i
878 U1  Bundesliga2:
879 Yd  letsstart:
880 RV  IF Anzeigeto(1)=0 THEN GOTO TZQ
881 11  TZR:
882 bn  FOR i = 1 TO Anzeigeto(1)
883 YO  Wideranzei1:
884 DT  zu=86
885 mU  z=0:RANDOMIZE TIMER
886 g9  z=4+INT(RND*(zu)+.5)
887 N2  Tito1(i)=z
888 l1  NEXT i
889 54  TZQ:
890 i1  Anzeiges=Anzeigeto(1)+Anzeigeto(2)
891 PA  IF Anzeigeto(2)=0 THEN GOTO SortierenAn
892 r0  FOR i = Anzeigeto(1)+1 TO Anzeiges
893 kJ  Wideranzei:
894 Nd  zu=86
895 we  z=0:RANDOMIZE TIMER
896 qJ  z=4+INT(RND*(zu)+.5)
897 XC  Tito1(i)=z
898 vB  NEXT i
899 0Q  SortierenAn:
900 dc  IF Anzeigeto(1)=0 AND Anzeigeto(2)=0 THEN NOHAPPENS=1:GOTO
         NOSORT
901 IN  FOR i = 1 TO Anzeiges
902 PQ  FOR j = 1 TO Anzeiges
903 0g  IF Tito1(j) > Tito1(i) THEN SWAP Tito1(i),Tito1(j)
904 3K  NEXT j
905 2I  NEXT i
906 NS  FOR i = 1 TO Anzeiges
907 Jp  IF Tito1(i)=Tito1(i+1) THEN Tito1(i)=Tito1(i)-1
908 5L  NEXT i
909 QV  FOR i = 1 TO Anzeiges
910 Ms  IF Tito1(i)=Tito1(i+1) THEN Tito1(i)=Tito1(i)-1
911 8O  NEXT i
912 KR  NOSORT:
913 1S  COLOR 7,3
914 IO  CLS
915 SJ  LINE (170,48)-(460,115),0,bf
916 lY  LINE (173,51)-(457,112),1,bf
917 ud  LINE (175,53)-(455,110),2,bf
918 js  LINE (210,115)-(220,140),0,bf
919 w9  LINE (410,115)-(420,140),0,bf
920 Rt  LINE (0,140)-(640,170),1,bf
921 Hx  LINE (0,140)-(640,141),2,bf
922 4Z  LINE (0,169)-(640,171),2,bf
923 Bk  LINE (0,170)-(640,256),5,bf
924 tK  LINE (0,185)-(640,188),7,bf
925 4N  PALETTE 7,.4,.6,1
926 4g  COLOR 7,1:LOCATE 19,7:PRINT"C O M M O D O R E  C O M P U T
        E R"
927 HP  PALETTE 7,1,1,1
928 6s  COLOR 4,1:LOCATE 21,15:PRINT"Kiss me  AMIGA !"
929 CK  COLOR 2,1:LOCATE 19,55:PRINT"MARKT & TECHNIK"
930 jZ  COLOR 6,1:LOCATE 20,53:PRINT"Software - Schulung"
931 2n  LOCATE 21,51:PRINT"Buecher - Zeitschriften"
932 Ff  COLOR 7,2
933 an  FOR w = 1 TO 91
934 MP  IF w = 1 THEN
935 OI3    LOCATE 9,26:PRINT Anzeige$(1):LOCATE 9,47:PRINT"0":LOCA
        TE 10,34:PRINT"-"
936 sA     LOCATE 11,26:PRINT Anzeige$(2):LOCATE 11,47:PRINT"0":LO
        CATE 13,38:PRINT"15 : 30"
937 9p     FOR u = 1 TO 1000:NEXT u
938 4x0 END IF
939 mv  IF w<30 THEN LOCATE 13,38:PRINT"15 :";w+30:GOTO GB
940 0V  IF w=30 THEN LOCATE 13,38:PRINT"16 : 00":GOTO GB
941 VX  IF w>30 AND w<40 THEN LOCATE 13,38:PRINT"16 : 0";:PRINT
        USING"# ";w-30:GOTO GB
942 y2  IF w>39 AND w<46 THEN LOCATE 13,38:PRINT"16 :";w-30:GOTO
         GB
```

Listing 1. »Anpfiff« (Fortsetzung)

```
943 7A  IF w>45 AND w<76 THEN LOCATE 13,38:PRINT"16 :";w-16:GOTO
        GB
944 ZF  IF w=76 THEN LOCATE 13,38:PRINT"17 : 00":GOTO GB
945 Na  IF w>76 AND w<86 THEN LOCATE 13,38:PRINT"17 : 0";:PRINT
        USING"#";w-76:GOTO GB
946 sW  IF w>85 AND w <=91 THEN LOCATE 13,38:PRINT"17 :";w-76:GO
        TO GB
947 rS  GB:
948 tW  IF w = 45 THEN GOSUB Schan
949 Ol  IF NOHAPPENS = 1 THEN GOTO ANSENDE
950 FY  FOR u = 1 TO Anzeiges
951 eF  IF Tito1(u)=w THEN
952 Of5      GOSUB slope
953 O17        FOR ih = 1 TO 3
954 2hB            GOSUB slope
955 iR             LOCATE 9,26:PRINT SPACE$(29):LOCATE 9,38:PRINT"
                   T  O  R":FOR jh = 1 TO 400:NEXT jh
956 Ll7        NEXT ih
957 RV5      ih=0
958 zN       FOR jh = 1 TO 1000:NEXT jh
959 kL       zu=1:z=0:RANDOMIZE TIMER
960 c1       z=0+INT(RND*(zu)+.5)
961 y9       LOCATE 9,33:PRINT SPACE$(17)
962 Wh       LOCATE 11,34:PRINT SPACE$(21)
963 Iq       IF Anzeigeto(1)=0 AND Anzeigeto(2)>0 THEN GOSUB Nogo
             al1:GOTO TWS
964 Qu       IF Anzeigeto(2)=0 AND Anzeigeto(1)>0 THEN GOSUB Nogo
             al2:GOTO TWS
965 5i       IF z=0 THEN GOSUB Nogoal2
966 2k       IF z=1 THEN GOSUB Nogoal1
967 XQO END IF
968 LJ  TWS:
969 Su  NEXT u
970 5C  GOTO TUIP
971 AK  Nogoal1:
972 Fl  Anto2=Anto2+1
973 IU  Nogoal3:
974 aB  LOCATE 9,26
975 yn  PRINT Anzeige$(1)
976 qK  LOCATE 9,47
977 if  PRINT Anto1
978 hs  LOCATE 10,33
979 ig  PRINT" -";:PRINT SPACE$(19)
980 v9  LOCATE 11,26
981 6w  PRINT Anzeige$(2)
982 4L  LOCATE 11,47
983 qo  PRINT Anto2
984 OQ  IF nogo=1 THEN RETURN
985 QE  Anzeigeto(2)=Anzeigeto(2)-1
986 Gs  RETURN
987 Te  Nogoal2:
988 oP  LOCATE 9,26
989 bo  Nogoal4:
990 D2  PRINT Anzeige$(1)
991 Qz  Anto1=Anto1+1
992 6a  LOCATE 9,47
993 yv  PRINT Anto1
994 x8  LOCATE 10,33
995 yw  PRINT" -";:PRINT SPACE$(19)
996 BP  LOCATE 11,26
997 MC  PRINT Anzeige$(2)
998 Kb  LOCATE 11,47
999 64  PRINT Anto2
1000 eg IF nogo=1 THEN RETURN
1001 ZL Anzeigeto(1)=Anzeigeto(1)-1
1002 W8 RETURN
1003 Su TUIP:
1004 qA ANSENDE:
1005 ev FOR Zeit1= 1 TO 300:NEXT Zeit1
1006 7b NEXT w
1007 nP FOR w = 1 TO 3000:NEXT w
1008 7z Anto1=0:Anto2=0:Anzeigeto(1)=0:Anzeigeto(2)=0:NOHAPPENS=0
1009 6t FOR i = 1 TO 14
1010 Cd Tito1(i)=0
1011 kO NEXT i
1012 sy CLS
1013 6O COLOR 1,0
1014 eN CLS:GOSUB USCH:LOCATE 3,28
1015 SY PRINT"E N D E R G E B N I S S E"
1016 fg LINE (47,48)-(596,199),2,bf
1017 CO LINE (37,43)-(586,194),1,bf
1018 uE LINE (230,214)-(430,234),2,bf
1019 Ji LINE (220,209)-(420,229),1,bf
```

```
1020 zy FOR i = 1 TO 18 STEP 2
1021 sv r=(a(i)):o=(a(i+1))
1022 ON LOCATE 6+i,11
1023 IZ PRINT TE$(r) TAB(34)"-"TAB(38) TE$(o) TAB(62); INSTORE(r)
        TAB(65);":" TAB(66) INSTORE(o)
1024 xD NEXT i
1025 38 SPIELFREI=0
1026 lI FOR i = 1 TO 18
1027 mi Team$(i)=TE$(i)
1028 eZ TPUN(i)=TABPUN(i)
1029 JE TGE(i)=TABGETORE(i)
1030 jy TBT(i)=TABTORE(i)
1031 z6 TGP(i)=TABGEPU(i)
1032 NC TBD(i)=DIFFERENZ(i)
1033 Sg L(i)=LOST(i)
1034 OB k(i)=EQUAL(i)
1035 LQ H(i)=WIN(i)
1036 gZ INSTORE(i)=0
1037 Jg IF sp = 1 THEN MN$(i)=Team$(i)
1038 BR NEXT i
1039 BG FOR i =  1 TO 5
1040 Vr LOCATE 28,35
1041 N4 PRINT"              "
1042 h8 FOR j=1 TO 250:NEXT j
1043 Yu LOCATE 28,35
1044 m8 PRINT"Bitte warten"
1045 Qm FOR j=1 TO 200:NEXT j
1046 JZ NEXT i
1047 6d FOR i = 1 TO 18
1048 Dg FOR j = 1 TO 18
1049 Mg IF TPUN(j) < TPUN(i) THEN SWAP TPUN(i),TPUN(j):SWAP TP(i)
        ,TP(j):SWAP Team$(i),Team$(j):SWAP TGE(i), TGE(j):SWAP TGP(
        i),TGP(j):SWAP TBT(i),TBT(j):SWAP TBD(i),TBD(j):SWAP H(i),H
        (j)
1050 K7 IF TRUN(j) < TRUN(i) THEN SWAP k(i),k(j):SWAP(i),L(j)
1051 JD IF TPUN(j) = TPUN(i) THEN GOSUB VERGLEICHEN
1052 R1 NEXT j
1053 Qg NEXT i
1054 vY LOCATE 28,35:PRINT"  Taste        "
1055 Y9 GOSUB warten
1056 n5 COLOR 1,0
1057 e3 TABELLE:
1058 Kv CLS:GOSUB USCH:LOCATE 3,34
1059 jF PRINT"T A B E L L E"
1060 pC LINE (16,50) - (630,60),2,bf
1061 rD LINE (6,45) - (623,55),1,bf
1062 AN LOCATE 7,2
1063 Jj PRINT"Pl  von    Mannschaft                Punkte   Tore
          Dif    Gew   Un    Ver":PRINT
1064 48 LINE (16,71) - (630,226),2,bf
1065 ez LINE (6,66) - (623,221),1,bf
1066 Pw FOR i = 1 TO 18
1067 Wz FOR j = 1 TO 18
1068 Fb IF Team$(i)=MN$(j) THEN
1069 u4 LOCATE 9+i,2
1070 DO PRINT USING"##. ";i;:PRINT"(";:PRINT USING"##";j;:PRIN
        T")   ";:PRINT Team$(i);TAB(37):PRINT USING"##  ";TPUN(i);
        :PRINT":";
1071 ke PRINT USING"##    ";TGP(i);:PRINT USING"##  ";TBT(i);:PR
        INT":";:PRINT USING"##   ";TGE(i);:PRINT USING"###";TBD
        (i);
1072 L1 PRINT"     ";:PRINT"(";:PRINT USING"##";H(i);:PRINT")";:
        PRINT"  ";:PRINT"(";:PRINT USING"##";k(i);:PRINT")";:PRIN
        T"  ";:PRINT"(";:PRINT USING"##";L(i);:PRINT")"
1073 F8 END IF
1074 n4 NEXT j
1075 m2 NEXT i
1076 7P COLOR 1,0
1077 uV GOSUB warten
1078 34 IF Tabaus = 1 THEN RETURN
1079 c9 FOR i = 1 TO 18
1080 wa IF Team$(i)=E$(1) THEN Vaplatz=i
1081 s8 NEXT i
1082 fC FOR i = 1 TO 18
1083 7D IF zu$(i)="verletzt" THEN
1084 HA3    zu(i)=zu(i)-1
1085 R3     IF zu(i)=0 THEN zu$(i)="OK":GOTO HOSPI
1086 kz     IF zu(i)>0 THEN GOTO HOSPI
1087 TMO END IF
1088 kk IF zu$(i)="Spital" THEN GOTO HOSPI
1089 n3 IF zu$(i)="Rote Karte" AND rote <1 THEN rote=rote+1:GOTO
        HOSPI
1090 PN zu$(i)="OK"
```

```
1091 8u  HOSPI:
1092 3J  NEXT i
1093 N2  THO:
1094 wf  CLS:GOSUB USCH:LOCATE 3,28
1095 a4  PRINT"S T A T U S B E R I C H T"
1096 os  LINE (16,55) - (300,130),2,bf
1097 oW  LINE (6,50) - (290,125),3,bf
1098 KK  LINE (16,158) - (300,233),2,bf
1099 vG  LINE (6,153) - (290,228),5,bf
1100 1M  LINE (330,55) - (620,130),2,bf
1101 BX  LINE (320,50) - (610,125),6,bf
1102 Yg  LINE (330,158) - (620,233),2,bf
1103 cP  LINE (320,153) - (610,228),4,bf
1104 W6  zu=2:GOSUB zufalleins
1105 GB  IF z = 0 THEN WETTER=1
1106 OL  IF z = 1 THEN WETTER=2
1107 WV  IF z = 2 THEN WETTER=3
1108 Ph  HALLO=0
1109 md  WETTERBER:
1110 7e  FOR i = 1 TO 18
1111 Xh  IF Team$(a(i))=E$(1) THEN HALLO=a(i)
1112 Nd  NEXT i
1113 vW  IF HALLO=1 THEN WETTER=3
1114 2Z  IF HALLO=2 THEN WETTER=3
1115 9c  IF HALLO=3 THEN WETTER=3
1116 Mz  IF sp > 26 AND HALLO=4 THEN WETTER=3
1117 Q4  IF sp > 26 AND HALLO=5 THEN WETTER=3
1118 Zr  HALLO=0
1119 YD  LOCATE 8,2:COLOR 2,3:PRINT"KONDITION:"
1120 hj  LOCATE 10,9:PRINT"Ihre Spieler verlieren"
1121 Wm  LOCATE 12,12:PRINT"um den Faktor ";WETTER
1122 c3  LOCATE 14,14:PRINT"an Leistung !"
1123 Vv  LOCATE 21,42:COLOR 2,4:PRINT"VERLETZUNG:"
1124 3C  LOCATE 8,42:COLOR 2,6:PRINT"GELBE KARTE:"
1125 Ze  IF GESCHOSS < 1 THEN LOCATE 21,2:COLOR 2,5:PRINT"Kein Tor
         ":GOTO Nichtwich
1126 yn  IF GESCHOSS < 2 THEN LOCATE 21,2:COLOR 2,5:PRINT"TOR:":GO
         TO TORkoenige
1127 5I  LOCATE 21,2:COLOR 2,5
1128 iq  PRINT"TORE:"
1129 2D  TORkoenige:
1130 18  FOR i = 1 TO GESCHOSS
1131 nF  Giltnicht:
1132 GG  zu=11:GOSUB zufallszahl
1133 hf  IF zu$(i)<>"OK" THEN GOTO Giltnicht
1134 8K  LOCATE 22+i,2:PRINT na$(z);:TORKO(z)=TORKO(z)+1:PRINT TAB
         (27):PRINT"(";:PRINT USING"# #";TORKO(z);:PRINT".)"
1135 kO  NEXT i
1136 aT  Nichtwich:
1137 XP  GESCHOSS=0
1138 ts  FOR i = 1 TO 11
1139 ZS  IF ko(i)=0 THEN GOTO isnich
1140 ea  IF zu$(i)="verliehen" THEN GOTO isnich
1141 Tm  ko(i)=ko(i)-WETTER
1142 mz  isnich:
1143 s8  NEXT i
1144 jf  FOR i = 12 TO 18
1145 2x  IF ko(i)=0 THEN GOTO datgehtnich
1146 94  IF zu$(i)="verliehen" THEN GOTO datgehtnich
1147 CB  IF zu$(i)="gesperrt" THEN GOTO datgehtnich
1148 uK  IF zu$(i)="Rote Karte" THEN GOTO datgehtnich
1149 g1  IF ko(i) <= 99 THEN ko(i)=ko(i)+3
1150 87  IF ko(i) > 100 THEN ko(i)=100
1151 6V  datgehtnich:
1152 1H  NEXT i
1153 tR  FOR i = 2 TO 18
1154 Ku  zu=2:GOSUB zufalleins
1155 SI  IF z = 0 THEN kn(i)=kn(i)-6
1156 SW  IF z = 1 THEN kn(i)=kn(i)-12
1157 Wb  IF z = 2 THEN kn(i)=kn(i)-21
1158 7N  NEXT i
1159 Qp  zu=8:GOSUB zufallszahl:Plfak=z
1160 xg  IF Ueok=1 THEN
1161 Xx3     zu=9:GOSUB zufallszahl:Plfak=z
1162 gZ0 END IF
1163 7Y  IF z=0 THEN Plfak=1
1164 hP  IF sp >=6 AND Vaplatz>=14 THEN Plfak=1
1165 mq  FOR i = 1 TO Plfak
1166 La  dasuebenwirnoch:
1167 sq  zu=2:GOSUB zufallszahl
1168 iA  Neuez=z
1169 Qt  IF z=0 THEN GOTO dasuebenwirnoch
1170 hO  Nochmalziehen:
1171 TU  zu=18
1172 6A  GOSUB zufallszahl
1173 jx  IF z = 1 THEN GOTO Nochmalziehen
1174 14  Neuezzw=z
1175 4b  IF Neuez = 1 AND ka(Neuezzw) >= 23 THEN GOTO Nochmalziehe
         n
1176 aQ  IF Neuez = 2 AND ks(Neuezzw) >= 23 THEN GOTO Nochmalziehe
         n
1177 WU  IF sp <=4 AND Neuez=1 AND ka(Neuezzw)>=20 THEN GOTO Noch
         malziehen
1178 jO  IF sp <=4 AND Neuez=2 AND ks(Neuezzw)>=20 THEN GOTO Noch
         malziehen
1179 jm  IF Neuez = 1 THEN ka(Neuezzw)=ka(Neuezzw)+1
1180 Pb  IF Neuez = 2 THEN ks(Neuezzw)=ks(Neuezzw)+1
1181 Uk  NEXT i
1182 ES  IF sp=ROTKAR THEN GOTO Karten
1183 sT  zu=3:GOSUB zufalleins
1184 81  IF z = 2 THEN GOTO ABRECHNUNG1
1185 B8  IF z = 0 THEN GOTO Karten
1186 9R  IF z = 1 THEN zu = 10
1187 PP  IF z = 3 THEN GOTO Karten
1188 3T  IF sp >= 5 AND Vaplatz >=13 THEN GOTO ABRECHNUNG1
1189 uC  zaelerverletzt=0
1190 Pw  FOR i = 1 TO 18
1191 al  IF zu$(i)="verletzt" THEN zaelerverletzt=zaelerverletzt+1
1192 fv  NEXT i
1193 Xc  IF zaelerverletzt >=2 THEN GOTO THFROESE
1194 hx  Invalide:
1195 mU  z=0:RANDOMIZE TIMER
1196 Uu  z=1+INT(RND*(zu)+.5)
1197 N9  IF zu$(z)="verliehen" THEN GOTO Invalide
1198 uC  IF zu$(z)="verletzt" THEN GOTO Invalide
1199 wY  IF zu$(z)="Spital" THEN GOTO Invalide
1200 jg  IF zu$(z)="Rote Karte" THEN GOTO Invalide
1201 OE  COLOR 2,4:LOCATE 23,42:PRINT na$(z)
1202 wN  zu$(z)="verletzt":verz=z
1203 2b  zu=1:GOSUB zufalleins
1204 aI  IF z=0 THEN zu(verz)=2
1205 aI  IF z=1 THEN zu(verz)=1
1206 xN  IF Vaplatz >=9 THEN zu(verz)=1
1207 QV  LOCATE 23,65:PRINT zu(verz);:PRINT"Sp."
1208 7g  zu=1:GOSUB zufalleins
1209 yL  IF z=0 THEN GOTO THFROESE
1210 fO  Invali2:
1211 uX  z=0:zu=10:RANDOMIZE TIMER
1212 kA  z=1+INT(RND*(zu)+.5)
1213 vL  IF zu$(z)="verliehen" THEN GOTO Invali2
1214 Nk  IF zu$(z)="verletzt" THEN GOTO Invali2
1215 Fw  IF zu$(z)="Spital" THEN GOTO Invali2
1216 vz  IF zu$(z)="Rote Karte" THEN GOTO Invali2
1217 mL  LOCATE 25,42:PRINT na$(z)
1218 Cd  zu$(z)="verletzt":verz=z
1219 Ir  zu=1:GOSUB zufalleins
1220 sb  IF z=1 THEN zu(verz)=2
1221 oV  IF z=0 THEN zu(verz)=1
1222 Dd  IF Vaplatz >=9 THEN zu(verz)=1
1223 qx  LOCATE 25,65:PRINT zu(verz);:PRINT"Sp."
1224 Xq  THFROESE:
1225 Ox  zu=1:GOSUB zufalleins
1226 Vs  IF z=0 THEN GOTO ABRECHNUNG1
1227 qU  Karten:
1228 PZ  z=0:zu = 10:RANDOMIZE TIMER
1229 1R  z=1+INT(RND*(zu)+.5)
1230 2W  IF zu$(z)="verliehen" THEN GOTO Karten
1231 Hr  IF zu$(z)="verletzt" THEN GOTO Karten
1232 5O  IF zu$(z)="Rote Karte" THEN GOTO Karten
1233 6r  GELBE(z)=GELBE(z)+1
1234 NL  LOCATE 10,42:COLOR 2,6:PRINT na$(z);:PRINT TAB (67):PRINT"
         ( ";:PRINT USING"# ";GELBE(z);:PRINT".)"
1235 JE  IF GELBE(z) = 2 THEN zu$(z)="gesperrt":LOCATE 13,42:PRINT"
         SPERRUNG:":LOCATE 13,52:PRINT na$(z):GELBE(z)=0
1236 5a  Kartenrot:
1237 Gk  IF sp=ROTKAR THEN
1238 Zj3    z=0:zu = 10:RANDOMIZE TIMER
1239 Bb     z=1+INT(RND*(zu)+.5)
1240 AE     IF zu$(z)="verliehen" THEN GOTO Kartenrot
1241 MT     IF zu$(z)="gesperrt" THEN GOTO Kartenrot
1242 qx     IF fa(z) < 3 THEN GOTO Kartenrot
1243 1w     LOCATE 15,42:PRINT"ROTE KARTE:":LOCATE 15,54:PRINT na$(
            z)
```

Listing 1. »Anpfiff« (Fortsetzung)

```
1244 Kg     zu$(z)="Rote Karte"
1245 1u0 END IF
1246 d1  ABRECHNUNG1:
1247 sA  COLOR 1,0
1248 fG  GOSUB warten
1249 Mv  FOR i=1 TO 11
1250 au  IF ko(i) < 76 AND na$(i) <>"" THEN sonstiges=1
1251 cs  NEXT i
1252 Pw  FOR i = 1 TO 18
1253 nS  IF TORKO(i) > 3  AND STOPPER(i)=0 THEN sonstiges=1
1254 fv  NEXT i
1255 Yw  IF sonstiges = 0 THEN GOTO nichtnotwen
1256 CP  CLS:GOSUB USCH:sonstiges=0:LOCATE 3,35:PRINT"A C H T U N G
         "
1257 3k  LINE (190,55)-(470,130),2,bf
1258 Df  LINE (180,50)-(460,125),5,bf
1259 P7  LOCATE 8,34:COLOR 2,5:PRINT"FAKTORERHOEHUNG:"
1260 lK  LINE (190,158)-(470,233),2,bf
1261 fD  LINE (180,153)-(460,228),4,bf
1262 Au  LOCATE 21,34:COLOR 2,4:PRINT"KONDITIONSGEFAHR:"
1263 a7  FOR i = 1 TO 18
1264 Ch  IF STOPPER(i)=1 THEN GOTO NICHTTWICE
1265 Q1  IF TORKO(i) > 3 THEN fa(i)=fa(i)+1:STOPPER(i)=1:COLOR 2,5
         :LOCATE 10+zeilenvor,34:zeilenvor=zeilenvor+1:PRINT na$(i):
         mf=mf+3
1266 ds  NICHTTWICE:
1267 s8  NEXT i
1268 v4  zeilenvor=0
1269 8o  FOR i=1 TO 18
1270 WB  IF i < 12 AND ko(i) < 76 AND na$(i) <> "" THEN GOSUB K
         ONMEL
1271 UV  IF ko(i) < 70 AND na$(i) <> "" THEN zu$(i)="Spital":IAR
         SCH(i)=1
1272 xD  NEXT i
1273 33  zeilenvor=0:COLOR 1,0:sonstiges=0
1274 5g  GOSUB warten
1275 6s  nichtnotwen:
1276 08  GOSUB Zufaelligkeit
1277 Ye  REM fuer zufaelligkeit minus betrag erst bei Abrechnung ab
         ziehen
1278 6j  CLS:GOSUB MITTIG:LOCATE 14,31
1279 WX  PRINT"EINEN MOMENT BITTE...":COLOR 1,0
1280 cg  WHILE TOTOTIPPEN=1
1281 XT  IF GEWINN < 5 THEN KGEW$="Kein Gewinn":MONEYHON=0:GOTO AB
         RECHNUNG2
1282 7A  FOR i = 1 TO 17 STEP 2
1283 Yd  HTOT=HTOT+a(i)
1284 J7  HTOT2=HTOT2+a(i+1)
1285 AQ  NEXT i
1286 mf  IF HTOT < HTOT2 THEN MUL6=MONETE*2:MUL7=MONETE*3:MUL8=MON
         ETE*4:MUL9=MONETE*5
1287 OS  IF HTOT = HTOT2 THEN MUL6=MONETE*2+5000:MUL7=MONETE*3+1000
         0&:MUL8=MONETE*4+15000&:MUL9=MONETE*5+20000&
1288 Oz  IF HTOT > HTOT2 THEN MUL6=MONETE*2+10000:MUL7=MONETE*3+20
         000&:MUL8=MONETE*4+25000&:MUL9=MONETE*5+35000&
1289 48  IF GEWINN=5 THEN MONEYHON=MONETE
1290 4h  IF GEWINN=6 THEN MONEYHON=MONETE+MUL6
1291 Ap  IF GEWINN=7 THEN MONEYHON=MONETE+MUL7
1292 Gx  IF GEWINN=8 THEN MONEYHON=MONETE+MUL8
1293 M5  IF GEWINN=9 THEN MONEYHON=MONETE+MUL9
1294 sv  mf=mf+3
1295 Oh  HTOT=0:HTOT2=0:MONETE=0
1296 tr  GOTO ABRECHNUNG2
1297 MA  WEND
1298 Vu  ABRECHNUNG2:
1299 CB  GOSUB LWBerechnung
1300 Bi  FOR i = 1 TO 18
1301 qb  LOHNGES=LOHNGES+Lohnmo(i)
1302 Rh  NEXT i
1303 tk  Varia=Vaplatz
1304 SG  Vari=19-Varia:GRMIETE=Vari*2000
1305 9L  IF ka < 0 THEN KREDI = 8000
1306 Rx  IF ka <= -25000 THEN KREDI = 15000
1307 Vn  IF ka <= -35000& THEN KREDI = 20000
1308 au  IF ka <= -50000& THEN KREDI = 25000
1309 bi  MINUSGES=LOHNGES+kd+KREDI+GRMIETE
1310 Zc  FOR i = 1 TO 17 STEP 2
1311 fl  IF TE$(a(i)) = E$(1) THEN ZUVA2=a(i+1)
1312 wK  IF TE$(a(i+1))= E$(1) THEN ZUVA2=a(i)
1313 cs  NEXT i
1314 Pw  FOR i = 1 TO 18
1315 ax  IF MN$(i)=E$(1) THEN ZUpl1=i
1316 oF  IF MN$(i)=E$(ZUVA2) THEN ZUpl2=i
1317 gw  NEXT i
1318 QO  GESZ=ZUpl1+ZUpl2
1319 tw  IF GESZ > 30 THEN ZUSCHAU=va-40000&
1320 4X  IF GESZ < 30 THEN ZUSCHAU=va-30000
1321 Cn  IF GESZ < 25 THEN ZUSCHAU=va-20000
1322 2h  IF GESZ < 20 THEN ZUSCHAU=va-15000
1323 S8  IF GESZ < 15 THEN zu=10:GOSUB zufallszahl:ZUSCHAU=va-z*10
         00
1324 o9  IF GESZ < 10 THEN zu=7:GOSUB zufallszahl:ZUSCHAU=va-z*100
         0
1325 RO  IF GESZ < 5 THEN ZUSCHAU=va
1326 ps  FOR i = 1 TO 17 STEP 2
1327 OV  IF TE$(a(i+1))= E$(1) THEN KEINH=1:GOTO KEINHEIM
1328 r7  NEXT i
1329 jT  EINAH=ZUSCHAU*3
1330 2Y  KEINHEIM:
1331 NR  zu=10:GOSUB zufallszahl
1332 Cd  SPEN=z*1000
1333 mY  IF sp>=9 AND Varia>=11 THEN SPEN=SPEN+10000
1334 6b  IF Varia=1 THEN BONUSP=5000
1335 bY  Varia=0
1336 b6  PLUSGES=EINAH+SPEN+MONEYHON+BONUSP
1337 WO  kat=ka:ka=ka+PLUSGES-MINUSGES
1338 WX  CLS:COLOR 0,1
1339 J1  LINE (20,16) - (148,33),2,bf
1340 Sz  LINE (10,11) - (138,28),4,bf
1341 Pq  LINE (200,16) - (620,71),2,bf
1342 sO  LINE (190,11) - (610,66),1,bf
1343 pr  LINE (20,88) - (148,105),2,bf
1344 LE  LINE (10,83) - (138,100),5,bf
1345 Br  LINE (200,88)-(620,143),2,bf
1346 YO  LINE (190,83)-(610,138),1,bf
1347 fz  LINE (20,160)-(148,177),2,bf
1348 wx  LINE (10,155)-(138,172),6,bf
1349 9S  LINE (200,160)-(620,208),2,bf
1350 Yw  LINE (190,155)-(610,203),1,bf
1351 j9  LINE (200,225)-(395,240),2,bf
1352 8d  LINE (190,220)-(385,235),1,bf
1353 c1  LINE (415,225)-(620,240),2,bf
1354 Ql  LINE (405,220)-(610,235),1,bf
1355 95  LOCATE 3,5:COLOR 2,4:PRINT"1. Ausgaben";:COLOR 0,1
1356 lJ  PRINT TAB(30):PRINT USING"# # # # # # #";GRMIETE;:PRINT"
         DM";:PRINT TAB(44) "Grundmiete"
1357 9a  PRINT TAB(28) "+" TAB(30):PRINT USING"# # # # # # #";LOHN
         GES;:PRINT" DM";:PRINT TAB(44) "Lohnkosten"
1358 jX  PRINT TAB(28) "+" TAB(30):PRINT USING"# # # # # # #";kd;:
         PRINT" DM";:PRINT TAB(44) "Ereignis"
1359 Z4  PRINT TAB(28) "+" TAB(30):PRINT USING"# # # # # # #";KRED
         I;:PRINT" DM";:PRINT TAB(44) "Kreditzinsen"
1360 Ao  PRINT TAB(30)"----------"
1361 mR  PRINT TAB(28) "=" TAB(30):PRINT USING"# # # # # # #";MINU
         SGES;:PRINT" DM";:PRINT TAB(44) "Gesamt"
1362 iT  LOCATE 12,5:COLOR 2,5:PRINT"2. Einahmen";:COLOR 0,1
1363 b0  IF KEINH=1 THEN PRINT TAB(30):PRINT USING"# # # # # # #";
         EINAH;:PRINT" DM";:PRINT TAB(44) "Kein Heimspiel":GOTO ABRE
         CHEINHALB
1364 dM  PRINT TAB(30):PRINT USING"# # # # # # #";EINAH;:PRINT" DM
         ";:PRINT TAB(44) "Zuschauereinahmen"
1365 7c  ABRECHEINHALB:
1366 U0  PRINT TAB(28) "+" TAB(30):PRINT USING"# # # # # # #";SPEN
         ;:PRINT" DM";:PRINT TAB(44) "Spenden"
1367 a4  PRINT TAB(28) "+" TAB(30):PRINT USING"# # # # # # #";MONE
         YHON;:PRINT" DM";:PRINT TAB(44) "Tototip"
1368 7y  PRINT TAB(28) "+" TAB(30):PRINT USING"# # # # # # #";BONU
         SP;:PRINT" DM";:PRINT TAB(44) "Bonus (1.Platz)"
1369 Jx  PRINT TAB(30)"----------"
1370 BH  PRINT TAB(28) "=" TAB(30):PRINT USING"# # # # # # #";PLUS
         GES;:PRINT" DM";:PRINT TAB(44) "Gesamt"
1371 lB  LOCATE 21,5:COLOR 2,6:PRINT"3. Gesamt:";:COLOR 0,1
1372 Tt  PRINT TAB(30):PRINT USING"# # # # # # #";kat;:PRINT" DM";
         :PRINT TAB(44) "Kapital"
1373 zo  PRINT TAB(28) "+" TAB(30):PRINT USING"# # # # # # #";PLUS
         GES;:PRINT" DM";:PRINT TAB(44) "Einnahmen"
1374 9d  PRINT TAB(28) "-" TAB(30):PRINT USING"# # # # # # #";MINU
         SGES;:PRINT" DM";:PRINT TAB(44) "Ausgaben"
1375 P3  PRINT TAB(30)"----------"
1376 vo  PRINT TAB(28) "=" TAB(30):PRINT USING"# # # # # # #";ka;:
         PRINT" DM";:PRINT TAB(44) "Kapital"
1377 kq  PRINT:PRINT:PRINT
1378 Fx  IF TOTOTIPPEN=0 THEN PRINT TAB(28):PRINT"Toto: nicht getip
         pt";:PRINT TAB(56)"Zuschauer: ";ZUSCHAU:GOTO ABRECHNUNG3
1379 TB  PRINT TAB(28):PRINT"Tototip: ";GEWINN;:PRINT" Richtige";:P
         RINT TAB(56)"Zuschauer: ";ZUSCHAU
```

```
1380 rH  ABRECHNUNG3:
1381 9B  kd=0
1382 Z8  Vaplatz=0:MINUSGES=0:LOHNGES=0:KREDI=0:GRMIETE=0
1383 cJ  EINAH=0:ZUSCHAU=0:SPEN=0
1384 vH  PLUSGES=0:MONEYHON=0:BONUSP=0:kat=0:KEINH=0
1385 6O  COLOR 1,0
1386 tU  GOSUB warten
1387 E3  IF ka < 0 THEN
1388 Eb3     voka=1+voka
1389 hJ     CLS:GOSUB USCH:LOCATE 3,35
1390 bf     PRINT"A C H T U N G"
1391 Jx     GOSUB MITTIG:LOCATE 14,21
1392 1F     PRINT"Ihr Kapital war zum";voka;:PRINT"Male unter DM 0.
            -"
1393 mx     GOSUB warten:COLOR 1,0
1394 QJ0 END IF
1395 Is  IF voka >=3 THEN
1396 iH3    CLS:GOSUB USCH:LOCATE 3,32
1397 Th     PRINT"S P I E L E N D E"
1398 rc     GOSUB MITTIG:LOCATE 14,3
1399 bt     PRINT"Aufgrund Ihrer schlechten Finanzpolitik werden Si
            e vom Manageramt enthoben"
1400 8h     GOSUB warten:COLOR 1,0:GOTO THEEND
1401 XQ0 END IF
1402 KA  IF sp=34 THEN COLOR 1,0:GOTO THEEND
1403 rR  CLS:GOSUB USCH:LOCATE 3,33
1404 4G  PRINT"A N G E B O T E"
1405 5B  VerleihSp=0:KaufSp=0:VerkaufSp=0
1406 jh  zu=2:GOSUB zufallszahl
1407 98  IF z=2 THEN
1408 KW3    LINE (20,70) - (620,90),2,bf
1409 Pq     LINE (10,65) - (610,85),1,bf
1410 q4     LOCATE 10,18
1411 Wj     PRINT"Es wird Ihnen ein Spieler zum Kauf angeboten":Kau
            fSp=1
1412 ib0 END IF
1413 Wo  IF ka <=0 THEN
1414 nZ3    LINE (20,102) - (620,122),2,bf
1415 YQ     LINE (10,97) - (610,117),1,bf
1416 4I     LOCATE 14,14
1417 fu     PRINT"Sie haben die Moeglichkeit einen Spieler zu verka
            ufen":VerkaufSp=1
1418 Je     GOTO isterledigt1
1419 pi0 END IF
1420 xv  zu=2:GOSUB zufallszahl
1421 NM  IF z=2 THEN
1422 vh3    LINE (20,102) - (620,122),2,bf
1423 gY     LINE (10,97) - (610,117),1,bf
1424 CQ     LOCATE 14,14
1425 n2     PRINT"Sie haben die Moeglichkeit einen Spieler zu verka
            ufen":VerkaufSp=1
1426 wp0 END IF
1427 42  zu=2:GOSUB zufallszahl
1428 UT  IF z=2 THEN
1429 aW3    LINE (20,134) - (620,154),2,bf
1430 yz     LINE (10,129) - (610,149),1,bf
1431 Xn     LOCATE 18,12
1432 Pt     PRINT"Ein Spieler kann zum naechsten Spieltag verliehen
             werden":VerleihSp=1
1433 fd     GOTO ister2
1434 4x0 END IF
1435 aM  isterledigt1:
1436 tB  IF ka <=0 THEN
1437 ie3    LINE (20,134) - (620,154),2,bf
1438 67     LINE (10,129) - (610,149),1,bf
1439 fv     LOCATE 18,12
1440 X1     PRINT"Ein Spieler kann zum naechsten Spieltag verliehen
             werden":VerleihSp=1
1441 B40 END IF
1442 On  ister2:
1443 AH  LINE (230,190) - (430,210),2,bf
1444 dV  LINE (220,185) - (420,205),1,bf
1445 wI  LOCATE 25,38
1446 NZ  PRINT"Taste":COLOR 1,0
1447 sT  GOSUB warten
1448 Kb  sp=sp+1:bms=1:TOTOTIPPEN=0:GEWINN=0:SPPA=0
1449 ZT  REM ab hier wird Spieltag gelesen
1450 Ix  IF sp=10 THEN GOSUB S2
1451 mY  IF sp=18 THEN GOSUB S1
1452 pc  IF sp=27 THEN GOSUB S2
1453 Kc  IF sp=2 OR sp=10 THEN pu=18
1454 Ib  IF sp=18 OR sp=27 THEN pu=18
1455 OM  IF sp=10 THEN GOTO S5
1456 8A  IF sp=18 OR sp=27 THEN GOTO S5
1457 iF  FOR i = 1 TO 18
1458 wK  pu=pu+1
1459 dW  a(i)=a(pu)
1460 zF  NEXT i
1461 uU  S5:
1462 jy  IF sp >=18 THEN
1463 253    FOR i = 1 TO 17 STEP 2
1464 1L     SWAP a(i),a(i+1)
1465 4K     NEXT i
1466 aT0 END IF
1467 3e  S6:
1468 WS  WHILE sp = 18
1469 zX  FOR i = 2 TO 18
1470 SO  kn(i)=1080
1471 AQ  NEXT i
1472 ct  FOR i =1 TO 18
1473 1a  IF na$(i) <> "" THEN ko(i)=100
1474 DT  NEXT i
1475 PT  GOTO STADIFRA
1476 F3  WEND
1477 rM  STADIFRA:
1478 8F  WHILE FERTIG=sp
1479 AY  bmst=0
1480 fU  va=va+PP
1481 nV  CLS:GOSUB MITTIG:LOCATE 14,27
1482 Yr  PRINT"Stadionausbau fertiggestellt ":COLOR 1,0
1483 S3  GOSUB warten
1484 pc  GOTO SCHLUSSWEI
1485 OC  WEND
1486 1F  SCHLUSSWEI:
1487 20  GOTO MANAGERMARKT
1488 9S  VERGLEICHEN:
1489 Wt  IF TBD(j) < TBD(i) THEN SWAP TBD(i),TBD(j):SWAP TP(i),TP(
         j):SWAP TBT(i),TBT(j):SWAP Team$(i),Team$(j):SWAP TGE(i), T
         GE(j):SWAP TGP(i),TGP(j)
1490 p3  IF TBD(j) < TBD(i) THEN SWAP TPUN(i),TPUN(j):SWAP H(i),H(
         j):SWAP k(i),k(j):SWAP L(i),L(j)
1491 MU  REM
1492 a6  IF TBD(j)=TBD(i) THEN GOSUB ver2
1493 R3  RETURN
1494 Tu  ver2:
1495 ON  IF TBT(j) < TBT(i) THEN SWAP TBT(i),TBT(j):SWAP TBD(i),TB
         D(j):SWAP TP(i),TP(j):SWAP Team$(i),Team$(j):SWAP TGE(i), T
         GE(j):SWAP TGP(i),TGP(j)
1496 zR  IF TBT(j) < TBT(i) THEN SWAP TPUN(i),TPUN(j):SWAP H(i),H(
         j):SWAP k(i),k(j):SWAP L(i),L(j)
1497 V7  RETURN
1498 gL  KONMEL:
1499 JJ  COLOR 2,4:LOCATE 23+zeilenvor,34
1500 zi  PRINT na$(i);:PRINT TAB (52):PRINT ko(i);:PRINT"%"
1501 v8  zeilenvor=zeilenvor+1
1502 aC  RETURN
1503 6o  THEEND:
1504 1J  COLOR 1,0
1505 L5  CLS:GOSUB MITTIG:LOCATE 14,29
1506 Lx  PRINT"S A I S O N E N D E !":mfneu=mf
1507 nn  FOR i = 15 TO 1 STEP-1
1508 W9  KNETq=50000&*i
1509 gR  IF ka >= KNETq THEN GOTO THATsIT
1510 n3  NEXT i
1511 rH  THATsIT:
1512 X1  KNET=i*3:mf=mf+KNET
1513 4B  IF ka < 0 THEN mf=mf-10:KNET=-10
1514 vd  IF va >= 90000& THEN mf=mf+75:tz=75:GOTO THATSITTWO
1515 fc  IF va >= 80000& THEN mf=mf+50:tz=50:GOTO THATSITTWO
1516 2i  IF va >= 70000& THEN mf=mf+25:tz=25:GOTO THATSITTWO
1517 vs  IF va >= 60000& THEN mf=mf+5:tz=5:GOTO THATSITTWO
1518 NY  THATSITTWO:
1519 iF  FOR i = 1 TO 18
1520 8I  IF Team$(a(i))=E$(1) THEN HALLO=a(i)
1521 yE  NEXT i
1522 od  IF HALLO=1 THEN mf=mf+150:ti=150:GOTO THATTHREE
1523 Yw  IF HALLO=2 THEN mf=mf+75:ti=75:GOTO THATTHREE
1524 Uz  IF HALLO=3 THEN mf=mf+50:ti=50:GOTO THATTHREE
1525 39  IF HALLO=4 THEN mf=mf+25:ti=25:GOTO THATTHREE
1526 NH  IF HALLO=5 THEN mf=mf+17:ti=17:GOTO THATTHREE
1527 xl  FOR i = 6 TO 15
1528 qc  IF HALLO=i THEN RATEM=18-i:ti=RATEM
1529 6M  NEXT i
```

Listing 1. »Anpfiff« (Fortsetzung)

```
1530 Xn  mf=mf+RATEM
1531 Cj  THATTHREE:
1532 Di  IF HALLO=1 AND voka < 3 THEN
1533 yM3    CLS:GOSUB MITTIG:LOCATE 14,5
1534 AY     PRINT"Glueckwunsch, Sie sind mit Ihrem Team Deutscher M
            eister geworden !"
1535 It     GOSUB warten
1536 ib0 END IF
1537 Mh  COLOR 1,0:CLS:GOSUB USCH:LOCATE 3,28
1538 VK  PRINT"E N D A B R E C H N U N G"
1539 L6  LINE (90,48)-(550,195),2,bf
1540 sQ  LINE (80,43)-(540,190),1,bf
1541 RO  LOCATE 9,10
1542 Rr  PRINT TAB(21):PRINT"  Managerpunkte bisher:     ";mfneu;:PR
         INT TAB(54):PRINT"Punkte":PRINT
1543 Mf  PRINT TAB(21):PRINT"+ Stadionausbau:           ";tz;:PRINT
          TAB(54):PRINT"Punkte":PRINT
1544 Qu  PRINT TAB(21):PRINT"+ Tabellenplatz:           ";ti;:PRINT
          TAB(54):PRINT"Punkte":PRINT
1545 u8  PRINT TAB(21):PRINT"+ Restkapital:             ";KNET;:PRI
         NT TAB(54):PRINT"Punkte":PRINT
1546 Rf  PRINT TAB(21):PRINT"---------------------------------------
         -":PRINT
1547 iv  PRINT TAB(21):PRINT"= Managerpunkte gesamt:    ";mf;:PRINT
          TAB(54):PRINT"Punkte"
1548 vM  LINE (260,216)-(390,231),2,bf
1549 rG  LINE (250,212)-(380,227),1,bf
1550 XD  LOCATE 28,38:PRINT"Taste":COLOR 1,0
1551 Y9  GOSUB warten
1552 we  CLS:GOSUB MITTIG:LOCATE 14,27
1553 11  PRINT"E N D E       Bis dann..."
1554 oj  END
1555 VG  GOTO TOPIC
1556 pH  Teingabe:
1557 A8  GOTO MANAGERMARKT
1558 In  Schan:
1559 k5  FOR Zeit=1 TO 7000
1560 wb  NEXT Zeit
1561 X9  RETURN
1562 Na  USCH:
1563 0I  COLOR 0,1
1564 a6  LINE (190,10) - (470,35),2,bf
1565 cm  LINE (180,5) - (460,30),1,bf
1566 cE  RETURN
1567 iz  Farben:
1568 D6  PALETTE 0,.4,.4,.4 'dunkelgrau
1569 Ya  PALETTE 1,.6,.6,.6 'mittelgrau
1570 bn  PALETTE 2,0,0,0    'schwarz
1571 Cj  PALETTE 4,.93,.2,0 'feuerwehrrot
1572 wo  PALETTE 5,.6,1,.1  'giftgruen
1573 3K  PALETTE 6,1,1,0    'gelb
1574 Ds  PALETTE 3,.47,.87,1'himmelblau
1575 zL  PALETTE 7,1,1,1    'weiss
1576 m0  RETURN
1577 rx  slope:
1578 xC  LOCATE 9,26:PRINT SPACE$(29):LOCATE 9,38:PRINT SPACE$(10)
1579 1D  IF ih > 0 THEN jh = 0:FOR jh=1 TO 1000:NEXT jh
1580 An  IF ih = 0 THEN LOCATE 10,34:PRINT SPACE$(2):LOCATE 11,26:P
         RINT SPACE$(9):LOCATE 11,34:PRINT USING"# # ";w-1;:PRINT".S
         pielminute     "
1581 rT  RETURN
1582 zS  MITTIG:
1583 Kc  COLOR 0,1
1584 cb  LINE (20,99) - (620,119),2,bf
1585 Om  LINE (10,94) - (610,114),1,bf
1586 wY  RETURN
1587 4Q  warten:
1588 iG  a$=INKEY$
1589 kA  IF a$= ""THEN warten
1590 Oc  RETURN
1591 Mr  Abfrageschleife:
1592 q8  Bes$=INKEY$:IF Bes$<>"" THEN Bes$= UCASE$(Bes$)
1593 d1  IF Bes$ = "J" THEN Bes$="":GOTO Veri
1594 GI  IF Bes$ = "N" THEN Bes$="":GOTO Negi
1595 5h  RETURN
1596 WK  Veri:
1597 4A  ON ver GOTO Roger1,,Roger3,Roger4,Roger5,Roger6
1598 a5  Negi:
1599 1H  ON neg GOTO Noger1,Noger2,Noger3,Noger4,MANAGERMARKT,MANAG
         ERMARKT
1600 xh  minusto:
1601 7z  z=0:mito=0
1602 ja  IF TOR > -5 AND TOR <= -3 THEN zu=7:GOSUB zufallszahl1
1603 ZZ  IF TOR > -3 THEN zu=4:GOSUB zufallszahl
1604 MM  IF z=2 THEN mito=1
1605 Fr  RETURN
1606 Hm  plusto:
1607 Ip  plto=0
1608 dr  IF TOR < 3 THEN
1609 BR3    zu=3:GOSUB zufallszahl1
1610 cq     IF z = 3 THEN plto = 1
1611 Xn     IF z = 3 AND TOR = 2 THEN plto = 2
1612 an     IF z = 2 THEN plto = 1
1613 xq0 END IF
1614 n2  IF TOR = 3 THEN
1615 GG3    zu=4:GOSUB zufallszahl
1616 8g     IF z = 3 OR z = 4 THEN plto = 2
1617 fs     IF z = 2 THEN plto = 1
1618 2v0 END IF
1619 y9  IF TOR = 4 THEN
1620 Wo3    zu=5:GOSUB zufallszahl1
1621 1J     IF z = 5 THEN plto = 3
1622 Em     IF z = 3 OR z = 4 THEN plto = 2
1623 ly     IF z = 2 THEN plto = 1
1624 810 END IF
1625 AH  IF TOR = 5 THEN
1626 bd3    zu=6:GOSUB zufallszahl
1627 EY     IF z = 6 THEN plto = 4
1628 T4     IF z = 4 OR z = 5 THEN plto = 3
1629 yD     IF z = 3 THEN plto = 2
1630 s5     IF z = 2 THEN plto = 1
1631 F80 END IF
1632 LT  IF TOR > 5 THEN
1633 nq3    zu=7:GOSUB zufallszahl
1634 Qk     IF z > 4 THEN plto=4:GOTO plback
1635 9w     IF z > 2 AND z < 5 THEN plto=3:GOTO plback
1636 pO     plto=2
1637 LV     IF TOR = 6 AND z = 2 THEN plto=0
1638 Vc     IF TOR = 7 AND z = 2 THEN plto=1
1639 NG0 END IF
1640 NJ  plback:
1641 pR  RETURN
1642 HD  zufalleins:
1643 0i  z=0:RANDOMIZE TIMER
1644 Se  z=0+INT(RND*(zu)+.5):RETURN
1645 r1  zufallszahl:
1646 31  z=0:RANDOMIZE TIMER
1647 Zm  z=1+INT(RND*(zu)+.5):RETURN
1648 9a  zufallszahl1:
1649 6o  z=0:RANDOMIZE TIMER
1650 cp  z=1+INT(RND*(zu)+.5):RETURN
1651 q0  U1:
1652 qZ  DATA "Borussia Dortmund","Werder Bremen","Bayern Muenchen"
         ,"1. FC Koeln"
1653 HB  DATA "Waldhof Mannheim","1. FC Nuernberg","VFB Stuttgart",
         "Hamburger SV"
1654 On  DATA "Borussia Mgladbach","Bayer Leverkusen","Eintracht Fr
         ankfurt"
1655 eq  DATA "Karlsruher SC","Hannover 96","1. FC Kaiserslautern",
         "Kickers Stuttgart"
1656 aX  DATA "FC St. Pauli","VFL Bochum","Bayer Uerdingen"
1657 wT  FOR i = 1 TO 18
1658 qk  READ E$(i)
1659 CS  NEXT i
1660 5f  U3:
1661 Nw  DATA "Brehme","Abwehr","Wheelan","Mittelfeld","Altobelli",
         "Angriff"
1662 rB  DATA "Michel","Abwehr","Butragueno","Mittelfeld","Vialli",
         "Angriff"
1663 c9  DATA "Dassajew","Torwart","Koeman","Abwehr","Matthaeus","M
         ittelfeld"
1664 8w  DATA "Protassow","Angriff","Rijkaard","Abwehr","Gullit","M
         ittelfeld"
1665 ON  DATA "van Basten","Angriff"
1666 bS  FOR i = 1 TO 13
1667 oL  READ Vna$(i),Vpo$(i)
1668 Lb  NEXT i
1669 Hs  U4:
1670 GR  DATA 3,3,3,4,4,4,5,5,5,5,6,6,6
1671 gX  FOR i = 1 TO 13
1672 jr  READ Vfa(i)
1673 Qg  NEXT i
1674 P1  U5:
1675 NU  DATA 17,16,1080,17,16,1080,17,16,1090,17,16,1095,16,16,110
         0,16,16,1100
1676 30  DATA 16,17,1080,16,17,1080,16,17,1080,16,17,1090,16,17,109
```

```
          5,17,17,1065
1677 JT   DATA 17,17,1065,18,15,1080,15,18,1080,15,18,1085,18,15,108
          5
1678 Mu   FOR i = 2 TO 18
1679 OH   READ ka(i),ks(i),kn(i)
1680 Xn   NEXT i
1681 ZC   U6:
1682 4w   DATA 4,3,3,4,2,3,4,2,2,2,4,2,2,2,2,2
1683 Ap   FOR i = 1 TO 16
1684 Hh   READ fa(i)
1685 cs   NEXT i
1686 hL   U7:
1687 ea   DATA "Teddy de Beer","Torwart","Mc Leod","Abwehr","Rulaend
          er","Abwehr"
1688 2O   DATA "Kutowski","Abwehr","Pagelsdorf","Abwehr","Moeller","
          Mittelfeld"
1689 4f   DATA "Rummenigge","Angriff","Zorc","Mittelfeld","Dickel","
          Angriff"
1690 Bk   DATA "Kroth","Mittelfeld","Mill","Angriff"
1691 OE   DATA "Lusch","Abwehr","Meyer","Torwart","Breitzke","Angrif
          f"
1692 os   DATA "Nikolic","Angriff","Storck","Abwehr"
1693 Kz   FOR i = 1 TO 16
1694 Yx   READ na$(i),po$(i)
1695 m2   NEXT i
1696 V1   S1:
1697 kr   stl=162:GOSUB S1
1698 lp   DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
1699 gB   DATA 4,1,14,11,18,15,12,9,10,7,8,5,6,3,2,17,16,13
1700 KO   DATA 1,6,3,8,5,10,7,12,9,14,11,16,13,18,2,4,17,15
1701 T1   DATA 12,5,18,11,14,7,10,3,8,1,6,2,4,17,15,13,16,9
1702 15   DATA 1,10,3,12,5,14,7,16,9,18,11,15,4,6,2,8,17,13
1703 ds   DATA 18,7,14,3,12,1,10,2,8,4,6,17,13,11,15,9,16,5
1704 rm   DATA 1,14,3,16,5,18,7,15,9,13,6,8,4,10,2,12,17,11
1705 Mo   DATA 12,4,18,3,14,2,10,6,8,17,11,9,13,7,15,5,16,1
1706 Pr   DATA 1,18,3,15,5,13,7,11,6,12,8,10,4,14,2,16,17,9
1707 tV   RETURN
```

```
1708 kH   S2:
1709 y5   stl=144:GOSUB S1
1710 rQ   DATA 15,1,18,2,14,6,12,8,10,17,9,7,11,5,13,3,16,4
1711 mp   DATA 1,13,3,11,5,9,10,12,8,14,6,16,4,18,2,15,17,7
1712 qq   DATA 18,6,14,10,12,17,7,5,9,3,11,1,13,2,15,4,16,8
1713 JJ   DATA 1,9,3,7,12,14,10,16,8,18,6,15,4,13,2,11,17,5
1714 uS   DATA 18,10,14,17,5,3,7,1,9,2,11,4,13,6,15,8,16,12
1715 Ec   DATA 1,5,3,17,12,18,10,15,8,13,6,11,4,9,2,7,16,14
1716 WJ   DATA 18,14,3,1,5,2,7,4,9,6,11,8,13,10,15,12,17,16
1717 gi   DATA 1,17,14,15,12,13,10,11,8,9,6,7,4,5,2,3,16,18
1718 4g   RETURN
1719 jC   S1:
1720 jw   IF sp=18 THEN
1721 MW3     RESTORE S1
1722 NK      FOR i = 1 TO 162
1723 9A      READ a(i)
1724 FV      NEXT i
1725 Bn      RETURN
1726 mf0  END IF
1727 7W   IF sp= 27 THEN
1728 Zf3     RESTORE S2
1729 MT      FOR i = 1 TO 144
1730 GH      READ a(i)
1731 Mc      NEXT i
1732 Iu      RETURN
1733 tm0  END IF
1734 1i   FOR i = 1 TO stl
1735 E7   a(i)=0
1736 Rh   NEXT i
1737 ol   FOR i = 1 TO stl
1738 OP   READ a(i)
1739 Uk   NEXT i
1740 Q2   RETURN
(C) 1989 M&T
```

Listing 1. »Anpfiff« (Schluß)

PATIENCE

Taktisches Denkvermögen und einen sicheren Blick benötigen Sie, wenn Sie bei diesem Spiel erfolgreich sein wollen. Meistern Sie diese Herausforderung.

Um eine Variante des bekannten Kartenspiels handelt es sich bei dem Programm »Patience« (Listing 1). Ziel des Spieles ist es, die 52 Blatt eines Kartenspiels nach Farben getrennt in einer bestimmten Reihenfolge auf einen »Stack« (Stapel) zu legen. Dabei machen verschiedene Regeln, die beim Auflegen der Karten beachtet werden müssen, jede Partie zu einem wahren Geduldsspiel, bei dem besonders taktisches Denken gefragt ist. Wenn Sie Listing 1 eingeben, steht dem Spiel-Spaß nichts mehr im Weg.

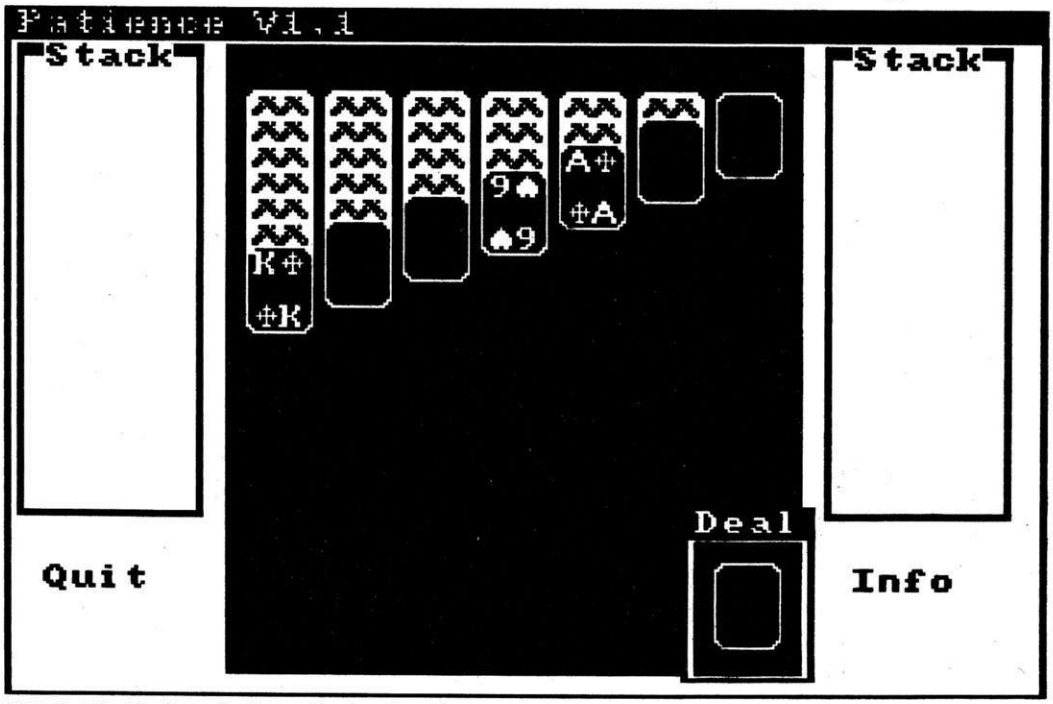


Bild 1. Die Karten sind gemischt, Ihre Sitzung bei »Patience« kann beginnen

Zu Beginn des Spiels mischt der Computer die Karten und legt auf dem Spielfeld, das in sieben Spalten eingeteilt ist, 28 Karten auf. Dabei ist nur jeweils die oberste Karte einer Spalte sichtbar (Bild 1). Die restlichen Karten liegen mit der Rückseite nach oben unter der aufgedeckten Karte. In der ersten Spalte befinden sich insgesamt sieben Karten, in der zweiten sechs, in der dritten fünf und so weiter. Sie müssen nun versuchen, die Karten

Hierarchische Strukturen

nach Spiel-Farben getrennt auf den »Stacks«, die sich links und rechts vom Spielfeld befinden, aufzulegen. Die Stacks sind dabei nach einer bestimmten Hierachie aufgebaut: Zuunterst liegt das »As« der jeweiligen Farbe, danach folgen die Karten in aufsteigender Reihenfolge, also die Blätter »2« bis »10« und »Bube«, »Dame«, »König« (Tabelle 1). In Bild 2 sehen Sie, wie langsam

die ersten Stacks an den Seiten des Spielfeldes entstehen.

Auf die Stacks können immer nur jene Karten gelegt werden, die in den Spalten oben liegen. Es müssen also innerhalb des Spielfeldes die Karten so aufgelegt und zwischen den Stapeln getauscht werden, daß die jeweils für die Stacks benötigten Karten oben liegen und dann abgelegt werden können. Damit die Angelegenheit nicht zu einfach wird, gelten innerhalb des Spielfeldes ganz bestimmte Regeln, wie die Karten umgelegt werden dürfen.

Es gilt das Prinzip der »hierarchisch absteigenden Reihenfolge mit alternierender Farbe«. Alles klar? Mit anderen Worten: Auf eine Karte kann jeweils nur die nächste rangniedrigere Karte einer anderen Farbe gelegt werden. Mit Farbe ist hier aber nicht die Spielfarbe, sondern Rot oder Schwarz gemeint. Auf eine »rote« Karte (Herz oder Karo) kann also nur eine »schwarze« (Pik oder Kreuz) gelegt werden, und umgekehrt. Karten werden innerhalb des Spielfeldes verschoben werden, indem man auf dem numerischen Zahlenblock der Tastatur die Quell- und die Zielspalte (1 bis 7) nacheinander eingibt. Normalerweise wird dann die oberste Karte der Quell- zur Zielspalte verschoben (sofern die Regeln dies zulassen). Befindet sich jedoch unter der Karte, die verschoben werden soll, bereits eine hierarchische Anordnung nach dem »Rot-Schwarz-Prinzip«, so werden diese Karten mitverschoben, was natürlich das Spiel etwas einfacher macht.

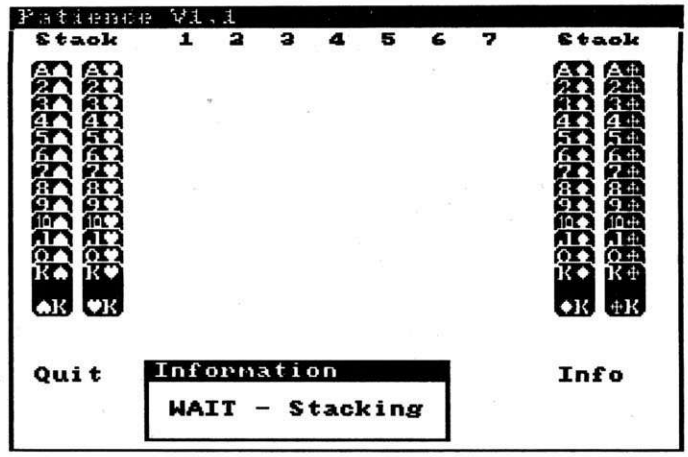


Bild 3. Geschafft, die Herausforderung des Computers ist bestanden — winkt auch im nächsten Spiel das Glück?

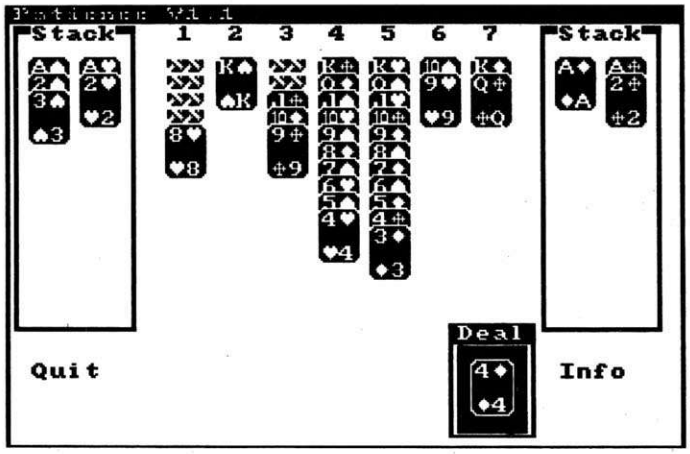


Bild 2. Auf dem Weg zur Lösung: Die ersten Stacks bilden sich bereits

Wird eine oder mehrere Karten verschoben, wird die jeweils darunterliegende aufgedeckt, sofern diese noch verdeckt war. Beim Jonglieren mit den Karten ist zu beachten, daß in eine leere Spalte nur ein König gesetzt werden kann.

Ab auf den Stapel

Wie kann man nun Karten auf den Stapel befördern? Ganz einfach: Wenn Sie die »ENTER«-Taste betätigen, werden alle Karten, die auf dem Spielfeld zuoberst liegen (auch die oberste Karte des weiter unter beschriebenen Päckchens) und nach den Regeln auf einen Stack passen, automatisch abgelegt (Bild 2).

Damit haben wir aber noch nicht alle Möglichkeiten des Spiels ausgeschöpft. Wie gesagt, besteht das Kartenspiel aus 52 Karten. Es befinden sich jedoch bei Beginn nur 28 davon auf dem Spielfeld. Der Rest liegt in einem Stapel, dessen oberste Karte Sie in der rechten unteren Ecke des Bildschirms sehen. Geraten Sie im Spiel an einen toten Punkt, können Sie von diesem Packen jederzeit eine passende Karte nehmen. Dies geschieht einfach, indem Sie die Taste <0> und anschließend die Taste der Spalte drücken, auf welche die Karte befördert werden soll. Sind Sie mit der Karte nicht zufrieden, können Sie die Taste <0> auch mehrmals betätigen, bis eine passende Karte erscheint. Allerdings wird immer nur ein Drittel der noch verfügbaren Karten angezeigt. Das heißt, nur jede dritte Karte des Reservestapels wird angezeigt. Wird nun eine Karte abgenommen, so werden natürlich wieder andere Karten des Stapels mit dem Rest angezeigt.

Herz	Karo	Pik	Kreuz
As	As	As	As
2	2	2	2
3	3	3	3
...	...	...	...
10	10	10	10
Bube	Bube	Bube	Bube
Dame	Dame	Dame	Dame
König	König	König	König

Tabelle 1. In dieser Hierarchie müssen die Karten auf den Stacks abgelegt werden

Leichter als man denkt

Sie sehen, Patience ist ein Spiel mit einer Vielzahl von Regeln und Gesetzmäßigkeiten. In der Theorie mag manches etwas verwirrend sein. Spätestens nach ein, zwei Spielen, werden Sie sehen, daß alles viel einfacher ist, als es klingt. Damit ist aber jetzt nicht das Spiel an sich, sondern der Umgang mit den Regeln gemeint. Das Spiel selbst wird Ihnen sicherlich so manches Kopfzerbrechen bereiten. Zeigt Ihr Bildschirm aber Ähnlichkeiten mit Bild 3, haben Sie eine Lösung gefunden.

Wenn Sie einmal gar keinen Ausweg mehr wissen, können Sie das Spiel mit <Q> abbrechen. Wenn Sie die darauf folgende Frage mit <Y> beantworten, wird das Programm verlassen. Drücken Sie <N>, beginnt das Spiel erneut — Sie können einen weiteren Versuch starten.

Bitte betätigen Sie während des Spiels keine der beiden Maustasten, da dies den Programmablauf stören könnte. Wenn Ihnen einmal die Regeln entfallen sollten, können Sie jederzeit mittels <I> eine — allerdings englische — Kurzanleitung ansehen.

(Martin Jobst/rs)

Programmname:	Patience
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

```
Programmautor: Nikolaj Stigo
-----------------------------

 1 tL0 ''''''''''''''''''''''''''''''''
 2 SF  ' "  P A T I E N C E - V1.1 "
 3 kQ  ' "  programmed by           "
 4 Yc  ' "  Nikolaj Stig           "
 5 nz  ' "  Gartnergade 5          "
 6 18  ' "  DK-2200 Kbenhavn N     "
 7 kO  ' "  Danmark                "
 8 W3  ' "  (W) August 1988        "
 9 1T  ''''''''''''''''''''''''''''''''
10 1b  DimVariables
11 XW  InitVariables
12 6L  SCREEN 2,320,211,5,1
13 WL  WINDOW 1,"Patience V1.1",,16,2
14 aA  MakeColors
15 jH  Start:
16 fn  WINDOW 3,"Deal",(208,152)-(240,192),16,2
17 9M  LINE (0,0)-(32,40),3,bf
18 h3  WINDOW OUTPUT 1
```

```
 19 ir   MakePlayboard
 20 RV   DisplayMessageEnd
 21 os   ShuffleCards
 22 Fw   DisplayDeal
 23 VX   PlayCards
 24 ip   GOTO Start
 25 BI   '''''''''''''''''''''''''''''''''''''
 26 ZG   SUB DisplayStack(i) STATIC
 27 It2    place=LEN(Stack$(i))
 28 xR     Xcor=i*3-3:Ycor=1
 29 30     IF i>2 THEN Xcor=Xcor+25
 30 fI     IF place>=1 THEN
 31 j84      Ycor=(ASC(LEFT$(Stack$(i),1)+CHR$(0)) AND 15)
 32 jD       PlaceCard Xcor,Ycor,LEFT$(Stack$(i),1)
 33 TM2    END IF
 34 ac0  END SUB
 35 Br   SUB MakePlayboard STATIC
 36 1K2    COLOR 13,0
 37 Jj     LOCATE 21,33:PRINT "I";
 38 MW     LOCATE 21,2:PRINT "Q";
 39 8S     COLOR 14,0
 40 1H     LOCATE 21,34:PRINT "nfo";
 41 Hf     LOCATE 21,3:PRINT "uit";
 42 IZ     PATTERN ,Empty
 43 kF     COLOR 13,10
 44 rK     LINE(64,0)-(240,192),2,bf
 45 MX     COLOR 13,2
 46 qq     LINE(66,4)-(238,190),2,bf
 47 4Q     FOR i=0 TO 6
 48 Fd4      LOCATE 1,i*3+10:PRINT USING "#";i+1;
 49 sx2    NEXT
 50 zy     SCROLL(73,0)-(232,7),4,0
 51 wT     COLOR 13,12
 52 K3     LINE(0,0)-(56,144),14,bf
 53 FX     LINE(248,0)-(304,144),14,bf
 54 hk     COLOR 13,4
 55 jy     LINE(2,4)-(54,142),4,bf
 56 Tr     LINE(250,4)-(302,142),4,bf
 57 13     LOCATE 1,33:PRINT "Stack";
 58 jR     LOCATE 1,2:PRINT "Stack";
 59 z10  END SUB
 60 05   SUB DisplayDeal STATIC
 61 y62    IF LEN(Deal$)>=1 THEN
 62 bp4      WINDOW OUTPUT 3
 63 P6       PlaceCard 0,0,LEFT$(Deal$,1)
 64 Rn       WINDOW OUTPUT 1
 65 eN2    ELSE
 66 L24      WINDOW CLOSE 3
 67 1u2    END IF
 68 8A0  END SUB
 69 zP   SUB ShuffleCards STATIC
 70 lw2    COLOR 13,2
 71 u1     LOCATE 10,11
 72 MX     PRINT "--  W  A  I  T  --";
 73 6F     LOCATE 12,11
 74 eT     PRINT "- Card shuffling -";
 75 ap     Deal$=""
 76 d1     FOR i=1 TO 7
 77 Df4      Back$(i)=""
 78 ru       FaceUp$(i)=""
 79 MR2    NEXT
 80 Yt     FOR i=1 TO 4
 81 ny4      Stack$(i)=""
 82 PU2    NEXT
 83 hs     FOR k=0 TO 3
 84 hI4      FOR i=1 TO 13
 85 uj6        Deal$=Deal$+CHR$((i OR 16*k))
 86 Fu4      NEXT i,k
 87 aj2    RANDOMIZE TIMER
 88 uj     FOR i=1 TO 200
 89 o04      Dummy1=INT(RND*51+1)
 90 rS       Dummy2=INT(RND*51+1)
 91 hU       Dum1$=MID$(Deal$,Dummy1,1)
 92 rg       Dum2$=MID$(Deal$,Dummy2,1)
 93 nb       MID$(Deal$,Dummy1,1)=Dum2$
 94 le       MID$(Deal$,Dummy2,1)=Dum1$
 95 yE2    NEXT i
 96 Wr     COLOR 2,2
 97 KR     LOCATE 10,11
 98 mx     PRINT "--  W  A  I  T  --";
 99 Wf     LOCATE 12,11
100 4t     PRINT "- Card shuffling -";
101 UP     i=1
```

```
102 1M     FOR Row=1 TO 7
103 h24      Ykor=Row
104 hQ       k=8-Row
105 PA       FOR Column=1 TO k
106 Lu6        Xkor=Column
107 aB         Card$=MID$(Deal$,i,1)
108 ke         i=i+1
109 O7         IF Row+Column<>8 THEN
110 4B8          Card$=CHR$(ASC(Card$) OR 128)
111 47           Back$(Column)=Back$(Column)+Card$
112 8P           Xkor=Column:Ykor=Row
113 JA           PlaceCard (Xkor)*3+5,Ykor,Back$(Column)
114 RA6        ELSE
115 qp8          FaceUp$(Column)=Card$
116 CT           Xkor=Column:Ykor=Row
117 tM           PlaceCard (Xkor)*3+5,Ykor,FaceUp$(Column)
118 qj6        END IF
119 054      NEXT
120 162    NEXT
121 yf     Deal$=RIGHT$(Deal$,24)
122 8v     EmptyBuffer
123 130  END SUB
124 FW   SUB MakeColors STATIC
125 3w2    PALETTE 0,.18,0,0          :REM background
126 Ck     PALETTE 1,.375,0,0         :REM border
127 6X     PALETTE 2,.3,.1,.3         :REM playfield
128 QH     PALETTE 3,.5,0,0           :REM dealfield
129 Cr     PALETTE 4,0,0,.25          :REM stackfield
130 Dh     PALETTE 5,0,0,0            :REM cardborder
131 At     PALETTE 6,1,.75,.625       :REM cardbackground
132 wP     PALETTE 7,.75,0,0          :REM card red
133 RG     PALETTE 8,.125,.125,.125   :REM card black
134 DS     PALETTE 9,.3,0,0           :REM cardback1
135 uE     PALETTE 10,.5,.3,.5        :REM playfield border
136 Fw     PALETTE 11,.75,0,0         :REM dealfield border
137 65     PALETTE 12,0,0,.4          :REM stackfield border
138 Bv     PALETTE 13,1,.5,.2         :REM text - plain
139 Sh     PALETTE 14,1,0,0           :REM text - allert
140 Ti     PALETTE 15,0,.2,0          :REM cardback2
141 JL0  END SUB
142 jO   SUB CheckEnd(Solved) STATIC
143 dV2    IF Solved THEN EXIT SUB
144 w1     Solved=0
145 bw     FOR i=1 TO 4
146 I04      IF LEN(Stack$(i))=13 THEN Solved=Solved+1
147 SX2    NEXT
148 x4     IF Solved=4 THEN
149 mp4      Message$="LET's PLAY AGAIN"
150 tM       DisplayMessage 7
151 qC       WINDOW OUTPUT 1
152 cP       EmptyBuffer
153 g1       Ask$=""
154 8p       WHILE (Ask$="")
155 MT6        Card$=CHR$((INT(RND*13)+1)+((INT(RND*4)+1)*16)+(INT(R
               ND*2)*128))
156 C0         Xcor=(INT(RND*7))*3+8
157 ul         Ycor=(INT(RND*5))*4+1
158 PW         PlaceCard Xcor,Ycor,Card$
159 Zv         Ask$=INKEY$
160 1p4      WEND
161 rC       FOR i=1 TO 4
162 6H6        Stack$(i)=""
163 in4      NEXT
164 vc       WINDOW CLOSE 3
165 rX       WINDOW CLOSE 2
166 eV2    END IF
167 vH     IF Solved<>4 THEN Solved=0
168 km0  END SUB
169 1B   SUB Instructions STATIC
170 a02    WINDOW 2,"How to play the game",(0,130)-(311,197),0,2
171 GY     WINDOW OUTPUT 2
172 c3     RESTORE
173 8t     READ pages
174 Fd     FOR i=1 TO pages
175 wn4      COLOR 13,7
176 OU       CLS
177 uq       READ lines
178 UB       PRINT " "
179 Zh       FOR Ycor=1 TO lines
```

Listing 1. »Patience« ist eine Herausforderung an Ihre Geduld. Bitte beachten Sie die Eingabehinweise auf Seite 159.

```
180 aF6       READ Ask$
181 OU        PRINT " ";Ask$
182 164     NEXT
183 Ec      COLOR 0,7
184 78      LOCATE 8,1:PRINT i;"of";pages;"- Press a key for next
            age";
185 fI      GetA
186 5A2   NEXT
187 Dt    WINDOW CLOSE 2
188 Rn    WINDOW OUTPUT 1
189 570 END SUB
190 7P  SUB PlayCards STATIC
191 hm2   Solved=0
192 j9    WHILE (Solved<>4)
193 jo4     Solved=0
194 oR      GetA
195 GK      DisplayMessageEnd
196 mE      From=VAL(Ask$)
197 Sq      IF Ask$="i" THEN
198 pd6       Instructions
199 Qh4     ELSEIF Ask$="q" OR Ask$="Q" THEN
200 zA6       OK=0
201 bS        QuitIt OK
202 CE        IF OK=1 THEN EXIT SUB
203 cq4     ELSEIF Ask$="D" OR Ask$="d" OR Ask$="D" THEN
204 nT6       DealIt
205 pY4     ELSEIF Ask$=CHR$(13) OR Ask$="s" OR Ask$="S" THEN
206 H16       StackIt
207 Er4     ELSEIF (From>=1 AND From<=7) THEN
208 Ta6       CheckIt From,OK
209 yh4     ELSE
210 KD      END IF
211 Lh      CheckEnd Solved
212 rf2   WEND
213 TV0 END SUB
214 LE  SUB QuitIt(OK) STATIC
215 X02   Message$="QUIT?  REALLY??  Y/N"
216 xQ    DisplayMessage 7
217 Bo    GetA
218 qo    IF Ask$="y" OR Ask$="Y" THEN
219 SC4     SYSTEM
220 9s2   ELSE
221 Pb4     OK=1
222 hl      DisplayMessageEnd
223 XQ2   END IF
224 eg0 END SUB
225 Xm  SUB CheckIt(Column,OK) STATIC
226 hk2   CheckColor LEFT$(Deal$,1),RIGHT$(FaceUp$(Column),1),OK
227 1Z    IF OK=0 THEN CALL CheckKing(LEFT$(Deal$,1),RIGHT$(FaceUp$
          (Column),1),OK)
228 Kh    IF OK THEN
229 SS4     PlayIt Column
230 J22   ELSE
231 P24     GetA
232 s0      OnTo=VAL(Ask$)
233 o7      IF OnTo <> Column THEN
234 jK      IF (OnTo>=1 AND OnTo<=7) THEN
235 zL6       CheckColor LEFT$(FaceUp$(Column),1),RIGHT$(FaceUp$(On
              To),1),OK
236 H1        IF OK=0 THEN CALL CheckKing(LEFT$(FaceUp$(Column),1),
              RIGHT$(FaceUp$(OnTo),1),OK)
237 8i        IF OK=0 THEN
238 068         SWAP OnTo,Column
239 3P          CheckColor LEFT$(FaceUp$(Column),1),RIGHT$(FaceUp$(
                OnTo),1),OK
240 L5          IF OK=0 THEN CALL CheckKing(LEFT$(FaceUp$(Column),1
                ),RIGHT$(FaceUp$(OnTo),1),OK)
241 pi6       END IF
242 Yv        IF OK THEN
243 7f8         MoveIt Column,OnTo
244 XG6       ELSE
245 e08         Message$="Move Error:"+STR$(OnTo)+" -> "+STR$(Colum
                n)
246 Ru          DisplayMessage 7
247 fI          GetA
248 7B          DisplayMessageEnd
249 xq6       END IF
250 yr4     END IF
251 zs      END IF
252 0t2   END IF
253 790 END SUB
254 T2  SUB DisplayMessageEnd STATIC
255 Jz4     WINDOW CLOSE 2
256 AC0 END SUB
257 E0  SUB DisplayMessage(Colour) STATIC
258 Y12   WINDOW 2,"Information",(64,168)-((LEN(Message$)+10)*8,192
          ),0,2
259 gy    WINDOW OUTPUT 2
260 x2    COLOR Colour,Colour
261 p6    PATTERN ,Empty
262 Pt    LINE(0,0)-(311,197),0,bf
263 dZ    COLOR 13,Colour
264 WU    LOCATE 2,2:PRINT Message$;
265 g2    WINDOW OUTPUT 1
266 KM0 END SUB
267 Ih  SUB MoveIt(From,OnTo) STATIC
268 xr2   Dummy2=LEN(FaceUp$(OnTo))+1
269 Rl    FaceUp$(OnTo)=FaceUp$(OnTo)+FaceUp$(From)
270 Ul    FaceUp$(From)=""
271 Ef    IF LEN(Back$(From))>0 THEN
272 Cu4     FaceUp$(From)=CHR$(ASC(RIGHT$(Back$(From),1)) XOR 128)
273 Q2      Back$(From)=LEFT$(Back$(From),LEN(Back$(From))-1)
274 MF2   END IF
275 Pk    COLOR 2,2
276 Bt    LINE((From-1)*24+70,LEN(Back$(From))*8+14)-((From-1)*24+9
          0,190),2,bf
277 i0    PlaceCard (From-1)*3+8,LEN(Back$(From))+1,LEFT$(FaceUp$(F
          rom),1)
278 y8    Dummy1=LEN(Back$(OnTo))
279 ac    FOR Row=Dummy2 TO LEN(FaceUp$(OnTo))
280 Xq4     Xcor=OnTo:Ycor=Row+Dummy1:IF Ycor<1 THEN Ycor=1
281 Te      PlaceCard (Xcor-1)*3+8,Ycor,MID$(FaceUp$(OnTo),Row,1)
282 di2   NEXT
283 bd0 END SUB
284 7J  SUB CheckColor(S$,P$,OK) STATIC
285 MX2   OK=0
286 CU    From=ASC(S$+CHR$(0)): OnTo=ASC(P$+CHR$(0))
287 Yk    IF (((From AND 15)+1) = (OnTo AND 15)) THEN
288 DF4     IF ((From AND 32) <> (OnTo AND 32)) THEN OK=1
289 bU2   END IF
290 ik0 END SUB
291 h7  SUB CheckKing(S$,P$,OK) STATIC
292 Vs2   Dummy1=ASC(S$+CHR$(0))
293 8X    IF (Dummy1 AND 15)=13 THEN
294 qC4     IF P$="" THEN OK=1
295 ha2   END IF
296 oq0 END SUB
297 j7  SUB CheckAce(P$,S$,OK) STATIC
298 Zk2   OK=0
299 Gn    From=ASC(P$+CHR$(0)): OnTo=ASC(S$+CHR$(0))
300 7Y    IF ((From AND 15) = ((OnTo AND 15)+1)) THEN
301 ZE4     IF ((From AND 48) = (OnTo AND 48)) THEN OK=1
302 oh2   END IF
303 Vv    IF (From AND 15)=1 THEN
304 Fe4     IF S$="" THEN OK=1
305 rk2   END IF
306 y00 END SUB
307 nj  SUB StackIt STATIC
308 EV2   Message$="WAIT - Stacking"
309 Mm    DisplayMessage 4
310 o9    StackItOK=1
311 o6    WHILE (StackItOK=1)
312 o84     StackItOK=0
313 Jg      FOR i=1 TO 4: REM === check Stack(i) ===
314 026       CheckAce LEFT$(Deal$,1),LEFT$(Stack$(i),1),OK
315 j6        IF OK THEN
316 xT8         StackItOK=OK
317 qN          Stack$(i)=LEFT$(Deal$,1)+Stack$(i)
318 Fe          EatDeal
319 Rr          DisplayStack i
320 6z6       END IF
321 m2        FOR k=1 TO 7
322 YY8         CheckAce RIGHT$(FaceUp$(k),1),LEFT$(Stack$(i),1),OK
323 rE          IF OK THEN
324 5bA           StackItOK=OK
325 PZ            Stack$(i)=RIGHT$(FaceUp$(k),1)+Stack$(i)
326 Qz            FaceUp$(k)=LEFT$(FaceUp$(k),LEN(FaceUp$(k))-1)
327 WB            IF LEN(FaceUp$(k))=0 THEN
328 HZC             IF LEN(Back$(k))>0 THEN
329 v4E               FaceUp$(k)=CHR$(ASC(RIGHT$(Back$(k),1)) XOR 1
                      28)
330 I1                Back$(k)=LEFT$(Back$(k),LEN(Back$(k))-1)
331 HAC             END IF
332 IBA           END IF
```

```
333 Lg            COLOR 2,2
334 dk            Dummy1=LEN(Back$(k))+LEN(FaceUp$(k))
335 5g            LINE((k-1)*24+70,Dummy1*8+14)-((k-1)*24+90,190),2
                  ,bf
336 ua            PlaceCard (k-1)*3+8,Dummy1,RIGHT$(FaceUp$(k),1)
337 j9            DisplayStack i
338 OH8         END IF
339 Yd6       NEXT
340 Ze4     NEXT
341 wk2   WEND
342 dh    DisplayMessageEnd
343 hU    EmptyBuffer
344 ac0 END SUB
345 te  SUB PlayIt(Column) STATIC
346 2m2   FaceUp$(Column)=FaceUp$(Column)+LEFT$(Deal$,1)
347 FY    Dummy1=LEN(Back$(Column))
348 N2    Dummy2=LEN(FaceUp$(Column))
349 HH    PlaceCard (Column-1)*3+8,Dummy1+Dummy2,RIGHT$(FaceUp$(Col
          umn),1)
350 lA    EatDeal
351 hj0 END SUB
352 3c  SUB EatDeal STATIC
353 zM2   Deal$=RIGHT$(Deal$,(LEN(Deal$)-1))
354 bI    DisplayDeal
355 lnO END SUB
356 kt  SUB PlaceCard(Xcor,Ycor,Card$) STATIC
357 qa2   BitCard=ASC(Card$+CHR$(0))
358 tP    IF Card$="" THEN EXIT SUB
359 Hm    Xcor=(Xcor)*8
360 Nu    Ycor=(Ycor)*8
361 Eg    COLOR 6,5
362 Sj    PATTERN ,Empty
363 VR    AREA(Xcor+6,Ycor+8):AREA(Xcor+8,Ycor+6)
364 cV    AREA(Xcor+24,Ycor+6):AREA(Xcor+26,Ycor+8)
365 Gb    AREA(Xcor+26,Ycor+30):AREA(Xcor+24,Ycor+32)
366 9o    AREA(Xcor+8,Ycor+32):AREA(Xcor+6,Ycor+30)
367 QW    AREAFILL
368 Np    COLOR 5,6
369 kL    IF (BitCard AND 128) THEN
370 PA4     COLOR 15,9
371 jv      PATTERN ,CardBack
372 kD      AREA(Xcor+8,Ycor+8):AREA(Xcor+24,Ycor+8)
373 QC      AREA(Xcor+24,Ycor+30):AREA(Xcor+8,Ycor+30)
374 Xd      AREAFILL
375 eN2   ELSE
376 gx4     PATTERN ,Empty
377 vv      AREA(Xcor+7,Ycor+9):AREA(Xcor+9,Ycor+7)
378 re      AREA(Xcor+22,Ycor+7):AREA(Xcor+25,Ycor+9)
379 6n      AREA(Xcor+25,Ycor+29):AREA(Xcor+23,Ycor+31)
380 AF      AREA(Xcor+9,Ycor+31):AREA(Xcor+7,Ycor+29)
381 ek      AREAFILL
382 9B      BitCard=(BitCard XOR 128)
383 r3      IF (BitCard AND 48)=0 THEN
384 NP6       COLOR 8:PATTERN ,Clubs
385 N14     ELSEIF (BitCard AND 48)=32 THEN
386 xL6       COLOR 7:PATTERN ,Diamonds
387 hN4     ELSEIF (BitCard AND 48)=48 THEN
388 816       COLOR 7:PATTERN ,Hearts
389 sb4     ELSE
390 dQ6       COLOR 8:PATTERN ,Spades
391 F84     END IF
392 z7      LINE (Xcor+16,Ycor+7)-(Xcor+23,Ycor+15),,bf
393 Wv      LINE (Xcor+9,Ycor+23)-(Xcor+16,Ycor+31),,bf
394 XT      BitCard=(BitCard AND 15)
395 hW      IF BitCard<1 THEN BitCard=1
396 1f      IF BitCard=10 THEN
397 Ws6       PATTERN ,Ten
398 4e        LINE (Xcor+9,Ycor+7)-(Xcor+16,Ycor+15),,bf
399 fq        LINE (Xcor+16,Ycor+23)-(Xcor+23,Ycor+31),,bf
400 3m4     ELSE
401 DO6       LOCATE Ycor\8+2,Xcor\8+2: PRINT MID$("A23456789TJQK",
              BitCard,1);
402 VZ        LOCATE Ycor\8+4,Xcor\8+3: PRINT MID$("A23456789TJQK",
              BitCard,1);
403 RK4     END IF
404 SL2   END IF
405 Zb0 END SUB
406 1y  SUB DimVariables STATIC
407 M82   DEFINT a-z
408 Kp    DIM SHARED Stack$(4), Back$(7), FaceUp$(7)
409 2j    DIM SHARED Ten(7), CardBack(7), Empty(7)
410 4B    DIM SHARED Hearts(7), Spades(7), Diamonds(7), Clubs(7)
411 GT    DIM SHARED Deal$, Ask$, Message$
412 h2    DIM SHARED succes%,failure%
413 hj0 END SUB
414 ev  SUB InitVariables STATIC
415 4J2   Deal$=""
416 4R    FOR i=0 TO 7
417 Yj4     Empty(i)=&H0
418 pu2   NEXT
419 sx    Hearts(0)=&H3636:Spades(0)=&H808 :Diamonds(0)=&H808 :Club
          s(0)=&H808
420 8y    Hearts(1)=&H7F7F:Spades(1)=&H1C1C:Diamonds(1)=&H1C1C:Club
          s(1)=&H1C1C
421 R0    Hearts(2)=&H7F7F:Spades(2)=&H3E3E:Diamonds(2)=&H3E3E:Club
          s(2)=&H2A2A
422 Ha    Hearts(3)=&H7F7F:Spades(3)=&H7F7F:Diamonds(3)=&H7F7F:Club
          s(3)=&H7F7F
423 4R    Hearts(4)=&H3E3E:Spades(4)=&H7F7F:Diamonds(4)=&H3E3E:Club
          s(4)=&H2A2A
424 Yn    Hearts(5)=&H1C1C:Spades(5)=&H7F7F:Diamonds(5)=&H1C1C:Club
          s(5)=&H808
425 M4    Hearts(6)=&H808 :Spades(6)=&H3636:Diamonds(6)=&H808 :Club
          s(6)=&H1C1C
426 y6    Hearts(7)=&H0   :Spades(7)=&H0   :Diamonds(7)=&H0   :Club
          s(7)=&H0
427 Iq    Ten(0)=&H2626   :CardBack(0)=&HC5C5
428 WV    Ten(1)=&H6F6F   :CardBack(1)=&HE3E3
429 OA    Ten(2)=&H2929   :CardBack(2)=&HD1D1
430 88    Ten(3)=&H2929   :CardBack(3)=&H8888
431 KT    Ten(4)=&H2929   :CardBack(4)=&H1D1D
432 Ah    Ten(5)=&H2F2F   :CardBack(5)=&H3E3E
433 iq    Ten(6)=&H2626   :CardBack(6)=&H5C5C
434 I8    Ten(7)=&H0      :CardBack(7)=&H8888
435 350 END SUB
436 Al  SUB DealIt STATIC
437 WM2   IF LEN(Deal$)<2 THEN EXIT SUB
438 BM    PALETTE 3,.7,0,0 :REM dealfield
439 1P    IF LEN(Deal$)>9 THEN
440 cf4     Deal$=RIGHT$(Deal$,3)+LEFT$(Deal$,(LEN(Deal$)-3))
441 1R2   ELSE
442 MV4     Deal$=RIGHT$(Deal$,1)+LEFT$(Deal$,(LEN(Deal$)-1))
443 5y2   END IF
444 3k    DisplayDeal
445 CL    PALETTE 3,.5,0,0 :REM dealfield
446 EG0 END SUB
447 7b  SUB EmptyBuffer STATIC
448 EH2   WHILE INKEY$<>""
449 gU    WEND
450 IK0 END SUB
451 8A  SUB GetA STATIC
452 Vq2   Ask$=""
453 LD    WHILE Ask$=""
454 Kg4     Ask$=INKEY$
455 ma2   WEND
456 OQ0 END SUB
457 Iz  ' ------------------------------------------
458 Y3  DATA 8,3," ","The object of the game is","to move all cards
        onto the Stack."
459 4E  DATA 5,"Active keys are:"
460 J4  DATA "                7"
461 3C  DATA "(numeric keypad) 4 5 6"
462 oR  DATA "                1 2 3 Enter-(Stack)"
463 gl  DATA "                0----------(Deal)"
464 mc  DATA 5,"Cards must be played in","descending order."," "
465 W9  DATA "The sequence is","RED ON BLACK  -  BLACK ON RED."
466 f3  DATA 5,"To PLAY cards from the Deal-Stack"
467 nM  DATA "press the reciving column number."
468 wG  DATA " ","To DEAL a new Card (every 3rd Card)","press '0' (
        zero)."
469 Xm  DATA 4," ","To MOVE cards or columns press","Source-Column
        number and then"
470 5X  DATA "Target-Column number."
471 zy  DATA 3," ","When a column is empty,","only a King may be mo
        ved there."
472 rn  DATA 4," ","Pressing 'ENTER' stacks cards from"
473 kt  DATA "Columns and Dealer-Stack into","stacking-area in Ace
        to King order."
474 iA  DATA 3," ","A new game may be dealt at","any time by pressi
        ng 'Q'uit"
(C) 1989 M&T
```

Listing 1. (Schluß)

Ping-pong pur

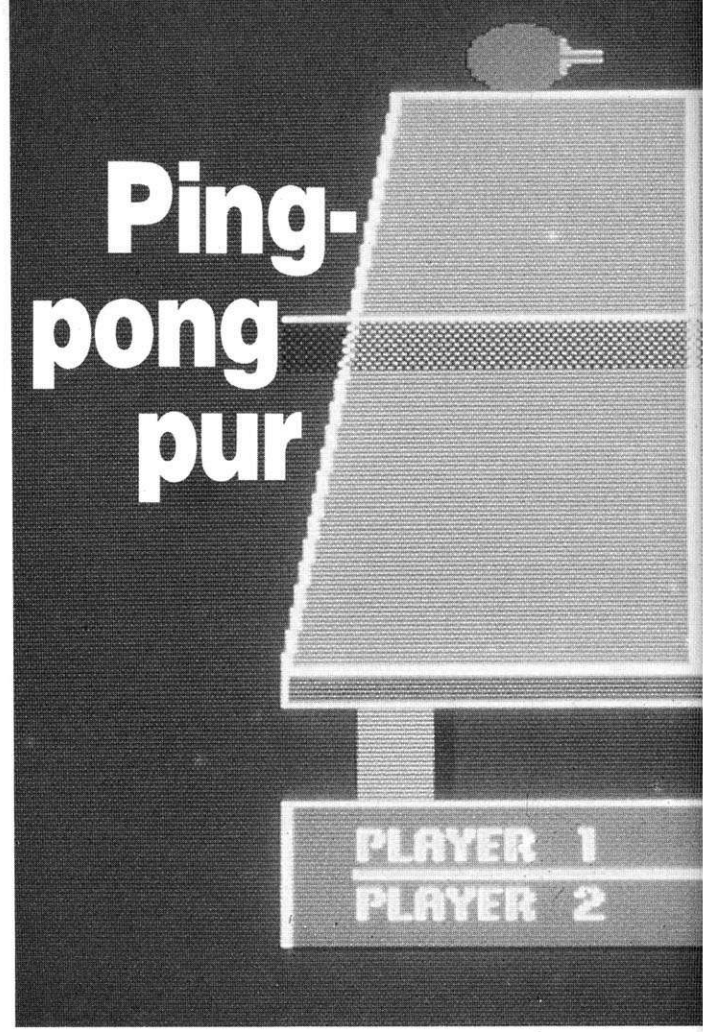


Bild 1. Sport, Spiel, Spannung mit der dreidimensionalen Tis

Ping Pong« (Listing 1) füllt eine Lücke im Spieleangebot für den Amiga. Es ist nämlich unseres Wissens nach die erste Tischtennis-Simulation für diesen Computer. Die Programmierung in Assembler verleiht dem Programm eine rasante Geschwindigkeit und bringt Schwung ins Spiel. Die dreidimensionale Grafik und der »klassische« Pingpong-Sound tragen ihren Teil dazu bei, daß das Spiel absolut realitätsnah ist und Sie an den Joystick fesseln wird (Bild 1).

»Ping Pong« ist außerdem eines jener wenigen Spiele, bei denen zwei Spieler gleichzeitig gegeneinander antreten können.

Action zu zweit

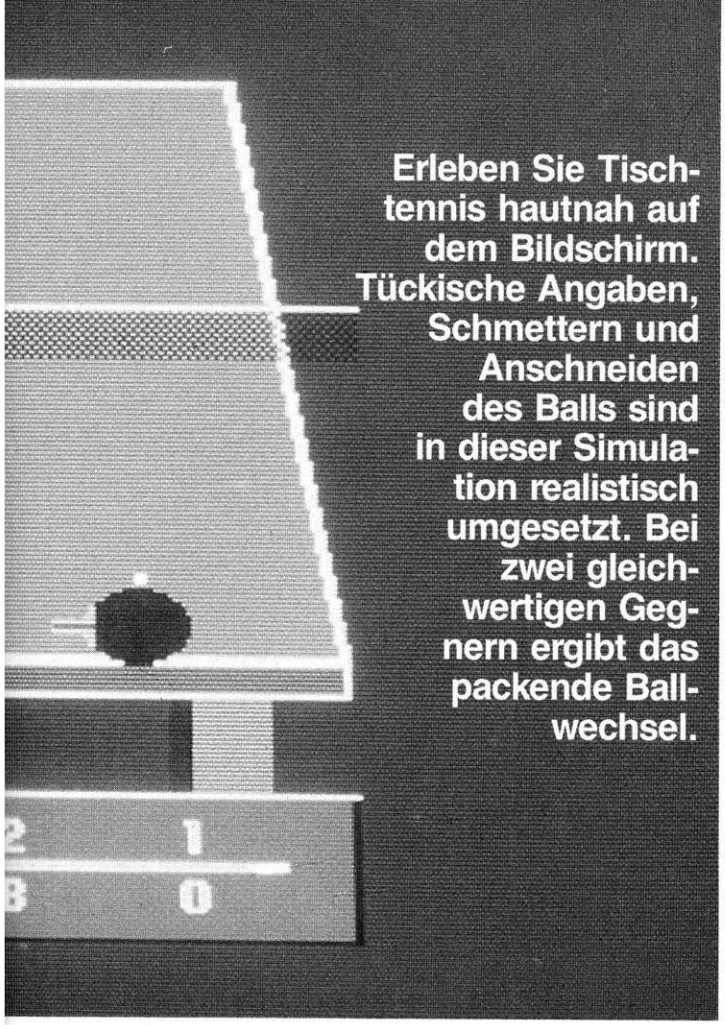
Alles, was Sie tun müssen, um den kleinen weißen Ball auch auf Ihrem Bildschirm zum Leben zu erwecken, ist, das Listing mit dem Checksummer einzugeben. Da das Programm in Assembler geschrieben wurde, wir aber möglichst vielen Lesern ermöglichen wollen, in den Genuß von »Ping Pong« zu kommen, haben wir uns entschlossen, das Listing als Basic-Lader abzudrucken. Es kann also unter Beachtung der Eingabehinweise (siehe Kasten) wie jedes andere Basic-Programm eingetippt werden. Nach dem ersten Start von »PingPong_Gen« wird das lauffähige Programm »Ping Pong« automatisch auf Diskette erzeugt (siehe Eingabehinweise im Kasten).

Das Spiel selbst läuft nach den üblichen Tischtennisregeln ab. Sieger ist der Spieler, der zwei Sätze gewinnt. Ein Satz ist gewonnen, wenn man 21 Punkte erreicht und mindestens zwei Punkte mehr auf seinem Konto als der Gegner hat.

Das Match beginnt mit der Angabe des Spielers im Vordergrund. Wenn er den Knopf an seinem Joystick drückt, fliegt der »simulierte Zelluloidball« in die Höhe, um sich dann gleich wieder dem Tischtennis-Schläger und dem Boden zu nähern. Der Joystick-Knopf muß nun ein weiteres Mal betätigt werden, um den Ball in die gegnerische Hälfte zu befördern — wichtig ist dabei der richtige Augenblick. Nach einigem Experimentie-

Programmname:	PingPong_gen
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic
Bemerkung:	erzeugt lauffähiges Programm auf eingelegter Diskette (s. Kasten)

```
Programmautor: S. Ören
----------------------
 1 0m0 REM Generiert lauffähiges Programm
 2 Kq  REM >>PingPong<< auf der Diske
       tte
 3 cU  REM im Laufwerk df0:
 4 c1  CLS
 5 oJ  OPEN "df0:pingpong" FOR OUTPUT AS
       1
 6 DU  READ anz
 7 qc  FOR i=1 TO anz
 8 5p1  READ h$
 9 OD2   wert1=ASC(LEFT$(h$,1))
10 dR    IF wert1>64 THEN wert1=wert1-87
          ELSE wert1=wert1-48
11 HK    wert1=wert1*16
12 9e    wert2=ASC(RIGHT$(h$,1))
13 yr    IF wert2>64 THEN wert2=wert2-87
          ELSE wert2=wert2-48
14 Rk    wert=wert1+wert2
15 BI    PRINT #1,CHR$(wert);
16 LQ0 NEXT
17 5p  CLOSE 1
18 2x  END
19 0e  Werte:
20 GE  DATA 9496
21 rj  DATA 00,00,03,f3,00,00,00,00,00,00
22 Se  DATA 00,01,00,00,00,00,00,00,00,00
23 aR  DATA 00,00,07,e6,00,00,03,e9,00,00
24 h2  DATA 07,e6,2c,78,00,04,20,3c,00,00
25 zV  DATA 78,00,22,3c,00,03,00,02,4e,ae
26 gS  DATA ff,3a,67,00,12,f6,23,c0,00,00
27 r0  DATA 1f,30,20,3c,00,00,28,00,22,3c
28 Bn  DATA 00,03,00,02,4e,ae,ff,3a,67,00
29 6m  DATA 12,dc,23,c0,00,00,1f,3c,2c,7c
30 or  DATA 00,00,1a,04,20,3c,00,00,18,c8
31 eY  DATA 22,3c,00,00,19,10,24,3c,00,00
32 q3  DATA 19,58,26,3c,00,00,19,64,28,3c
33 gM  DATA 00,00,19,fc,2c,3c,00,00,19,70
34 6x  DATA 2e,3c,00,00,19,b8,3d,40,00,06
35 C0  DATA 48,40,3d,40,00,02,3d,41,00,0e
36 xk  DATA 48,41,3d,41,00,0a,3d,42,00,16
37 iU  DATA 48,42,3d,42,00,12,3d,43,00,1e
38 TE  DATA 48,43,3d,43,00,1a,3d,44,00,26
39 6z  DATA 3d,44,00,2e,48,44,3d,44,00,22
40 TJ  DATA 3d,44,00,2a,3d,46,00,36,48,46
41 Q6  DATA 3d,46,00,32,3d,47,00,3e,48,47
42 53  DATA 3d,47,00,3a,20,3a,1e,78,22,00
43 31  DATA 24,01,06,81,00,00,28,00,06,82
44 NY  DATA 00,00,50,00,23,c1,00,00,1f,34
45 iF  DATA 23,c2,00,00,1f,38,26,3a,1e,64
46 Dm  DATA 48,40,3d,40,00,42,48,40,3d,40
47 hM  DATA 00,46,48,41,3d,41,00,4a,48,41
48 73  DATA 3d,41,00,4e,48,42,3d,42,00,52
49 Bp  DATA 48,42,3d,42,00,56,48,43,3d,43
50 iJ  DATA 00,5a,48,43,3d,43,00,5e,3c,3c
51 Qr  DATA 00,40,06,80,00,00,06,e8,20,40
52 4x  DATA 20,3c,00,00,00,04,22,3c,00,00
53 xV  DATA 00,09,61,00,16,1e,91,fc,00,00
54 vY  DATA 00,02,20,3c,00,00,00,04,22,3c
55 RX  DATA 00,00,00,0f,61,00,16,08,22,48
56 HT  DATA e2,5e,20,3c,00,00,00,06,61,00
57 o0  DATA 16,2e,20,49,e2,5e,d1,fc,00,00
58 4m  DATA 50,00,20,3c,00,00,00,05,61,00
59 hE  DATA 16,1a,22,48,d1,fc,00,00,00,2a
60 xH  DATA 20,3c,00,00,00,14,61,00,16,08
61 G9  DATA ea,5e,20,49,d1,fc,00,00,00,2c
62 7E  DATA 20,3c,00,00,00,14,61,00,15,f4
63 1G  DATA 20,49,91,fc,00,00,27,d4,20,3c
64 6d  DATA 00,00,00,14,3c,3c,22,00,61,00
65 wc  DATA 15,de,20,7a,1d,94,d1,fc,00,00
66 4f  DATA 06,fa,3c,3c,08,00,20,3c,00,00
67 Tb  DATA 00,04,22,3c,00,00,00,0b,61,00
68 zi  DATA 15,a6,d1,fc,00,00,00,02,20,3c
69 IO  DATA 00,00,00,04,22,3c,00,00,00,0d
70 9d  DATA 61,00,15,90,22,48,e3,5e,20,3c
71 HU  DATA 00,00,00,06,61,00,15,9c,20,49
72 EL  DATA d1,fc,00,00,50,00,e3,5e,20,3c
73 dj  DATA 00,00,00,05,61,00,15,88,3a,06
74 Mm  DATA eb,5e,20,3c,00,00,00,14,22,48
75 wv  DATA d1,fc,00,00,00,24,61,00,15,72
76 0G  DATA 20,49,91,fc,00,00,27,dc,20,3c
77 6g  DATA 00,00,00,14,3c,3c,01,10,61,00
78 MA  DATA 15,5c,3c,05,20,49,d1,fc,00,00
79 yF  DATA 00,26,20,3c,00,00,00,14,61,00
80 3r  DATA 15,48,20,7a,1d,02,3c,3c,00,10
81 VB  DATA d1,fc,00,00,06,e8,20,3c,00,00
82 iq  DATA 00,04,22,3c,00,00,00,0b,61,00
83 Vb  DATA 14,f6,91,fc,00,00,00,02,20,3c
84 Xd  DATA 00,00,00,04,22,3c,00,00,00,0d
85 hk  DATA 61,00,14,e0,91,fc,00,00,00,28
86 6w  DATA 30,bc,00,00,91,fc,00,00,00,28
87 St  DATA 30,bc,00,00,20,7a,1c,ba,d1,fc
88 9K  DATA 00,00,06,fa,3c,3c,20,00,20,3c
89 ci  DATA 00,00,00,04,22,3c,00,00,00,0d
90 0s  DATA 61,00,14,c8,d1,fc,00,00,00,02
91 ha  DATA 20,3c,00,00,00,04,22,3c,00,00
92 VL  DATA 00,0b,61,00,14,b2,91,fc,00,00
93 nz  DATA 00,28,30,bc,00,00,91,fc,00,00
94 6M  DATA 00,28,30,bc,00,00,20,7a,1c,72
95 R5  DATA d1,fc,00,00,07,18,3c,3c,00,22
96 YO  DATA 20,3c,00,00,00,7e,61,00,14,a0
97 ZC  DATA 20,7a,1c,5a,31,7c,00,00,06,e8
98 yj  DATA 31,7c,00,00,06,fa,20,7a,1c,46
99 sc  DATA 20,3c,00,00,77,fe,42,83,61,00
100 j5 DATA 15,9e,22,3c,00,00,00,0b,20,7a
101 8F DATA 1c,30,d1,fc,00,00,0e,de,3c,3c
102 Eu DATA ff,ff,20,3c,00,00,00,00,61,00
103 qa DATA 14,70,91,fc,00,00,00,26,51,c9
104 20 DATA ff,ea,22,3c,00,00,00,0b,20,7a
105 tf DATA 1c,0c,d1,fc,00,00,0e,de,3c,3c
106 Mu DATA 00,00,20,3c,00,00,00,00,61,00
107 Rc DATA 14,48,91,fc,00,00,00,26,51,c9
108 64 DATA ff,ea,22,3c,00,00,00,0b,20,7a
109 jg DATA 1b,ec,d1,fc,00,00,0f,06,3c,3c
```



s-Simulation. Zwei Spieler treten dabei gegeneinander an.

ren findet man schnell den günstigsten Zeitpunkt heraus.

Schaffen Sie es, den Ball über das Netz zu befördern, ist der Gegner an der Reihe. Dieser muß ebenfalls den Ball mit seinem Schläger treffen und dabei gleichzeitig den Feuerknopf drücken. Mit der Bewegung des Joysticks während des Schlages nach rechts oder links kann der Abprallwinkel des Balls manipuliert werden. So wird das Anschneiden des Balls realisiert. Wird der Joystick nach vorne bewegt, steigert sich die Geschwindigkeit (Schmettern).

Nach jeweils fünf Punkten wechselt das Aufgaberecht. Ist ein Satz beendet, erfolgt ein Seitenwechsel. Spieler 1 führt den braunen Schläger und verwendet den Joystick in Port 1. Der zweite Spieler steuert seinen (roten) Schläger mit dem Joystick in Port 2.

Anfangs ist es sicherlich etwas schwierig, beispielsweise das Anschneiden des Balls geschickt einzusetzen. Nach kurzer Zeit werden Sie den Ball immer besser unter Kontrolle haben und Ball und Gegner beherrschen.

(Martin Jobst/rs)

Eingabehinweise

Starten Sie zur Generierung des lauffähigen Programms Amiga-Basic, das sich auf der, jedem Amiga beiliegenden, »Extras«-Diskette befindet. Laden Sie anschließend »PingPong_Gen« und entfernen den Schreibschutz auf der Diskette im eingebauten Laufwerk. Starten Sie nun die Programmausführung mit RUN. Jetzt leuchtet die LED des Laufwerks df0:, das Programm »Ping Pong« wird auf die eingelegte Diskette geschrieben. Nachdem sich Amiga-Basic mit »O.K.« meldet, können Sie mit QUIT den Basic-Interpreter verlassen.

Das auf diese Weise erzeugte Programm »Ping Pong« kann von jetzt an vom CLI aus gestartet werden. Wenn Sie nicht wissen, wie man Programme vom CLI oder das CLI selbst aufruft, schauen Sie bitte in Ihrem Amiga-Handbuch nach.

Besitzer einer Speichererweiterung oder eines Amiga 2000 sollten beachten, daß »Ping Pong« nur ohne Zusatzspeicher lauffähig ist. Dieser muß also vor dem Start unbedingt mit »NoFastMem« oder hardwaremäßig abgeschaltet werden.

```
110 s8  DATA 55,55,20,3c,00,00,00,0a,61,00
111 zM  DATA 14,20,91,fc,00,00,01,b6,51,c9
112 HH  DATA ff,ea,20,3c,00,00,00,1e,20,7a
113 vB  DATA 1b,b8,d1,fc,00,00,1f,be,3c,3c
114 ga  DATA c0,00,61,00,13,f0,22,3c,00,00
115 Oh  DATA 00,0b,20,7a,1b,ac,d1,fc,00,00
116 qG  DATA 1f,be,3c,3c,ff,ff,20,3c,00,00
117 Cr  DATA 00,1e,61,00,13,e0,91,fc,00,00
118 7O  DATA 04,d6,51,c9,ff,ea,22,3c,00,00
119 fU  DATA 00,0b,20,7a,1b,80,d1,fc,00,00
120 uK  DATA 1f,be,3c,3c,ff,ff,20,3c,00,00
121 YI  DATA 00,1e,61,00,13,b8,91,fc,00,00
122 z0  DATA 04,d6,51,c9,ff,ea,20,3c,00,00
123 O7  DATA 00,1e,20,7a,1b,58,d1,fc,00,00
124 aW  DATA 1f,d4,3c,3c,ff,fc,61,00,13,88
125 zj  DATA 22,3c,00,00,00,0b,20,7a,1b,38
126 T3  DATA d1,fc,00,00,1f,96,3c,3c,ff,ff
127 AT  DATA 20,3c,00,00,00,00,61,00,13,78
128 w5  DATA 91,fc,00,00,00,26,51,c9,ff,ea
129 bF  DATA 22,3c,00,00,00,0b,20,7a,1b,14
130 ZM  DATA d1,fc,00,00,24,96,3c,3c,ff,ff
131 Sb  DATA 20,3c,00,00,00,00,61,00,13,50
132 09  DATA 91,fc,00,00,00,26,51,c9,ff,ea
133 nM  DATA 22,3c,00,00,00,09,20,7a,1a,ec
134 Yv  DATA d1,fc,00,00,21,f0,3c,3c,ff,ff
135 1G  DATA 20,3c,00,00,00,01,61,00,13,28
136 PL  DATA 91,fc,00,00,00,4e,51,c9,ff,ea
137 JX  DATA 20,7a,1a,ca,d1,fc,00,00,20,88
138 kj  DATA 20,3c,00,00,00,06,22,7c,00,00
139 Mj  DATA 1a,68,61,00,13,14,20,7a,1a,b0
140 SG  DATA d1,fc,00,00,22,90,20,3c,00,00
141 ef  DATA 00,06,22,7c,00,00,1a,68,61,00
142 fZ  DATA 12,fa,20,7a,1a,96,d1,fc,00,00
143 Ck  DATA 20,8e,20,3c,00,00,00,03,22,7c
144 Ic  DATA 00,00,1a,92,61,00,13,06,20,7a
145 NL  DATA 1a,7c,d1,fc,00,00,22,96,20,3c
146 ec  DATA 00,00,00,03,22,7c,00,00,1a,9a
147 S8  DATA 61,00,12,ec,13,fc,00,15,00,00
148 MA  DATA 1f,90,61,00,0e,80,33,fc,01,2c
149 eU  DATA 00,00,1f,58,13,fc,00,44,00,00
150 6K  DATA 1f,81,33,fc,00,b4,00,00,1f,5c
151 fy  DATA 13,fc,00,c6,00,00,1f,80,2c,78
152 VM  DATA 00,04,4e,ae,ff,7c,4b,f9,00,df
153 fe  DATA f0,00,3b,7c,03,ef,00,96,2b,7c
154 CL  DATA 00,00,1a,f2,00,a0,2b,7c,00,00
155 Xc  DATA 1a,e2,00,b0,2b,7c,02,1f,02,5f
156 5e  DATA 00,a4,3b,7c,00,08,00,b4,3b,7c
157 Vi  DATA 00,40,00,a8,2b,7c,01,90,00,40
158 Rh  DATA 00,b6,3b,7c,00,ff,00,9e,2b,7c
159 Us  DATA 00,00,1a,04,00,80,42,6d,00,88
160 vC  DATA 2b,7c,29,81,29,f2,00,8e,2b,7c
161 7F  DATA 00,38,00,d0,00,92,2b,7c,42,00
162 Sj  DATA 00,00,01,00,3b,7c,00,17,01,04
163 Ng  DATA 2b,7c,00,06,0f,ff,01,80,2b,7c
164 vt  DATA 00,00,00,90,01,84,2b,7c,07,77
165 jO  DATA 00,60,01,88,2b,7c,03,33,00,00
166 1C  DATA 01,8c,2b,7c,0f,a6,0f,00,01,a2
167 UE  DATA 3b,7c,0b,40,01,a6,2b,7c,0f,ff
168 8J  DATA 00,00,01,aa,2b,7c,0f,a6,06,00
169 6W  DATA 01,ba,3b,7c,0b,40,01,be,20,7c
170 yu  DATA 00,df,f1,90,20,3c,00,00,00,07
171 d2  DATA 30,bc,00,00,d1,fc,00,00,00,02
172 FU  DATA 51,c8,ff,f4,2b,7c,00,0f,0f,ff
173 Qk  DATA 01,98,3b,7c,0f,f0,01,9c,3b,7c
174 28  DATA 83,a0,00,96,22,3a,19,78,10,3a
175 dd  DATA 19,9d,0c,39,00,01,00,00,1f,93
176 Jx  DATA 67,00,00,0e,34,39,00,df,f0,0a
177 Wq  DATA 4e,f9,00,00,06,04,34,39,00,df
178 gD  DATA f0,0c,08,02,00,01,67,00,00,1e
179 o1  DATA 48,41,0c,41,01,90,62,00,00,12
180 Dx  DATA 48,41,06,81,00,00,55,56,23,c1
181 34  DATA 00,00,1f,58,48,41,48,41,08,02
182 Um  DATA 00,09,67,00,00,1e,48,41,0c,41
183 Um  DATA 00,82,63,00,00,12,48,41,04,81
184 5D  DATA 00,00,55,56,23,c1,00,00,1f,58
185 8P  DATA 48,41,48,41,22,7c,00,00,1f,40
186 uF  DATA 24,7c,00,00,1f,44,12,80,14,80
187 3N  DATA 48,41,08,01,00,00,67,00,00,14
188 A1  DATA 13,7c,00,01,00,03,15,7c,00,01
189 Sp  DATA 00,03,4e,f9,00,00,06,84,13,7c
190 Zx  DATA 00,00,00,03,15,7c,00,00,00,03
191 5J  DATA e2,49,13,41,00,01,06,01,00,08
192 sQ  DATA 15,41,00,01,06,00,00,11,13,40
193 XX  DATA 00,02,04,00,00,01,15,40,00,02
194 vW  DATA 22,3a,18,b8,10,3a,18,d8,0c,39
195 XJ  DATA 00,01,00,00,1f,93,67,00,00,0e
196 we  DATA 34,39,00,df,f0,0c,4e,f9,00,00
197 Zp  DATA 06,c8,34,39,00,df,f0,0a,08,02
198 ym  DATA 00,01,67,00,00,1e,48,41,0c,41
199 g9  DATA 01,90,62,00,00,12,48,41,06,81
200 od  DATA 00,00,55,56,23,c1,00,00,1f,5c
201 Rs  DATA 48,41,48,41,08,02,00,09,67,00
202 5c  DATA 00,1e,48,41,0c,41,00,82,63,00
203 10  DATA 00,12,48,41,04,81,00,00,55,56
204 6Z  DATA 23,c1,00,00,1f,5c,48,41,48,41
205 jY  DATA 22,7c,00,00,1f,50,24,7c,00,00
206 3X  DATA 1f,54,12,80,14,80,48,41,08,01
207 Oi  DATA 00,00,67,00,00,14,13,7c,00,01
208 t6  DATA 00,03,15,7c,00,01,00,03,4e,f9
209 10  DATA 00,00,07,48,13,7c,00,00,00,03
210 8N  DATA 15,7c,00,00,00,03,e2,59,13,41
211 gG  DATA 00,01,50,01,15,41,00,01,06,00
212 F5  DATA 00,11,13,40,00,02,15,40,00,02
213 kk  DATA 0c,39,00,01,00,00,1f,83,67,00
214 cs  DATA 00,c6,0c,39,00,01,00,00,1f,8d
215 1L  DATA 66,00,00,32,0c,39,00,01,00,00
216 BM  DATA 1f,93,67,00,00,14,08,39,00,07
217 Lp  DATA 00,bf,e0,01,66,00,00,68,4e,f9
218 DM  DATA 00,00,07,d2,08,39,00,06,00,bf
219 7C  DATA e0,01,66,00,00,56,4e,f9,00,00
220 rj  DATA 07,d2,0c,39,00,01,00,00,1f,93
221 T2  DATA 67,00,00,14,08,39,00,06,00,bf
222 CH  DATA e0,01,66,00,00,38,4e,f9,00,00
223 sQ  DATA 07,d2,08,39,00,07,00,bf,e0,01
224 IU  DATA 66,00,00,26,23,fc,00,00,14,00
225 eD  DATA 00,00,1f,74,23,fc,00,20,00,00
226 hI  DATA 00,00,1f,78,13,fc,00,01,00,00
227 vO  DATA 1f,83,13,fc,00,01,00,00,1f,82
228 7t  DATA 0c,39,00,01,00,00,1f,8d,66,00
229 UK  DATA 00,1c,36,3a,17,58,06,43,00,12
230 Zu  DATA 61,00,0f,1e,33,fc,d5,00,00,00
```

Listing 1. Geben Sie »PingPong_Gen« bitte mit dem Checksummer (Seite 159) ein

```
231 on  DATA 1f,72,4e,f9,00,00,08,30,36,3a
232 3K  DATA 17,3a,06,43,00,09,61,00,0f,04
233 yS  DATA 33,fc,55,00,00,00,1f,72,20,3a
234 ng  DATA 17,42,91,b9,00,00,1f,78,20,3a
235 QS  DATA 17,3c,d1,b9,00,00,1f,7c,30,3a
236 DK  DATA 17,1e,d1,79,00,00,1f,72,18,3a
237 uV  DATA 17,22,20,3a,17,0c,d1,b9,00,00
238 pI  DATA 1f,66,20,3a,17,08,d1,b9,00,00
239 qh  DATA 1f,6e,36,3a,17,06,0c,39,00,46
240 fF  DATA 00,00,1f,72,62,00,00,12,13,fc
241 J9  DATA 00,01,00,00,1f,86,13,fc,00,01
242 GN  DATA 00,00,1f,8b,0c,39,00,eb,00,00
243 QJ  DATA 1f,72,65,00,00,12,13,fc,00,01
244 5b  DATA 00,00,1f,86,13,fc,00,02,00,00
245 vM  DATA 1f,8b,0c,39,00,3c,00,00,1f,7c
246 lw  DATA 63,00,01,1a,0c,39,00,01,00,00
247 yC  DATA 1f,82,66,00,00,2a,0c,39,00,55
248 6U  DATA 00,00,1f,72,66,00,00,0a,13,fc
249 Vd  DATA 00,01,00,00,1f,86,0c,39,00,d5
250 8W  DATA 00,00,1f,72,66,00,00,0a,13,fc
251 U8  DATA 00,01,00,00,1f,86,23,fc,00,1a
252 SN  DATA 00,00,00,00,1f,78,23,fc,00,00
253 5T  DATA 18,00,00,00,1f,74,0c,39,00,01
254 sz  DATA 00,00,1f,8c,66,00,00,0a,13,fc
255 DE  DATA 00,01,00,00,1f,86,0c,04,00,d6
256 JU  DATA 65,00,00,12,13,fc,00,01,00,00
257 cg  DATA 1f,86,13,fc,00,02,00,00,1f,8b
258 7W  DATA 0c,04,00,53,62,00,00,12,13,fc
259 bR  DATA 00,01,00,00,1f,86,13,fc,00,01
260 4i  DATA 00,00,1f,8b,3a,3c,00,00,3c,3c
261 IK  DATA 00,d6,9c,04,04,46,00,05,0c,46
262 QM  DATA 00,81,62,00,00,08,52,45,4e,fa
263 Z9  DATA ff,f0,3c,3c,00,b0,dc,45,bc,43
264 rr  DATA 65,00,00,0a,13,fc,00,01,00,00
265 Wa  DATA 1f,86,3c,3c,01,6e,9c,45,bc,43
266 kh  DATA 62,00,00,0a,13,fc,00,01,00,00
267 ta  DATA 1f,86,0c,39,00,00,00,00,1f,86
268 OH  DATA 66,00,00,3e,13,fc,00,01,00,00
269 pe  DATA 1f,8a,33,fc,01,90,00,df,f0,a6
270 hJ  DATA 33,fc,82,01,00,df,f0,96,13,fc
271 ch  DATA 00,14,00,00,1f,89,0c,39,00,00
272 gH  DATA 00,00,1f,82,67,00,00,12,13,fc
273 jJ  DATA 00,00,00,00,1f,8a,13,fc,00,00
274 ie  DATA 00,00,1f,82,04,39,00,01,00,00
275 Y6  DATA 1f,89,0c,39,00,00,00,00,1f,89
276 wh  DATA 66,00,00,0a,33,fc,00,01,00,df
277 H5  DATA f0,96,0c,39,00,93,00,00,1f,72
278 S7  DATA 67,00,00,14,0c,39,00,92,00,00
279 go  DATA 1f,72,67,00,00,08,4e,f9,00,00
280 eZ  DATA 0a,38,0c,39,00,0c,00,00,1f,7c
281 4M  DATA 62,00,00,2e,3a,3c,00,00,9a,79
282 fS  DATA 00,00,1f,64,33,c5,00,00,1f,64
283 sp  DATA 23,fc,00,00,30,00,00,00,1f,74
284 71  DATA 23,fc,00,10,00,00,00,00,1f,78
285 sx  DATA 13,fc,00,01,00,00,1f,8c,13,fc
286 nV  DATA 00,ff,00,00,1f,85,1a,04,9a,3a
287 14  DATA 15,38,13,fc,00,01,00,00,1f,84
288 8E  DATA 1c,05,9c,39,00,00,1f,80,20,3c
289 kW  DATA 00,00,00,04,1e,3c,00,00,bc,07
290 T4  DATA 66,00,00,08,13,c7,00,00,1f,85
291 Jg  DATA 52,07,51,c8,ff,f0,20,3c,00,00
292 Ap  DATA 00,04,1e,3c,00,0c,bc,07,66,00
293 ot  DATA 00,08,13,c0,00,00,1f,85,52,07
294 B4  DATA 51,c8,ff,f0,20,3c,00,00,00,04
295 Qe  DATA 1e,3c,00,06,bc,07,66,00,00,0a
296 Z8  DATA 13,fc,00,06,00,00,1f,85,52,07
297 bM  DATA 51,c8,ff,ee,20,3c,00,00,00,01
298 Pb  DATA 1e,3c,00,04,bc,07,66,00,00,0a
299 Y5  DATA 13,fc,00,04,00,00,1f,85,52,07
300 FF  DATA bc,07,66,00,00,0a,13,fc,00,04
301 5q  DATA 00,00,1f,85,5c,07,51,c8,ff,ec
302 ck  DATA 3c,3a,14,92,9c,7a,14,7c,0c,39
303 KU  DATA 00,00,00,00,1f,85,66,00,00,24
304 vP  DATA 0c,06,00,0e,65,00,00,1c,0c,06
305 P5  DATA 00,15,62,00,00,14,13,fc,00,00
306 xQ  DATA 00,00,1f,84,23,fc,00,00,00,00
307 Mw  DATA 00,00,1f,6a,0c,39,00,01,00,00
308 R6  DATA 1f,85,66,00,00,3e,0c,06,00,0b
309 WZ  DATA 65,00,00,36,0c,06,00,18,62,00
310 5M  DATA 00,2e,13,fc,00,00,00,00,1f,84
311 Vc  DATA 0c,06,00,0e,62,00,00,0c,23,fc
312 3n  DATA ff,ff,f9,00,00,00,1f,6a,0c,06
313 9Z  DATA 00,16,65,00,00,0c,23,fc,00,00
314 DI  DATA 07,00,00,00,1f,6a,0c,39,00,02
315 7q  DATA 00,00,1f,85,66,00,00,3e,0c,06
316 xL  DATA 00,0a,65,00,00,36,0c,06,00,19
317 ph  DATA 62,00,00,2e,13,fc,00,00,00,00
318 eE  DATA 1f,84,0c,06,00,0e,62,00,00,0c
319 6r  DATA 23,fc,ff,ff,f9,00,00,00,1f,6a
320 To  DATA 0c,06,00,16,65,00,00,0c,23,fc
321 xl  DATA 00,00,07,00,00,00,1f,6a,0c,39
322 nk  DATA 00,03,00,00,1f,85,66,00,00,50
323 TF  DATA 0c,06,00,0a,65,00,00,48,0c,06
324 yj  DATA 00,1b,62,00,00,40,13,fc,00,00
325 9G  DATA 00,00,1f,84,0c,06,00,0e,62,00
326 mw  DATA 00,0c,23,fc,ff,ff,f9,00,00,00
327 2k  DATA 1f,6a,0c,06,00,16,65,00,00,0c
328 XG  DATA 23,fc,00,00,07,00,00,00,1f,6a
329 02  DATA 0c,06,00,1a,65,00,00,0c,23,fc
330 4c  DATA 00,00,0e,00,00,00,1f,6a,0c,39
331 2v  DATA 00,04,00,00,1f,85,66,00,00,50
332 c0  DATA 0c,06,00,0a,65,00,00,48,0c,06
333 Du  DATA 00,1c,62,00,00,40,13,fc,00,00
334 IP  DATA 00,00,1f,84,0c,06,00,0e,62,00
335 v5  DATA 00,0c,23,fc,ff,ff,f9,00,00,00
336 Bt  DATA 1f,6a,0c,06,00,16,65,00,00,0c
337 gP  DATA 23,fc,00,00,07,00,00,00,1f,6a
338 9B  DATA 0c,06,00,1a,65,00,00,0c,23,fc
339 Dl  DATA 00,00,0e,00,00,00,1f,6a,0c,39
340 N8  DATA 00,06,00,00,1f,85,66,00,00,50
341 lX  DATA 0c,06,00,0a,65,00,00,48,0c,06
342 M3  DATA 00,1c,62,00,00,40,13,fc,00,00
343 RY  DATA 00,00,1f,84,0c,06,00,0e,62,00
344 4E  DATA 00,0c,23,fc,ff,ff,f9,00,00,00
345 K2  DATA 1f,6a,0c,06,00,16,65,00,00,0c
346 pY  DATA 23,fc,00,00,07,00,00,00,1f,6a
347 IK  DATA 0c,06,00,1a,65,00,00,0c,23,fc
348 Mu  DATA 00,00,0e,00,00,00,1f,6a,0c,39
349 xZ  DATA 00,d5,00,00,1f,72,67,00,00,46
350 yd  DATA 0c,39,00,d4,00,00,1f,72,67,00
351 wk  DATA 00,3a,0c,39,00,d6,00,00,1f,72
352 TZ  DATA 67,00,00,2e,0c,39,00,d3,00,00
353 BM  DATA 1f,72,67,00,00,22,0c,39,00,d2
354 w0  DATA 00,00,1f,72,67,00,00,16,0c,39
355 SZ  DATA 00,d7,00,00,1f,72,67,00,00,0a
356 lQ  DATA 13,fc,00,01,00,00,1f,84,0c,39
357 Ir  DATA 00,01,00,00,1f,84,67,00,00,cc
358 89  DATA 0c,39,00,01,00,00,1f,93,67,00
359 oI  DATA 00,14,08,39,00,07,00,bf,e0,01
360 AL  DATA 66,00,00,b4,4e,f9,00,00,0d,34
361 r4  DATA 08,39,00,06,00,bf,e0,01,66,00
362 mW  DATA 00,a2,33,fc,ff,dd,00,00,1f,64
363 q5  DATA 61,00,09,ae,0c,39,00,01,00,00
364 ul  DATA 1f,93,67,00,00,0e,34,39,00,df
365 jn  DATA f0,0c,4e,f9,00,00,0d,5e,34,39
366 8J  DATA 00,df,f0,0a,08,02,00,01,67,00
367 tt  DATA 00,0c,23,fc,ff,ff,ff,ff,00,00
368 MN  DATA 1f,60,08,02,00,09,67,00,00,0c
369 jZ  DATA 23,fc,00,00,00,01,00,00,1f,60
370 PB  DATA 0c,39,00,01,00,00,1f,8d,66,00
371 dF  DATA 00,1e,0c,39,00,01,00,00,1f,82
372 EQ  DATA 66,00,00,12,23,fc,00,00,00,00
373 Y3  DATA 00,00,1f,78,4e,f9,00,00,0d,d4
374 Rg  DATA 3c,02,e2,4e,b5,46,08,06,00,08
375 Lf  DATA 67,00,00,1e,33,fc,ff,c4,00,00
376 oF  DATA 1f,64,23,fc,00,20,00,00,00,00
377 Tg  DATA 1f,78,23,fc,00,00,30,00,00,00
378 Xx  DATA 1f,74,13,fc,00,01,00,00,1f,84
379 ia  DATA 13,fc,00,ff,00,00,1f,85,1e,05
380 Aq  DATA 9c,3a,11,99,20,3c,00,00,00,04
381 Jm  DATA 1e,3c,00,00,bc,07,66,00,00,08
382 Y3  DATA 13,c7,00,00,1f,85,52,07,51,c8
383 IY  DATA ff,f0,20,3c,00,00,00,04,1e,3c
384 wm  DATA 00,0c,bc,07,66,00,00,08,13,c0
385 Rb  DATA 00,00,1f,85,52,07,51,c8,ff,f0
386 nT  DATA 20,3c,00,00,00,04,1e,3c,00,06
387 oq  DATA bc,07,66,00,00,0a,13,fc,00,06
388 hh  DATA 00,00,1f,85,52,07,51,c8,ff,ee
389 X8  DATA 20,3c,00,00,00,01,1e,3c,00,04
390 hh  DATA bc,07,66,00,00,0a,13,fc,00,04
391 n4  DATA 00,00,1f,85,52,07,bc,07,66,00
392 8i  DATA 00,0a,13,fc,00,04,00,00,1f,85
393 Vp  DATA 5c,07,51,c8,ff,ec,3c,3a,10,fe
394 NE  DATA 9c,7a,10,e4,0c,39,00,00,00,00
395 y8  DATA 1f,85,66,00,00,24,0c,06,00,07
396 2e  DATA 65,00,00,1c,0c,06,00,0e,62,00
397 f5  DATA 00,14,13,fc,00,00,00,00,1f,84
398 6i  DATA 23,fc,00,00,00,00,00,00,1f,6a
399 qr  DATA 0c,39,00,01,00,00,1f,85,66,00
400 9w  DATA 00,3e,0c,06,00,04,65,00,00,36
401 Vq  DATA 0c,06,00,11,62,00,00,2e,13,fc
402 FV  DATA 00,00,00,00,1f,84,0c,06,00,07
403 Dz  DATA 62,00,00,0c,23,fc,ff,ff,f9,00
404 mn  DATA 00,00,1f,6a,0c,06,00,0e,65,00
405 JF  DATA 00,0c,23,fc,00,00,07,00,00,00
406 93  DATA 1f,6a,0c,39,00,02,00,00,1f,85
407 0A  DATA 66,00,00,3e,0c,06,00,03,65,00
408 6q  DATA 00,36,0c,06,00,12,62,00,00,2e
409 9h  DATA 13,fc,00,00,00,00,1f,84,0c,06
410 Ll  DATA 00,07,62,00,00,0c,23,fc,ff,ff
411 06  DATA f9,00,00,00,1f,6a,0c,06,00,0e
412 Xg  DATA 65,00,00,0c,23,fc,00,00,07,00
413 Co  DATA 00,00,1f,6a,0c,39,00,03,00,00
414 hk  DATA 1f,85,66,00,00,50,0c,06,00,01
415 0v  DATA 65,00,00,48,0c,06,00,12,62,00
416 pY  DATA 00,40,13,fc,00,00,00,00,1f,84
417 j6  DATA 0c,06,00,07,62,00,00,0c,23,fc
418 lV  DATA ff,ff,f9,00,00,00,1f,6a,0c,06
419 6m  DATA 00,03,62,00,00,0c,23,fc,ff,ff
420 Cn  DATA f2,00,00,00,1f,6a,0c,06,00,0e
421 gp  DATA 65,00,00,0c,23,fc,00,00,07,00
422 P2  DATA 00,00,1f,6a,0c,39,00,04,00,00
423 ln  DATA 1f,85,66,00,00,50,0c,06,00,00
424 94  DATA 65,00,00,48,0c,06,00,12,62,00
425 yh  DATA 00,40,13,fc,00,00,00,00,1f,84
426 sF  DATA 0c,06,00,07,62,00,00,0c,23,fc
427 S5  DATA ff,ff,f2,00,00,00,1f,6a,0c,06
428 Fv  DATA 00,03,62,00,00,0c,23,fc,ff,ff
429 Lw  DATA f2,00,00,00,1f,6a,0c,06,00,0e
430 py  DATA 65,00,00,0c,23,fc,00,00,07,00
431 gL  DATA 00,00,1f,6a,0c,39,00,06,00,00
432 uw  DATA 1f,85,66,00,00,50,0c,06,00,00
433 ID  DATA 65,00,00,48,0c,06,00,12,62,00
434 7q  DATA 00,40,13,fc,00,00,00,00,1f,84
435 1O  DATA 0c,06,00,07,62,00,00,0c,23,fc
436 3n  DATA ff,ff,f9,00,00,00,1f,6a,0c,06
437 04  DATA 00,03,62,00,00,0c,23,fc,ff,ff
438 U5  DATA f2,00,00,00,1f,6a,0c,06,00,0e
439 y7  DATA 65,00,00,0c,23,fc,00,00,07,00
440 0j  DATA 00,00,1f,6a,0c,39,00,55,00,00
441 zX  DATA 1f,72,67,00,00,46,0c,39,00,54
442 E1  DATA 00,00,1f,72,67,00,00,3a,0c,39
443 Rx  DATA 00,56,00,00,1f,72,67,00,00,2e
444 2m  DATA 0c,39,00,57,00,00,1f,72,67,00
445 PV  DATA 00,22,0c,39,00,58,00,00,1f,72
446 6k  DATA 67,00,00,16,0c,39,00,53,00,00
447 5k  DATA 1f,72,67,00,00,0a,13,fc,00,01
448 N6  DATA 00,00,1f,84,0c,39,00,01,00,00
449 NS  DATA 1f,84,67,00,00,d6,0c,39,00,01
450 CQ  DATA 00,00,1f,93,67,00,00,14,08,39
451 TE  DATA 00,06,00,bf,e0,01,66,00,00,be
452 Nb  DATA 4e,f9,00,00,10,c8,08,39,00,07
453 gU  DATA 00,bf,e0,01,66,00,00,ac,33,fc
454 NS  DATA 00,23,00,00,1f,64,61,00,06,1a
455 hi  DATA 0c,39,00,01,00,00,1f,93,67,00
456 aT  DATA 00,0e,34,39,00,df,f0,0a,4e,f9
457 6H  DATA 00,00,10,f2,34,39,00,df,f0,0c
458 83  DATA 08,02,00,01,67,00,00,0c,23,fc
459 Qj  DATA ff,ff,ff,ff,00,00,1f,60,08,02
460 r6  DATA 00,09,67,00,00,0c,23,fc,00,00
461 tz  DATA 00,01,00,00,1f,60,0c,39,00,01
462 ZQ  DATA 00,00,1f,8d,67,00,00,28,0c,39
463 aN  DATA 00,01,00,00,1f,82,66,00,00,1c
464 vt  DATA 23,fc,00,00,00,00,00,00,1f,78
465 ol  DATA 23,fc,00,00,30,00,00,00,1f,74
466 wu  DATA 4e,f9,00,00,11,72,3c,02,e2,4e
467 V4  DATA b5,46,08,06,00,00,67,00,00,1e
468 ld  DATA 33,fc,00,3c,00,00,1f,64,23,fc
469 mS  DATA 00,20,00,00,00,00,1f,78,23,fc
470 CE  DATA 00,00,30,00,00,00,1f,74,0c,39
471 PZ  DATA 00,01,00,00,1f,86,66,00,01,08
472 zk  DATA 0c,39,00,01,00,00,1f,8c,66,00
473 8x  DATA 00,26,0c,39,00,93,00,00,1f,72
474 fH  DATA 62,00,00,0e,52,39,00,00,1f,88
475 YO  DATA 4e,f9,00,00,12,74,52,39,00,00
476 Dh  DATA 1f,87,4e,f9,00,00,12,74,0c,39
477 G3  DATA 00,01,00,00,1f,8a,66,00,00,32
478 1l  DATA 0c,39,00,01,00,00,1f,8b,66,00
479 Kd  DATA 00,0e,52,39,00,00,1f,88,4e,f9
```

```
480 VT DATA 00,00,12,74,0c,39,00,02,00,00
481 Sh DATA 1f,8b,66,00,00,0e,52,39,00,00
482 Jn DATA 1f,87,4e,f9,00,00,12,74,0c,39
483 QK DATA 00,00,00,00,1f,8a,66,00,00,80
484 1z DATA 0c,39,00,01,00,00,1f,82,66,00
485 K9 DATA 00,26,0c,39,00,93,00,00,1f,72
486 rT DATA 62,00,00,0e,52,39,00,00,1f,88
487 ka DATA 4e,f9,00,00,12,74,52,39,00,00
488 Pt DATA 1f,87,4e,f9,00,00,12,74,0c,39
489 bf DATA 00,93,00,00,1f,72,62,00,00,0e
490 wZ DATA 52,39,00,00,1f,87,4e,f9,00,00
491 T6 DATA 12,74,52,39,00,00,1f,88,4e,f9
492 da DATA 00,00,12,74,0c,39,00,01,00,00
493 et DATA 1f,8b,66,00,00,0e,52,39,00,00
494 Vz DATA 1f,87,4e,f9,00,00,12,74,0c,39
495 zm DATA 00,02,00,00,1f,8b,66,00,00,08
496 1h DATA 52,39,00,00,1f,88,61,00,00,e4
497 RT DATA 0c,39,00,01,00,00,1f,94,67,00
498 s8 DATA 00,8a,22,7c,00,00,1f,48,24,7c
499 Dp DATA 00,00,1f,4c,08,03,00,00,67,00
500 Ad DATA 00,14,13,7c,00,01,00,03,15,7c
501 ru DATA 00,01,00,03,4e,f9,00,00,12,b6
502 7w DATA 13,7c,00,00,00,03,15,7c,00,00
503 pd DATA 00,03,e2,4b,14,84,54,04,15,44
504 EB DATA 00,02,15,43,00,01,12,85,54,05
505 pf DATA 13,45,00,02,13,43,00,01,33,fc
506 fc DATA a0,20,00,df,f0,9c,23,fa,0c,66
507 Ks DATA 00,00,19,70,23,fa,0c,62,00,00
508 lu DATA 19,b8,23,fa,0c,5e,00,00,19,58
509 6c DATA 23,fa,0c,5a,00,00,19,64,23,fa
510 m5 DATA 0c,56,00,00,18,c8,23,fa,0c,52
511 18 DATA 00,00,19,10,4e,fa,f2,d4,2a,3c
512 qE DATA 00,00,00,14,61,00,03,c2,51,cd
513 SX DATA ff,fa,61,00,03,54,61,00,02,f8
514 bg DATA 13,fc,00,00,00,00,1f,8d,13,fc
515 LA DATA 00,00,00,00,1f,91,13,fc,00,00
516 Wm DATA 00,00,1f,92,61,00,00,1e,13,fc
517 cW DATA 00,00,00,00,1f,96,13,fc,00,00
518 Pn DATA 00,00,1f,94,13,fc,00,00,00,00
519 pn DATA 1f,95,4e,fa,f2,86,24,3c,00,00
520 VT DATA 00,00,14,39,00,00,1f,87,d4,39
521 P6 DATA 00,00,1f,88,84,fc,00,05,48,42
522 Hc DATA 0c,42,00,00,66,00,00,16,0c,39
523 bL DATA 00,01,00,00,1f,95,67,00,00,0a
524 tK DATA 08,79,00,00,00,00,1f,8d,0c,39
525 IM DATA 00,01,00,00,1f,93,67,00,00,18
526 Tr DATA 13,fa,0b,eb,00,00,1f,8e,13,fa
527 Wj DATA 0b,e4,00,00,1f,8f,4e,f9,00,00
528 WZ DATA 13,c0,13,fa,0b,d5,00,00,1f,8f
529 MI DATA 13,fa,0b,ce,00,00,1f,8e,0c,39
530 oo DATA 00,15,00,00,1f,87,65,00,00,18
531 EK DATA 08,79,00,00,00,00,1f,8d,13,fc
532 HF DATA 00,01,00,00,1f,95,4e,f9,00,00
533 P7 DATA 13,fe,0c,39,00,15,00,00,1f,88
534 WJ DATA 65,00,00,12,08,79,00,00,00,00
535 wJ DATA 1f,8d,13,fc,00,01,00,00,1f,95
536 dP DATA 24,3c,00,00,00,00,14,3a,0b,88
537 s1 DATA b4,3a,0b,86,65,00,00,2e,94,39
538 mr DATA 00,00,1f,8f,0c,02,00,01,66,00
539 p1 DATA 00,0e,52,39,00,00,1f,90,4e,f9
540 Gc DATA 00,00,14,3c,08,79,00,00,00,00
541 lF DATA 1f,93,52,39,00,00,1f,91,61,00
542 xb DATA 02,36,14,3a,0b,51,b4,3a,0b,4e
543 Yz DATA 65,00,00,30,94,39,00,00,1f,8e
544 E9 DATA 0c,02,00,01,66,00,00,0e,52,39
545 4m DATA 00,00,1f,90,4e,f9,00,00,14,76
546 gc DATA 08,79,00,00,00,00,1f,93,06,39
547 ww DATA 00,01,00,00,1f,92,61,00,01,fc
548 43 DATA 0c,39,00,02,00,00,1f,91,66,00
549 V2 DATA 00,0a,13,fc,00,01,00,00,1f,94
550 AA DATA 0c,39,00,02,00,00,1f,92,66,00
551 X4 DATA 00,0a,13,fc,00,01,00,00,1f,94
552 Jw DATA 14,39,00,00,1f,91,d4,39,00,00
553 P3 DATA 1f,92,0c,02,00,02,66,00,00,2e
554 UY DATA 0c,39,00,01,00,00,1f,96,67,00
555 Bh DATA 00,22,0c,39,00,0a,00,00,1f,8e
556 i1 DATA 66,00,00,06,61,00,01,de,0c,39
557 ZM DATA 00,0a,00,00,1f,8f,66,00,00,06
558 fe DATA 61,00,01,ce,0c,39,00,01,00,00
559 1t DATA 1f,93,67,00,00,18,33,fc,0f,00
560 wS DATA 00,df,f1,a4,33,fc,06,00,00,df
561 X2 DATA f1,bc,4e,f9,00,00,15,10,33,fc
562 eg DATA 06,00,00,df,f1,a4,33,fc,0f,00
563 OQ DATA 00,df,f1,bc,14,3a,0a,7c,84,fc
564 fV DATA 00,0a,20,7a,0a,16,d1,fc,00,00
565 wa DATA 20,92,20,3c,00,00,00,03,61,00
566 JO DATA 02,dc,61,00,02,b2,48,42,4e,f9
567 9o DATA 00,00,15,3c,14,3a,0a,54,20,7a
568 xH DATA 09,f2,d1,fc,00,00,20,94,20,3c
569 UL DATA 00,00,00,03,61,00,02,b8,61,00
570 De DATA 02,68,24,3c,00,00,00,00,14,3a
571 MJ DATA 0a,33,84,fc,00,0a,20,7a,09,cc
572 nO DATA d1,fc,00,00,22,9a,20,3c,00,00
573 7n DATA 00,03,61,00,02,92,61,00,02,68
574 8r DATA 48,42,4e,f9,00,00,15,86,14,3a
575 jS DATA 0a,0b,20,7a,09,a8,d1,fc,00,00
576 xT DATA 22,9c,20,3c,00,00,00,03,61,00
577 zG DATA 02,6e,61,00,02,1e,20,7a,09,94
578 ke DATA d1,fc,00,00,20,98,20,3c,00,00
579 wM DATA 00,03,14,39,00,00,1f,91,61,00
580 cY DATA 02,50,61,00,02,26,20,7a,09,76
581 rJ DATA d1,fc,00,00,22,a0,20,3c,00,00
582 3U DATA 00,03,14,39,00,00,1f,92,61,00
583 CD DATA 02,32,61,00,02,08,23,fc,00,00
584 7H DATA 00,00,00,00,19,58,23,fc,00,00
585 3j DATA 00,00,00,00,19,64,33,fc,82,02
586 TA DATA 00,df,f0,96,3b,7c,01,40,00,b6
587 Kh DATA 61,00,00,d8,3b,7c,01,ab,00,b6
588 rK DATA 61,00,00,ce,33,fc,00,02,00,df
589 0a DATA f0,96,61,00,00,04,4e,75,13,fc
590 fQ DATA 00,01,00,00,1f,82,13,fc,00,00
591 St DATA 00,00,1f,83,13,fc,00,00,00,00
592 at DATA 1f,86,23,fc,00,00,00,00,00,00
593 Kk DATA 1f,74,23,fc,00,00,00,00,00,00
594 T7 DATA 1f,60,33,fc,00,00,00,00,1f,64
595 oj DATA 23,fc,00,00,00,00,00,00,1f,66
596 jl DATA 13,fc,00,00,00,00,1f,8a,13,fc
597 0b DATA 00,00,00,00,1f,8b,13,fc,00,00
598 x3 DATA 00,00,1f,8c,4e,75,13,fc,00,01
599 AO DATA 00,00,1f,8d,13,fc,00,00,00,00
600 Ll DATA 1f,95,13,fc,00,15,00,00,1f,90
601 6S DATA 13,fc,00,00,00,00,1f,87,13,fc
602 3y DATA 00,00,00,00,1f,88,13,fc,00,00
603 GV DATA 00,00,1f,8e,13,fc,00,00,00,00
604 1k DATA 1f,8f,4e,75,08,79,00,00,00,00
605 U3 DATA 1f,93,13,fc,00,01,00,00,1f,96
606 EG DATA 14,3a,08,cb,13,fa,08,c8,00,00
607 eS DATA 1f,87,13,c2,00,00,1f,88,08,79
608 kV DATA 00,00,00,00,1f,8d,4e,75,24,3c
609 Pa DATA 00,00,00,01,51,c8,ff,fe,20,3c
610 f0 DATA ff,ff,ff,ff,51,ca,ff,f4,4e,75
611 A4 DATA 13,fc,00,00,00,00,1f,8a,33,fc
612 5K DATA 02,5f,00,df,f0,a6,33,fc,82,01
613 wZ DATA 00,df,f0,96,13,fc,00,3c,00,00
614 dY DATA 1f,89,23,fc,00,1a,00,00,00,00
615 va DATA 1f,78,23,fc,00,00,18,00,00,00
616 VD DATA 1f,74,23,fa,08,48,00,00,1f,66
617 yt DATA 4e,75,33,c3,00,00,1f,6e,23,fc
618 a7 DATA 13,00,00,00,00,00,1f,7c,20,3c
619 ff DATA 00,00,00,00,4e,75,30,86,d1,fc
620 yN DATA 00,00,00,28,51,c8,ff,f6,20,3c
621 Qb DATA 00,00,00,04,e3,5e,51,c9,ff,ea
622 Jc DATA 4e,75,30,86,d1,fc,00,00,00,28
623 gf DATA 51,c8,ff,f6,20,3c,00,00,00,04
624 GH DATA e2,5e,51,c9,ff,ea,4e,75,30,86
625 y9 DATA d1,fc,00,00,00,28,51,c8,ff,f6
626 Ng DATA 4e,75,30,86,d1,fc,00,00,00,28
627 Mb DATA e3,5e,51,c8,ff,f4,4e,75,24,11
628 XT DATA 20,82,d3,fc,00,00,00,04,d1,fc
629 MQ DATA 00,00,00,04,34,11,30,82,d3,fc
630 KG DATA 00,00,00,02,d1,fc,00,00,00,24
631 dG DATA 51,c8,ff,de,4e,75,c4,38,00,0f
632 uR DATA 14,11,10,82,d1,fc,00,00,00,28
633 Jd DATA 14,29,00,01,10,82,d3,fc,00,00
634 w4 DATA 00,02,d1,fc,00,00,00,28,51,c8
635 QZ DATA ff,de,4e,75,14,11,11,42,00,01
636 xl DATA d1,fc,00,00,00,28,14,29,00,01
637 sj DATA 11,42,00,01,d3,fc,00,00,00,02
638 qk DATA d1,fc,00,00,00,28,51,c8,ff,de
639 7m DATA 4e,75,0c,02,00,00,66,00,00,08
640 CX DATA 22,7c,00,00,1a,da,0c,02,00,01
641 Uy DATA 66,00,00,08,22,7c,00,00,1a,92
642 IR DATA 0c,02,00,02,66,00,00,08,22,7c
643 D6 DATA 00,00,1a,9a,0c,02,00,03,66,00
644 qY DATA 00,08,22,7c,00,00,1a,a2,0c,02
645 q0 DATA 00,04,66,00,00,08,22,7c,00,00
646 pI DATA 1a,aa,0c,02,00,05,66,00,00,08
647 NO DATA 22,7c,00,00,1a,b2,0c,02,00,06
648 2w DATA 66,00,00,08,22,7c,00,00,1a,ba
649 Zn DATA 0c,02,00,07,66,00,00,08,22,7c
650 Ch DATA 00,00,1a,c2,0c,02,00,08,66,00
651 5a DATA 00,08,22,7c,00,00,1a,ca,0c,02
652 RH DATA 00,09,66,00,00,08,22,7c,00,00
653 CZ DATA 1a,d2,4e,75,22,3c,00,00,00,07
654 Q6 DATA d1,fc,00,00,00,01,1c,10,08,06
655 R9 DATA 00,07,67,00,00,06,08,43,00,00
656 dB DATA 08,03,00,00,67,00,00,06,08,c6
657 pr DATA 00,07,e3,1e,51,c9,ff,e4,10,86
658 4g DATA 51,c8,ff,d0,4e,75,a0,5a,b1,00
659 9I DATA 00,00,00,01,00,00,00,0f,00,00
660 zs DATA 00,1f,00,00,00,7f,00,80,00,7f
661 Np DATA 00,80,00,7f,01,80,00,7f,ff,80
662 JP DATA 00,7f,ff,80,ff,ff,ff,80,00,7f
663 PT DATA 01,80,00,7f,00,80,00,7f,00,80
664 lW DATA 00,7f,00,00,00,7f,00,00,00,1f
665 YP DATA 00,00,00,0f,00,00,00,01,a0,5a
666 Um DATA b1,00,00,00,fc,00,00,00,ff,80
667 zx DATA 00,00,ff,c0,00,00,ff,f0,00,00
668 ny DATA ff,f8,00,00,ff,f8,00,00,ff,fc
669 Ff DATA 00,00,ff,fc,00,00,ff,fc,00,00
670 b0 DATA ff,fc,00,00,ff,fc,00,00,ff,f8
671 bV DATA 00,00,ff,f8,00,00,ff,f0,00,00
672 HJ DATA ff,c0,00,00,ff,80,00,00,fc,00
673 Oj DATA a0,6a,a2,00,c0,00,00,00,c0,00
674 yq DATA 00,00,a8,6a,aa,00,00,00,c0,00
675 Bf DATA 00,00,c0,00,a0,5a,b1,00,00,00
676 t4 DATA 00,fe,00,00,07,ff,00,00,0f,ff
677 PL DATA 00,00,3f,ff,00,00,7f,ff,00,00
678 Mb DATA 7f,ff,00,00,ff,ff,00,00,ff,ff
679 hD DATA 00,00,ff,ff,00,00,ff,ff,00,00
680 Kb DATA ff,ff,00,00,7f,ff,00,00,7f,ff
681 uj DATA 00,00,3f,ff,00,00,0f,ff,00,00
682 Rc DATA 07,ff,00,00,00,fe,a0,62,b0,00
683 8D DATA 00,00,00,00,00,00,c0,00,00,00
684 kF DATA e0,00,00,00,f8,00,04,00,f8,00
685 oO DATA 04,00,f8,00,06,00,f8,00,07,fc
686 df DATA f8,00,07,fc,ff,fc,07,fc,f8,00
687 5v DATA 06,00,f8,00,04,00,f8,00,04,00
688 UK DATA f8,00,00,00,f8,00,00,00,e0,00
689 af DATA 00,00,c0,00,00,00,00,00,00,00
690 ir DATA 00,00,01,20,00,00,01,22,00,00
691 zM DATA 01,24,00,00,01,26,00,00,01,28
692 UD DATA 00,00,01,2a,00,00,01,2c,00,00
693 58 DATA 01,2e,00,00,01,30,00,00,01,32
694 JX DATA 00,00,01,34,00,00,01,36,00,00
695 Ai DATA 01,38,00,00,01,3a,00,00,01,3c
696 re DATA 00,00,01,3e,00,00,00,e0,00,00
697 WN DATA 00,e2,00,00,00,e4,00,00,00,e6
698 Nz DATA 00,00,00,e8,00,00,00,ea,00,00
699 ZE DATA 00,ec,00,00,00,ee,00,00,ff,ff
700 uu DATA ff,fe,7c,c1,e6,6f,be,00,66,c3
701 nW DATA 36,6c,33,00,66,c3,33,cc,33,00
702 MI DATA 6c,c3,f1,8f,36,00,60,c3,31,8c
703 g6 DATA 36,00,60,c3,31,8c,33,00,60,fb
704 Sh DATA 31,8f,33,00,38,18,18,18,18,18
705 uN DATA 18,00,3c,66,06,1c,30,60,7e,00
706 wa DATA 3c,66,06,1c,06,66,3c,00,66,66
707 7k DATA 66,3e,06,06,06,00,7e,60,7c,06
708 Bz DATA 06,06,7c,00,3c,60,60,7c,66,66
709 6J DATA 3c,00,7c,0c,0c,18,18,30,30,00
710 2f DATA 3c,66,66,3c,66,66,3c,00,3c,66
711 ii DATA 66,3e,06,06,3c,00,3c,66,66,66
712 hW DATA 66,66,3c,00,00,31,5a,75,7f,75
713 JS DATA 5a,31,00,cf,a6,8b,81,8b,a6,cf
714 zy DATA 70,80,3d,b7,64,a0,80,80,80,9f
715 1l DATA 5c,d8,80,80,80,bd,3f,61,10,88
716 mN DATA 80,80,9a,3f,7f,84,80,a4,f3,ff
717 Oj DATA 3b,42,d8,80,80,3f,74,24,c0,80
718 Os DATA 80,97,4b,68,f0,84,80,1f,34,d0
719 J2 DATA e4,80,bf,bf,d7,00,2f,e8,80,80
720 Za DATA cf,3f,20,a6,8b,80,82,ff,40,e8
721 7K DATA 80,80,cf,20,00,f7,e0,a0,80,9f
722 Ue DATA 30,02,cb,ba,80,9b,df,fd,c0,a9
723 4z DATA 99,a6,d7,19,1c,a4,80,9f,ef,cc
724 dh DATA f3,e0,b0,94,8b,d8,04,d0,bf,ac
725 XI DATA a8,af,0f,08,ca,87,90,b7,ef,f2
```

Listing 1. (Fortsetzung)

```
726 Ps  DATA fd,a0,80,a7,f2,f3,cc,a8,92,a7
727 I8  DATA 08,f1,c8,b7,8c,a2,f7,07,e0,d5
728 cN  DATA e9,b8,93,b0,d6,d7,ca,df,b0,94
729 9V  DATA d3,e1,d0,b3,94,c9,cf,02,fe,d0
730 6K  DATA 80,b7,b0,f7,f1,c8,9e,97,ce,be
731 su  DATA df,d0,a7,c5,d0,e5,cc,bc,d7,d0
732 29  DATA ec,80,b3,ff,b8,b7,e5,c2,88,9b
733 dL  DATA 17,28,d8,80,97,bd,f7,f3,a8,90
734 1f  DATA 99,c9,fa,f4,c8,97,9f,ad,db,eb
735 Gw  DATA e0,d1,8f,c4,f5,f4,ba,ae,b5,d9
736 BH  DATA a7,ca,e7,e9,d8,bf,a3,a6,ef,00
737 uU  DATA e2,89,95,bf,db,dc,d6,b0,84,a7
738 PE  DATA f1,f3,d2,b0,9b,c7,c0,df,00,b8
739 Zf  DATA 9d,b0,bf,bf,e0,e5,ac,a2,bf,e7
740 NR  DATA ec,d9,cd,b0,a7,c4,df,eb,d8,b3
741 TO  DATA a7,d0,e7,d8,bc,af,b8,c7,c1,b8
742 MJ  DATA cb,bc,a0,93,bf,e0,df,c8,b2,bd
743 cr  DATA c0,d7,e9,d0,a4,b7,c8,cf,d2,ce
744 dx  DATA c9,b6,bf,df,ea,d5,bf,c5,c2,d7
745 v7  DATA dc,cf,c9,cf,cf,c0,bb,dd,d8,c1
746 1c  DATA b3,c0,ab,c7,cf,cd,b6,b3,b9,c8
747 Wl  DATA c9,c0,b4,b3,c5,c4,c3,c4,ce,ad
748 7P  DATA c0,cf,d3,e8,d6,c0,bf,d7,cd,cf
749 FO  DATA d3,cc,c3,b7,d0,d7,c0,ba,b3,ce
750 1G  DATA d8,d7,e0,db,c6,b8,bd,db,d4,c6
751 aB  DATA b4,b8,b5,ce,d7,c3,b0,af,bb,e0
752 ML  DATA d8,c6,c8,b3,b4,cf,d2,ca,c3,c5
753 J1  DATA c0,c5,d3,d4,d3,c2,bc,bf,e0,df
754 1l  DATA d0,c0,af,bf,d8,d1,c3,bf,b7,b0
755 B2  DATA c7,d7,d0,c3,b7,b8,cb,d0,de,d3
756 XK  DATA b8,be,c5,d0,d3,bf,c0,bf,bf,c8
757 OI  DATA cd,c3,cc,c1,cb,d3,cd,d0,cf,c9
758 7t  DATA cc,bf,ba,cc,cf,c8,c7,bf,c4,d7
759 B9  DATA d0,d0,bd,b5,c6,cd,cc,cf,b9,b0
760 ru  DATA cf,c9,c8,ca,c3,cc,c3,c2,ca,c5
761 Il  DATA c8,c7,c5,c4,cd,c7,c8,c2,cb,d6
762 nq  DATA cd,ca,cb,cb,d6,d3,c4,ca,d3,cb
763 8F  DATA ce,c9,ca,ca,bd,c8,d3,c5,c4,c5
764 HP  DATA c0,c4,bf,c1,c4,bf,c0,c7,c6,c8
765 Ya  DATA c3,c3,c8,c7,c5,c8,cf,d0,cf,c4
766 Pk  DATA c0,d3,ca,c4,bd,cb,d0,bf,c2,cf
767 XO  DATA c7,c4,c7,c1,c4,c9,c4,c8,c7,c2
768 vA  DATA cf,cf,d0,cf,cb,c8,cb,c5,c4,bf
769 YI  DATA c3,c8,c7,c4,c7,cb,d0,cf,d1,d0
770 Mc  DATA cf,d5,d8,d2,cc,cd,c7,c8,cf,c7
771 gl  DATA c8,c4,bb,c8,c9,c0,bc,b3,b0,b7
772 TW  DATA bf,c8,bf,bd,c0,cb,d1,d8,ca,c0
773 HB  DATA c7,cd,d0,cf,c5,c8,c7,c9,d4,d5
774 cE  DATA d0,cf,c7,c8,cb,c7,ca,c9,bb,bc
775 ol  DATA be,c6,cc,c9,c9,c3,c2,c4,c7,c2
776 5Z  DATA c4,c3,ba,bf,c3,c9,cf,c7,c1,cb
777 ot  DATA cf,d2,cf,c3,c6,c5,c9,cf,ce,cc
778 Zf  DATA cb,c9,cc,cb,c7,cc,cb,c2,c6,c9
779 mr  DATA c5,cb,c5,c4,c6,c5,cc,d3,cc,d2
780 Jb  DATA cb,c6,cd,cb,c8,cd,c5,c4,c7,c9
781 w8  DATA d0,c7,c3,ca,c7,cc,d3,ce,cc,c6
782 S8  DATA c3,ca,c7,c8,ca,bf,c2,cd,cb,c8
783 5X  DATA c5,c1,ca,cb,cc,d8,ca,c6,cf,cb
784 vO  DATA c4,c7,c6,cc,ca,c6,ca,c5,c4,cb
785 z2  DATA c9,c4,c7,c3,c0,c3,c1,c4,bd,bb
786 tw  DATA c6,c7,c4,c7,c9,c8,cb,c7,d0,cf
787 Ki  DATA c9,d8,d3,d0,cf,cb,ca,cb,c9,d0
788 PZ  DATA cd,c9,d0,ca,c0,c4,c2,c0,c3,bf
789 PN  DATA c2,c7,c3,c8,c7,c3,cc,c7,c8,cd
790 kv  DATA cb,d2,cd,cb,d0,cf,ce,d4,d5,d0
791 Fr  DATA d5,cf,d0,d3,c5,c8,c7,c5,cc,c5
792 Qw  DATA c0,c7,c2,c0,c3,bd,c0,bf,be,c2
793 Me  DATA c3,c3,ca,c3,c8,cb,c5,ca,cb,c8
794 YA  DATA cc,cb,c4,cc,cb,c8,cd,c5,c4,c9
795 yb  DATA c7,d0,cf,cb,d0,ca,c4,cd,c7,c4
796 x9  DATA c7,c5,c8,cb,cb,d0,cd,c5,cc,cd
797 Nu  DATA ca,d1,c9,cc,cd,cd,d0,cf,c9,ce
798 Wx  DATA cb,c9,ce,c9,c8,cb,c7,cc,cd,cb
799 qZ  DATA d0,cd,c8,cf,c7,c4,c7,c1,c2,c2
800 oI  DATA bd,c2,bd,b9,c4,bf,c3,cb,c9,cc
801 EF  DATA cd,c7,d0,ca,c8,cc,cb,cc,d3,d1
802 DW  DATA d0,cf,cb,d0,d3,c9,ce,cb,c4,c7
803 S4  DATA c3,c4,c7,c1,c8,c9,c3,c8,c9,c5
804 Xp  DATA cc,c3,c8,c7,bf,c4,c7,c6,ca,cb
805 2x  DATA c9,ce,c9,c9,cf,ca,c8,cf,cb,cc
806 I8  DATA cf,cd,d0,cb,ca,d3,cf,ca,d5,d3
807 Sz  DATA d0,ce,c7,cc,c7,c5,ca,bf,c0,bf
808 Mv  DATA bd,c0,c3,c0,c8,c5,c0,c8,c3,c3
809 T3  DATA cb,c9,cc,cf,cb,d0,d2,cd,d4,ce
810 0g  DATA cc,d5,cf,d2,d3,cf,d0,cf,c7,c8
811 2S  DATA c6,c1,ca,c3,c4,c7,c1,c0,c5,c6
812 we  DATA c8,c9,c1,c8,c3,c5,ce,c7,c8,cb
813 kC  DATA c5,c8,c4,c4,c8,c5,c3,ce,cb,cc
814 9n  DATA d3,d1,d2,d7,d4,dc,d9,d5,dc,d3
815 jI  DATA d2,d7,d3,d8,d6,d2,d2,cf,cc,cc
816 kV  DATA c6,c0,c7,bd,c0,bf,b7,c0,bb,b9
817 aj  DATA c4,bf,c0,c6,be,c0,bf,be,c0,c3
818 6v  DATA be,c8,c9,c8,ce,c7,ca,cf,c9,c9
819 jn  DATA cd,c7,cc,cb,c8,d2,cf,d0,d7,d1
820 h4  DATA cc,cf,c9,d0,cd,c5,cc,c7,c8,cc
821 7A  DATA c7,c8,cb,c7,c8,cb,c7,cc,cc,c9
822 jr  DATA ce,c5,c1,c7,c3,37,00,00,00,00
823 HX  DATA 00,00,00,00,00,00,00,00,00,00
824 IY  DATA 00,00,00,00,00,00,00,00,00,00
825 JZ  DATA 00,00,00,00,00,00,00,00,00,00
826 Ka  DATA 00,00,00,00,00,00,00,00,00,00
827 Lb  DATA 00,00,00,00,00,00,00,00,00,00
828 Mc  DATA 00,00,00,00,00,00,00,00,00,00
829 Nd  DATA 00,00,00,00,00,00,00,00,00,00
830 Oe  DATA 00,00,00,00,00,00,00,00,00,00
831 Pf  DATA 00,00,00,00,00,00,00,00,00,00
832 Qg  DATA 00,00,00,00,00,00,00,00,00,00
833 w1  DATA 00,00,03,ec,00,00,01,53,00,00
834 rR  DATA 00,00,00,00,18,8e,00,00,18,80
835 Bp  DATA 00,00,18,72,00,00,18,64,00,00
836 cx  DATA 18,56,00,00,18,48,00,00,18,3a
837 Eh  DATA 00,00,18,2c,00,00,18,1e,00,00
838 bC  DATA 18,10,00,00,17,36,00,00,17,2c
839 SB  DATA 00,00,17,24,00,00,17,1c,00,00
840 zQ  DATA 17,12,00,00,17,08,00,00,16,f0
841 gH  DATA 00,00,16,d0,00,00,16,c8,00,00
842 O9  DATA 16,c2,00,00,16,b6,00,00,16,ae
843 IA  DATA 00,00,16,a4,00,00,16,9c,00,00
844 SO  DATA 16,94,00,00,16,8c,00,00,16,84
845 eC  DATA 00,00,16,7c,00,00,16,74,00,00
846 b3  DATA 16,6a,00,00,16,62,00,00,16,5a
847 1v  DATA 00,00,16,52,00,00,16,48,00,00
848 nN  DATA 16,40,00,00,16,36,00,00,16,2c
849 SE  DATA 00,00,16,24,00,00,16,1c,00,00
850 0h  DATA 15,ea,00,00,15,e0,00,00,15,ce
851 G3  DATA 00,00,15,b0,00,00,15,7e,00,00
852 3r  DATA 15,34,00,00,14,fc,00,00,14,e2
853 Er  DATA 00,00,14,d2,00,00,14,c2,00,00
854 Gm  DATA 14,b6,00,00,14,a6,00,00,14,a0
855 p7  DATA 00,00,14,9a,00,00,14,8e,00,00
856 89  DATA 14,86,00,00,14,7a,00,00,14,6e
857 2m  DATA 00,00,14,66,00,00,14,5e,00,00
858 sp  DATA 14,58,00,00,14,4a,00,00,14,34
859 1y  DATA 00,00,14,2e,00,00,14,26,00,00
860 PY  DATA 14,20,00,00,14,12,00,00,13,fa
861 kO  DATA 00,00,13,f2,00,00,13,e6,00,00
862 Kv  DATA 13,de,00,00,13,d8,00,00,13,d0
863 Jr  DATA 00,00,13,c4,00,00,13,bc,00,00
864 j9  DATA 13,b4,00,00,13,ac,00,00,13,a6
865 iJ  DATA 00,00,13,9e,00,00,13,92,00,00
866 rO  DATA 13,8a,00,00,13,7e,00,00,13,68
867 cW  DATA 00,00,13,62,00,00,13,52,00,00
868 eA  DATA 13,4a,00,00,13,42,00,00,13,36
869 11  DATA 00,00,13,2e,00,00,13,26,00,00
870 01  DATA 13,04,00,00,12,fc,00,00,12,f4
871 6c  DATA 00,00,12,ec,00,00,12,e4,00,00
872 Dv  DATA 12,dc,00,00,12,a6,00,00,12,8c
873 8q  DATA 00,00,12,86,00,00,12,7c,00,00
874 Gp  DATA 12,70,00,00,12,66,00,00,12,5e
875 vq  DATA 00,00,12,58,00,00,12,4e,00,00
876 pj  DATA 12,46,00,00,12,40,00,00,12,3a
877 HA  DATA 00,00,12,34,00,00,12,2a,00,00
878 5G  DATA 12,22,00,00,12,1c,00,00,12,16
879 7L  DATA 00,00,12,10,00,00,12,06,00,00
880 pT  DATA 11,fa,00,00,11,ee,00,00,11,e6
881 XI  DATA 00,00,11,e0,00,00,11,d6,00,00
882 Me  DATA 11,ce,00,00,11,c8,00,00,11,be
883 yb  DATA 00,00,11,b2,00,00,11,aa,00,00
884 27  DATA 11,a4,00,00,11,9e,00,00,11,98
885 ZJ  DATA 00,00,11,8e,00,00,11,82,00,00
886 GN  DATA 11,76,00,00,11,6e,00,00,11,64
887 7x  DATA 00,00,11,5a,00,00,11,44,00,00
888 x6  DATA 11,3e,00,00,11,34,00,00,11,26
889 UV  DATA 00,00,11,1a,00,00,11,12,00,00
890 ht  DATA 11,00,00,00,10,e8,00,00,10,d8
891 1m  DATA 00,00,10,cc,00,00,10,b8,00,00
892 6D  DATA 10,a2,00,00,10,96,00,00,10,8e
893 67  DATA 00,00,10,82,00,00,10,76,00,00
894 3G  DATA 10,6a,00,00,10,5e,00,00,10,52
895 ik  DATA 00,00,10,46,00,00,10,3e,00,00
896 nh  DATA 10,2c,00,00,10,1a,00,00,10,08
897 mk  DATA 00,00,0f,ec,00,00,0f,e4,00,00
898 fi  DATA 0f,d2,00,00,0f,c0,00,00,0f,ae
899 hi  DATA 00,00,0f,92,00,00,0f,8a,00,00
900 hV  DATA 0f,78,00,00,0f,66,00,00,0f,54
901 wZ  DATA 00,00,0f,38,00,00,0f,30,00,00
902 OX  DATA 0f,1e,00,00,0f,0c,00,00,0e,f0
903 Pk  DATA 00,00,0e,e8,00,00,0e,d6,00,00
904 1r  DATA 0e,c4,00,00,0e,a8,00,00,0e,a0
905 ip  DATA 00,00,0e,96,00,00,0e,7a,00,00
906 41  DATA 0e,64,00,00,0e,54,00,00,0e,36
907 Lr  DATA 00,00,0e,18,00,00,0d,fc,00,00
908 jA  DATA 0d,e0,00,00,0d,d8,00,00,0d,d0
909 t4  DATA 00,00,0d,c6,00,00,0d,bc,00,00
910 v4  DATA 0d,a6,00,00,0d,a0,00,00,0d,92
911 o6  DATA 00,00,0d,86,00,00,0d,7e,00,00
912 SJ  DATA 0d,6c,00,00,0d,54,00,00,0d,44
913 1n  DATA 00,00,0d,38,00,00,0d,24,00,00
914 1n  DATA 0d,0e,00,00,0d,02,00,00,0c,fa
915 Wd  DATA 00,00,0c,ee,00,00,0c,e2,00,00
916 Ik  DATA 0c,d6,00,00,0c,ca,00,00,0c,be
917 Sd  DATA 00,00,0c,b2,00,00,0c,aa,00,00
918 rc  DATA 0c,98,00,00,0c,86,00,00,0c,74
919 2i  DATA 00,00,0c,58,00,00,0c,50,00,00
920 z0  DATA 0c,3e,00,00,0c,2c,00,00,0c,1a
921 r2  DATA 00,00,0b,fe,00,00,0b,f6,00,00
922 fd  DATA 0b,e4,00,00,0b,d2,00,00,0b,c0
923 YG  DATA 00,00,0b,a4,00,00,0b,9c,00,00
924 PC  DATA 0b,8a,00,00,0b,78,00,00,0b,5c
925 vb  DATA 00,00,0b,54,00,00,0b,42,00,00
926 aD  DATA 0b,30,00,00,0b,14,00,00,0b,0c
927 wp  DATA 00,00,0b,02,00,00,0a,e6,00,00
928 29  DATA 0a,d0,00,00,0a,c0,00,00,0a,a2
929 bG  DATA 00,00,0a,84,00,00,0a,68,00,00
930 Jc  DATA 0a,52,00,00,0a,4a,00,00,0a,3c
931 jL  DATA 00,00,0a,34,00,00,0a,2c,00,00
932 wf  DATA 0a,22,00,00,0a,18,00,00,0a,12
933 7L  DATA 00,00,0a,04,00,00,09,fc,00,00
934 aq  DATA 09,f2,00,00,09,e6,00,00,09,d2
935 ep  DATA 00,00,09,ca,00,00,09,c2,00,00
936 7z  DATA 09,ba,00,00,09,ae,00,00,09,a6
937 Zt  DATA 00,00,09,8e,00,00,09,82,00,00
938 32  DATA 09,7a,00,00,09,66,00,00,09,36
939 yO  DATA 00,00,09,2e,00,00,09,1e,00,00
940 y2  DATA 09,16,00,00,09,06,00,00,08,fa
941 qF  DATA 00,00,08,f2,00,00,08,e8,00,00
942 pN  DATA 08,de,00,00,08,d2,00,00,08,ca
943 bw  DATA 00,00,08,be,00,00,08,b2,00,00
944 yG  DATA 08,a6,00,00,08,9e,00,00,08,96
945 Pi  DATA 00,00,08,8a,00,00,08,82,00,00
946 Aq  DATA 08,7a,00,00,08,6e,00,00,08,62
947 fC  DATA 00,00,08,58,00,00,08,4a,00,00
948 XF  DATA 08,40,00,00,08,36,00,00,08,2c
949 87  DATA 00,00,08,18,00,00,08,12,00,00
950 lc  DATA 07,fa,00,00,07,f2,00,00,07,ea
951 hC  DATA 00,00,07,e2,00,00,07,d8,00,00
952 K5  DATA 07,c2,00,00,07,ac,00,00,07,a4
953 4N  DATA 00,00,07,92,00,00,07,7c,00,00
954 JM  DATA 07,70,00,00,07,64,00,00,07,38
955 46  DATA 00,00,07,18,00,00,07,12,00,00
956 cF  DATA 07,08,00,00,06,e4,00,00,06,be
957 SI  DATA 00,00,06,ae,00,00,06,74,00,00
958 Uu  DATA 06,54,00,00,06,4e,00,00,06,44
959 Js  DATA 00,00,06,20,00,00,05,fa,00,00
960 em  DATA 05,ea,00,00,05,44,00,00,05,1a
961 VY  DATA 00,00,05,12,00,00,04,f8,00,00
962 2C  DATA 04,f0,00,00,04,e8,00,00,04,e0
963 K3  DATA 00,00,04,d4,00,00,04,c8,00,00
964 A6  DATA 04,ae,00,00,04,94,00,00,04,7a
965 Q0  DATA 00,00,00,d2,00,00,00,cc,00,00
966 90  DATA 00,64,00,00,00,5e,00,00,00,58
967 dU  DATA 00,00,00,52,00,00,00,4c,00,00
968 nd  DATA 00,46,00,00,00,40,00,00,00,3a
969 GC  DATA 00,00,00,34,00,00,00,1a,00,00
970 zw  DATA 00,00,00,00,03,f2
(C) 1989 M&T
```

Listing 1. (Schluß)

Ein Klassiker im neuen Gewand

Solitaire ist ein fesselndes Spiel, das durch ausgereifte Grafik in Verbindung mit einer bewährten Spielidee für ein faszinierendes Spielgeschehen sorgt.

Fast jeder kennt das klassische Brettspiel Solitaire. Der Spieler muß dabei mit den Steinen so geschickt andere überspringen, daß am Ende nur noch einer übrig ist. Hier finden Sie eine Umsetzung auf den Amiga, die durch tolle Grafik besticht und wegen der großen Anzahl von Steinen einen höheren Schwierigkeitsgrad aufweist.

Nach dem Start von Solitaire sehen Sie den Spielplan bis auf das zentrale Feld mit Steinen besetzt (Bild 1). Spielziel ist es, so viele Spielsteine wie möglich abzuräumen. Springen Sie mit einem beliebigen Stein über einen benachbarten, so wird der übersprungene Stein vom Feld genommen.

Wie schon gesagt, bleibt beim Start das mittlere Feld frei; auf dieses muß der erste Zug erfolgen. Dazu klickt man mit der Maus auf den gewünschten Spielstein und bewegt den Zeiger mit gedrückter Maustaste auf das Zielfeld. Bitte beachten Sie, daß Ihre Steine nur horizontal oder vertikal, aber nicht diagonal springen können. Außerdem dürfen keine Leerstellen übersprungen werden, sondern Sie müssen bei jedem Zug einen Stein vom Spielfeld entfernen.

Nach jedem gültigen Zug zeigt die Zuganzeige einen Punkt mehr an und die Score-Anzeige einen weniger (der Score bezeichnet die Anzahl der auf dem Brett verbliebenen Steine). Im Gegensatz zum klassischen Brettspiel Solitaire genügt es nicht, 32 Steine abzuräumen. Bei Solitaire für den Amiga haben Sie es gleich mit 124 Steinen zu tun.

Wenn Sie lange genug tüfteln, werden Sie sicher immer bessere Spieltaktiken herausfinden. Hier einige Tips für den Einstieg:

— Räumen Sie grundsätzlich eine Ecke nach der anderen ab. Wechseln Sie zwischen den einzelnen Ecken, bevor diese vollständig abgeräumt sind, so bleiben häufig einzelne Spielsteine stehen. Diese sind später nur sehr schwer aus ihrer Isolation zu holen.

— Streben Sie am Ende eine L-förmige Konstellation aus vier Steinen an. Gelingt dies, ist es sehr einfach, das Spiel zu lösen.

— Wem die Aufgabe zu leicht wird, der sollte versuchen, den letzten Stein in die Mitte zu bringen oder ein beliebiges anderes Feld für den letzten Stein zu bestimmen.

Taktik ist alles

Mit der rechten Maustaste haben Sie zusätzlich zwei Pull-Down-Menüs zur Verfügung. Das linke Menü (»Solitaire«) dient dazu, das Spiel zu beenden (beispielsweise wenn Sie keine Zugmöglichkeit mehr haben). Im rechten (»Hilfe«) können Sie jeweils den letzten Zug zurücknehmen. Falls Sie das Spiel noch nicht so gut beherrschen, können Sie sich alle Zugmöglichkeiten eines beliebigen Steins anzeigen lassen. Ist diese Funktion aktiviert, erscheint rechts unten im Spielfeld das Wort »HELP«.

Bitte verwenden Sie zum Eingeben den Checksummer auf Seite 159. Laden Sie zuerst Amiga-Basic und von Basic aus Solitaire »Open«. Starten Sie das Programm mit »Start« aus dem »Run«-Menü.

(Christian Buchner/so)

Programmname:	Solitaire
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

```
Programmautor: Uwe Grunenberg

----------------------------
 1 E20 REM ****************************
 2 nR  REM *                          *
 3 eu  REM *    S O L I T A I R E     *
 4 pT  REM *                          *
 5 qU  REM *                          *
 6 ku  REM *    von Uwe Grunenberg    *
 7 sW  REM *                          *
 8 Yh  REM * (C) 1988 Markt & Technik *
 9 uY  REM *                          *
10 NB  REM ****************************
11 j1  DIM feld%(106),leer%(106),mark%(106)
12 9E  DIM SF(17,17)
13 PM  SCREEN 1,320,250,5,1
14 vB  WINDOW 2,"",,0,1
15 or  PALETTE 0,0,0,0
16 5f  PALETTE 1,.5,.5,.5
17 JN  PALETTE 6,0,.74,.24
18 42  HiScore=124
19 rx  CLS
20 WD  GOSUB PutGrafik
21 Xy  Menue:
22 00  MENU 1,0,1," Solitaire      "
23 Sm  MENU 1,1,1," Neues Spiel    "
24 R9  MENU 1,2,1," Kein Zug mehr! "
25 t8  MENU 1,3,1," Programmende   "
26 kv  MENU 2,0,1,"Hilfe"
27 Rj  MENU 2,1,1,"Zug zeigen"
28 6N  MENU 2,2,1,"Zug zurück"
29 JP  MENU 3,0,0,""
30 MT  MENU 4,0,0,""
31 HK  ON MENU GOSUB MenueAnwahl
32 OY  Start:
33 GP  Help=0 : Zug=0 : Score=124
34 6C  CLS
35 83  GOSUB FeldArray
36 AB  GOSUB Spielfeld
37 Hq  GOSUB Kugelaufbau
38 6S  GOSUB Anzeigen
39 Gw  Schleife:
40 pZ  GOSUB Test
41 s7  GOTO Schleife
42 Pf  MenueAnwahl:
43 Xd  Men=MENU(0)
44 ck  Menpkt=MENU(1)
45 IT  IF Men=1 THEN
46 hE1  ON Menpkt GOTO Spielende,Spielende,Programmende
47 M50 ELSE
48 f61  ON Menpkt GOTO Zuganzeige,Zugruecknahme
49 jc0 END IF
50 gW  GOTO MenueAuswahl
51 Tq  FeldArray:
52 R6  FOR i=1 TO 17
53 Y92   FOR j=1 TO 17
54 JC4     SF(i,j)=-1
55 Md2   NEXT j
56 Lb0 NEXT i
57 VA  FOR i=2 TO 16
58 X82   FOR j=7 TO 11
59 474     SF(i,j)=1
60 R12   NEXT j
61 Qg0 NEXT i
62 VA  FOR i=7 TO 11
63 hI2   FOR j=2 TO 16
64 9C4     SF(i,j)=1
65 Wn2   NEXT j
66 V10 NEXT i
67 f8  SF(9,9)=0
68 S4  RETURN
69 S5  Kugelaufbau:
70 j0  FOR i=1 TO 17
71 qR2   FOR j=1 TO 17
72 1r4     IF SF(i,j)=1 THEN
73 wa5       PUT (i*12,j*12),feld% ,PSET :SOUND 500,1:FOR d=1 TO 10
             0:NEXT d
74 814     END IF
```

Listing 1. Solitaire geben Sie bitte mit dem Checksummer auf Seite 159 ein

```
 75 gx2    NEXT j
 76 fv0  NEXT i
 77 a4   PUT (9*12,9*12),leer%,PSET :SOUND 600,1
 78 cE   RETURN
 79 wu   Test:
 80 Ww   MENU ON
 81 ry   MENU STOP
 82 xE   XPos=INT(MOUSE(1)/12):YPos=INT(MOUSE(2)/12)
 83 gv   IF MOUSE(0)<>0 THEN Setz
 84 iK   RETURN
 85 Yc   Setz:
 86 jy   IF MOUSE(0)<>0 THEN Setz
 87 sz   IF XPos>17 OR XPos<1 OR YPos<1 OR YPos>17 THEN Test
 88 eu   IF SF(XPos,YPos)=1 THEN GOTO ZugTest
 89 Gt   GOTO Test
 90 JG   ZugTest:
 91 7t   IF Help=1 THEN GOSUB Markierung
 92 Xa   MouseWait:
 93 iw   IF MOUSE(0)<>0 THEN MouseWait
 94 cb   XMove=INT(MOUSE(1)/12):YMove=INT(MOUSE(2)/12)
 95 BN   IF XMove>0 AND XMove<18 AND YMove>0 AND YMove<18 THEN
 96 nY1   PUT (XPos*12,YPos*12),leer%,PSET:SOUND 600,1
 97 x8    IF SF(XMove,YMove)=0 THEN
 98 5x    XDist=ABS(XPos-XMove):YDist=ABS(YPos-YMove)
 99 Yl2    IF XDist=2 AND YDist=0 OR XDist=0 AND YDist=2 THEN
100 uf3     UeberX=(XPos+XMove)/2:UeberY=(YPos+YMove)/2
101 P1      IF SF(UeberX,UeberY)=1 THEN
102 SS4      GOTO Zugausfuehren
103 bU3     END IF
104 cV2    END IF
105 dW1   END IF
106 eX0 END IF
107 33  PUT (XPos*12,YPos*12),feld%,PSET:SOUND 500,1:GOTO Test
108 yP  Zugausfuehren:
109 iu  PUT (UeberX*12,UeberY*12),leer%,PSET:SOUND 500,1
110 Cn  PUT (XMove*12,YMove*12),feld%,PSET:SOUND 600,1
111 AM  SF(XPos,YPos)=0:SF(UeberX,UeberY)=0:SF(XMove,YMove)=1
112 60  XPosAlt=XPos:YPosAlt=YPos
113 TM  XMoveAlt=XMove:YMoveAlt=YMove
114 1m  UeberXAlt=UeberX:UeberYAlt=UeberY
115 eY  GOSUB Punkte
116 hK  GOTO Test
117 dY  END
118 Gz  Punkte:
119 BB  Score=Score-1
120 ir  Zug=Zug+1
121 SK  IF Score<HiScore THEN HiScore=Score
122 hn  COLOR 10
123 lR  LOCATE 16,32: PRINT USING"# # # ";Score
124 yE  LOCATE 24,32:PRINT USING"# # # ";HiScore
125 DN  LOCATE 8,32:PRINT USING"# # # ";Zug
126 00  RETURN
127 jE  Markierung:
128 mf  Move=0
129 it  IF XPos<15 THEN
130 EV1   IF SF(XPos+2,YPos)=0 AND SF(XPos+1,YPos)=1 AND SF(XPos+1,Y
          Pos)<>-1 THEN PUT ((XPos+2)*12,YPos*12),mark%,PSET:Move=1
131 3w0 END IF
132 CV  IF XPos>3 THEN
133 n21   IF SF(XPos-2,YPos)=0 AND SF(XPos-1,YPos)=1 AND SF(XPos-1,Y
          Pos)<>-1 THEN PUT ((XPos-2)*12,YPos*12),mark%,PSET:Move=1
134 6z0 END IF
135 t5  IF YPos<15 THEN
136 3A1   IF SF(XPos,YPos+2)=0 AND SF(XPos,YPos+1)=1 AND SF(XPos,YPo
          s+1)<>-1 THEN PUT (XPos*12,(YPos+2)*12),mark%,PSET:Move=1
137 920 END IF
138 Nh  IF YPos>3 THEN
139 Yd1   IF SF(XPos,YPos-2)=0 AND SF(XPos,YPos-1)=1 AND SF(XPos,YPo
          s-1)<>-1 THEN PUT (XPos*12,(YPos-2)*12),mark%,PSET:Move=1
140 C50 END IF
141 mk  IF Move=1 THEN
142 YD1   FOR Warten=1 TO 500:NEXT Warten
143 uZ    FOR i=1 TO 17
144 1c2     FOR j=1 TO 17
145 ty3       IF SF(i,j)=0 THEN
146 f14        PUT (i*12,j*12),leer%,PSET
147 JC3       END IF
148 r82     NEXT j
149 q61   NEXT i
150 MF0 END IF
151 nP  RETURN
152 qJ  Programmende:
153 17  CLS
154 GL  SCREEN CLOSE 1
155 hN  WINDOW CLOSE 2
156 HI  PALETTE 0,1,1,1
157 SC  SYSTEM
158 AZ  Spielende:
159 7D  CLS
160 u1  GOTO Start
161 9T  Zuganzeige:
162 xV  IF Help=1 THEN
163 Uy1   Help=0:COLOR 1,0:LOCATE 27,31:PRINT "     "
164 FyO ELSE
165 yf1   Help=1:COLOR 1,0:LOCATE 27,31:PRINT "HELP!"
166 cVO END IF
167 W9  GOTO Test
168 bI  Zugruecknahme:
169 IU  IF Zug=0 OR Zug=Zurueck THEN Test
170 Kk  PUT (XPosAlt*12,YPosAlt*12),feld%,PSET:SF(XPosAlt,YPosAlt)=
        1
171 K5  PUT (UeberXAlt*12,UeberYAlt*12),feld%,PSET:SF(UeberXAlt,Ueb
        erYAlt)=1
172 cC  PUT (XMoveAlt*12,YMoveAlt*12),leer%,PSET:SF(XMoveAlt,YMoveA
        lt)=0
173 C6  Score=Score+1:Zurueck=Zug
174 Xd  COLOR 10
175 bH  LOCATE 16,32: PRINT USING"# # # ";Score
176 fI  GOTO Test
177 Vt  Spielfeld:
178 S7  FOR i=2 TO 16
179 sm1   LINE(i*12+6,18)-(i*12+6,210),6
180 Lb0 NEXT i
181 bC  FOR j=2 TO 16
182 y41   LINE(18,j*12+6)-(210,j*12+6),6
183 Qh0 NEXT j
184 qV  LINE(18,18)-(210,210),23,b
185 H8  LINE(20,16)-(212,16),25 : LINE -(212,208),25
186 89  LINE(21,15)-(213,15),25 : LINE -(213,207),25
187 Nz  RETURN
188 4e  Anzeigen:
189 qF  LINE (233,36)-(286,51),23,bf
190 cs  LINE (235,34)-(288,34),25 : LINE -(288,49),25
191 KU  CIRCLE (235,43),2,0 : CIRCLE (284,43),2,0
192 7X  LINE (243,53)-(276,67),23,bf
193 tv  LINE (278,53)-(278,65),25
194 xC  LOCATE 6,31:COLOR 17:PRINT "MOVES"
195 sy  COLOR 10
196 MW  LOCATE 8,32:PRINT USING"# # # ";Zug
197 WZ  LINE (233,100)-(286,115),23,bf
198 UC  LINE (235,98)-(288,98),25 : LINE -(288,113),25
199 Zs  CIRCLE (235,107),2,0 : CIRCLE (284,107),2,0
200 zy  LOCATE 14,31:COLOR 17:PRINT "SCORE"
201 v4  LINE (243,117)-(276,131),23,bf
202 06  LINE (278,117)-(278,129),25
203 06  COLOR 10
204 Jm  LOCATE 16,32: PRINT USING"# # # ";Score:SOUND 420,1
205 k7  LINE (233,164)-(286,179),23,bf
206 00  LINE (235,162)-(288,162),25 : LINE -(288,177),25
207 dS  CIRCLE (235,171),2,0 : CIRCLE (284,171),2,0
208 Xr  LINE (243,181)-(276,195),23,bf
209 P9  LINE (278,181)-(278,193),25
210 oo  LOCATE 22,31:COLOR 17:PRINT "HI SC"
211 8E  COLOR 10
212 Oe  LOCATE 24,32:PRINT USING"# # # ";HiScore
213 nP  RETURN
214 N6  PutGrafik:
215 TC  CIRCLE (100,100),5,23:CIRCLE (100,100),4,24
216 MF  CIRCLE (100,100),3,25:CIRCLE (100,100),2,26
217 0h  CIRCLE (100,100),1,27
218 dY  PSET (100,100),28:PSET (102,99),27
219 2q  PSET (98,99),27 :PSET (98,101),27
220 sv  PSET (102,101),27
221 zL  PSET (100,106),10 : PSET (106,100),10
222 xI  PSET (94,100),10 : PSET (100,94),10
223 hF  GET(94,94)-(106,106),feld%
224 AG  CLS
225 6H  LINE (100,94)-(100,106),10 : LINE (94,100)-(106,100),10
226 pS  GET(94,94)-(106,106),leer%
227 DJ  CLS
228 3t  LINE (100,94)-(100,106),17 : LINE (94,100)-(106,100),17
229 2H  GET(94,94)-(106,106),mark%
230 4g  RETURN
(C) 1988 M&T
```

Listing 1. (Schluß)

Moderne Zeiten

Sind Sie ein Fan von Leiter- und Hüpfspielen, dann sollten Sie »Mad Factory« laden, das Telefon abstellen und Freund oder Freundin vor den Fernseher setzen.

Mad Factory« (Listing 1) verpflanzt Sie in eine Umgebung aus feindseligen Maschinen, brüchigen Böden und jeder Menge Überraschungen. Sie sind — wie einst Charlie Chaplin in »Moderne Zeiten« — ein gestreßter Arbeitnehmer, der seinem verdienten Feierabend entgegenfiebert. Aber bevor Sie den Joystick aus der Hand legen dürfen und Ihre Stempeluhr drücken, müssen Sie erst alle acht Abteilungen (Level) der Mad Factory durchlaufen.

Ihre Arbeit besteht darin, die in den Fabrikhallen verstreuten Diamanten aufzusammeln. Von der Anzahl der gesammelten Diamanten ist Ihre Entlohnung (Score) abhängig.

Dies hört sich einfacher an als es ist. Nicht alle Diamanten können sofort eingesammelt werden. Sie sollten umsichtig vorgehen: Gier führt zum Verlust eines Spielers. Zudem sind in den Abteilungen die Arbeitsmaschinen außer Kontrolle geraten, sie spielen verrückt! In den Fabrikhallen rasen Loren über alles hinweg, was sich ihnen in den Weg stellt. Kolben rasen auf und

Diamonds are forever

nieder, da Sie keine ausreichende Schutzkleidung tragen, schlägt sich die Berührung mit dem Kolben in der Unfallstatistik in der Rubrik Todesfälle nieder. Manchmal gibt der Boden nach, wenn Sie sich zu lange an einer Stelle aufhalten. Säure ist auf dem Boden verschüttet worden und wildgewordene Schraubendreher stechen alles nieder. Jede Berührung mit einer dieser verrückten Maschinen kostet eines Ihrer drei kostbaren Leben. Jede Menge Hindernisse stehen also auf Ihrem Weg zum Erfolg. Kurz: Eine Aufgabe für qualifizierte Fachkräfte.

Akkordarbeitern (Joystick-Akrobaten) wird es nicht leicht gemacht eine schnelle Mark zu machen; denn der Weg zu Diamanten und Ausgängen ist manchmal durch Türen versperrt. Um nun doch noch an die wertvollen Diamanten hinter den Türen zu gelangen, benötigen Sie die passenden Schlüssel, mit denen sich die Türen öffnen lassen. Vorsicht: Für Hin- und Rückweg benötigen Sie zwei Schlüssel!

Haben Sie in einer Fabrikhalle alle Diamanten eingesammelt oder einfach nur »die Nase voll«, können Sie diese Halle jederzeit durch den violetten Torbogen verlassen. Für dieses Tor benötigt man keinen Schlüssel. Von dort gelangen Sie dann in die nächste Fabrikhalle. Schlüssel, von denen Sie nicht wissen, ob sie gleich gebraucht werden, sollten für Türen in den nächsten Hallen gesammelt werden. Die Schlüssel gehen nicht verloren, auch nicht bei Verlust eines Lebens der Spielfigur.

Haben Sie sich einmal festgefahren, weil Sie ohne Plan in den Fabrikhallen herumgelaufen sind (Beispiel: Sie stehen zwischen zwei Türen und haben keinen Schlüssel mehr), hilft Ihnen nur noch die »HELP«-Taste. Dies kostet allerdings eines Ihrer drei Leben und Sie beginnen in derselben Halle von vorne.

Wie kann man nun in dieser verrückten Fabrik zu einem erfolgreichen Vertreter der Arbeiterklasse werden? Der Weg dazu führt nur über eine geschickte Steuerung des Joysticks. Mit einem Joystick in Port 2 manövrieren Sie Ihre Spielfigur über die Klippen der Arbeitswelt. Die Figur bewegt sich nach rechts beziehungsweise links, wenn der Joystick in die entsprechende Richtung

Tip für Speicher-Riesen

Mad Factory kommt zwar problemlos mit einem größeren Speicherausbau zurecht und ist auch ohne »NoFastMem« lauffähig. Aber wenn Sie einen Amiga mit mehr als 512 KByte Speicher haben, so läuft das Programm bis zu 20 Prozent schneller, wenn Sie zuvor »NoFastMem« gestartet haben. Dieses Utility befindet sich in der Systemschublade der Workbench-Diskette. Leider konnten wir noch nicht mit letzter Sicherheit feststellen, woran dies lag. Wir vermuten aber, daß von Basic aus zunächst — soweit verfügbar — Fast-Memory belegt wird. Daten, für die Chip-Memory benötigt wird, werden bei Gelegenheit dorthin kopiert.

gehalten wird. Wenn dabei zusätzlich der Feuerknopf gedrückt ist, springt die Spielfigur in die jeweilige Richtung.

Zum Erklettern einer Leiter muß der Joystick nach oben oder unten gehalten werden. Schießen kann die Spielfigur erst, wenn vorher ein Blitz eingesammelt wurde. Der Joystick muß dann nach rechts oder links unten gedrückt werden. Der Schuß kann keine Maschinen oder Hindernisse zerstören, sondern kurzzeitig Löcher in den Boden brechen. Gegenstände (Blitz, Diamanten, Birne...) werden eingesammelt, indem sie von der Spielfigur berührt werden.

Wie im richtigen Leben

Wie im richtigen Leben hat das Fragezeichen eine besondere Bewandtnis: Man weiß nie genau, was sich dahinter verbirgt. Entweder gibt es einen Schlüssel, einen Diamanten, eine Glühbirne oder man wird in eine andere Fabrikhalle zurückgeschickt.

Die gefährlichen Hindernisse können mit dem richtigen Timing einfach übersprungen werden. Haben Sie einen Blitz eingesammelt und den Boden an der gewünschten Stelle weggeschossen, so bleiben Ihnen höchstens vier Sekunden Zeit diesen Durchgang zu nutzen. Danach läuft das in den Boden geschossene Loch voll Säure. Der Säureboden kann dann nicht mehr weggeschossen und auch nicht berührt werden. Da Ihnen ansonsten kein Zeitlimit für die Bewältigung Ihrer Arbeit gesetzt ist, können Sie sich die Fabrikhalle in Ruhe ansehen und einen sinnvollen Weg durch die Halle planen. Drauflosstürmen führt in den meisten Fällen zum Verlust eines Spielerlebens oder in eine Sackgasse.

Ins siebte Level sollten Sie mindestens fünf Schlüssel mitbringen, um durch die Türen zum Ausgang zu gelangen. Im achten Level ist zu Beginn ein Schlüssel nötig, um sich hinter der Tür vor dem Schraubendreher zu verschanzen. Dieser Schlüssel muß aus einer der vorigen Hallen mitgebracht werden!

Für die erfahrenen Basic-Programmierer unter Ihnen bietet es sich an, das Listing einzusehen und die für eine Trainingsversion des Spiels wichtigen Variablen zu verändern. Verwenden Sie vorsichtshalber eine Sicherheitskopie für diese Manipulationen.

Für Mad Factory gibt es einen Leveleditor, der mit Hilfe der Maus eigene Level zeichnet oder bereits vorhandene Level verändert. Dieser «LevelEd« ist ebenfalls ein Basic-Programm (Listing 3), mit dem Sie eigene Dateien anlegen. Sie haben damit die Möglichkeit, vom Arbeitnehmer zum Arbeitgeber aufzusteigen und Ihre eigene Fabrikhalle zu bauen.

Der Editor darf nur mit der Einstellung von 80 Zeichen pro Zeile verwendet werden. Alle verfügbaren Gegenstände (wie Mauer, Birne, Schraubendreher...) werden am unteren Bildschirmrand dargestellt (Icons). Das eigentliche Spielfeld des Levels wird als Raster auf dem Bildschirm dargestellt. Als Positionierhilfe sind die x-Koordinaten als Zahlen unter dem Raster angegeben. Um in einem Spielfeld die verschiedenen Gegenstände unterzubringen, werden diese am unteren Bildschirmrand mit dem Mauspfeil angeklickt und an der gewünschten Position durch nochmaliges Klicken abgelegt. Die mausgesteuerte Benutzeroberfläche des Editors verfügt über zwei Menüs. Das eine befindet sich rechts unten auf dem Bildschirm und enthält die Befehle »CLS«, »INFO« und »MOVE«. Das zweite wird durch Drücken der rechten Maustaste am oberen Bildschirmrand eingeblendet. Dieses sogenannte Projekt-Menü enthält die Befehle »SAVE«, »LOAD«, »NEW«, »RASTER«, »SPEZIAL« und »QUIT«.

»CLS« löscht den Bildschirm und die Position der Spielfigur. Die bestehenden Koordinaten für die Bewegung der Objekte bleiben jedoch erhalten.

»INFO« zeigt die derzeitigen Daten für die Anfangsposition und die Bewegungskoordinaten der Objekte und der Spielfigur an.

Durch Anklicken von »MOVE« können Sie die Daten für die Anfangsposition und die Bewegungskoordinaten der Objekte eingeben. Das Programm wartet auf die Eingabe von zwei oder vier Zahlenwerten, die durch Kommata zu trennen sind. Als Hilfe werden die x-Koordinaten unter dem Rasterfeld angezeigt. Die Koordinaten reichen von 0 bis 300 in Zwanziger-Schritten, wobei jedoch jeweils nur die beiden letzten Ziffern angezeigt werden. Nachdem Sie die erforderlichen Werte übergeben haben (abschließendes <RETURN> nicht vergessen), wird die entsprechende Stelle im Rasterfeld durch einen kleinen Kreis kenntlich gemacht. Der gesamte Bewegungsraum ist mit einem farbigen Rechteck umrandet. Sie können damit überprüfen, ob Sie »spielbare« Eingaben gemacht haben.

Der Boß sind Sie

Mit »SAVE« wird Ihr neu editierter Level gespeichert. Nach Eingabe eines Dateinamens werden Sie aufgefordert, die Nummer des eben fertiggestellten Levels einzugeben. Dazu muß noch die Nummer des Levels übergeben werden, bei dem nach Erreichen der lila Tür das Spiel fortgesetzt werden soll. Ferner fragt das Programm, wie es die im Spiel verwendeten Fragezeichen behandeln soll. Hier geben Sie bitte eine Zahl zwischen 0 und 4 ein. Dabei bedeuten:

0: Zufall (das Programm wählt zufällig aus den Ziffern 1 bis 3).

1: Das erste aufgesammelte Fragezeichen wird zu einem Schlüssel.

2: Das erste aufgesammelte Fragezeichen wird zu einem Diamanten.

3: Das erste aufgesammelte Fragezeichen wird zu einer Glühbirne.

4: Sie werden in einen anderen Level versetzt.

Der Menüpunkt »SAVE« kann nur gestartet werden, wenn alle Informationen über die Objekte und die Position der Spielfigur bekannt sind. Haben Sie diesen Menüpunkt versehentlich aufgerufen, so können Sie ihn wieder verlassen, indem Sie statt eines Dateinamens nur <RETURN> betätigen.

Achtung:

Während des Speicherns sollten Sie das Programm nicht unterbrechen, weil dann wegen des unvollständig abgespeicherten Levels Fehler auftreten können. Der Speichervorgang ist erst abgeschlossen, wenn die Meldung »Bitte warten« vom Bildschirm verschwindet!

Mit »LOAD« können Sie einen bereits bestehenden Level aus einer Datei einladen. Dazu geben Sie bitte den Dateinamen (eventuell mit Zugriffspfad) und die gewünschte Nummer an. Der LevelEd lädt alle nötigen Informationen. Auf dem Bildschirm wird der von Ihnen gewünschte Level, wie vom Spiel her gewohnt, aufgebaut. Jetzt können Sie nach Herzenslust Veränderungen vornehmen.

Der »NEW«-Befehl ist ähnlich dem »CLS«-Befehl. Neu ist lediglich, daß mit »NEW« nicht nur Informationen zum Spielfeld eines Levels gelöscht werden, sondern auch die Bewegungskoordinaten und Startpositionen der Objekte.

Mit dem »RASTER«-Befehl erreichen Sie einen Neuaufbau des Rasterfeldes auf dem Bildschirm.

Mit Hilfe des »SPEZIAL«-Befehls sind Sie in der Lage, die Spielfigur auch außerhalb des Bildschirms zu positionieren, was mit der Maussteuerung nicht möglich ist. Das Programm erwartet dann die Eingabe von x- und y-Koordinaten.

Mit dem »QUIT«-Befehl kommen Sie nach einer Sicherheitsabfrage zurück in das Basic-Fenster, um neu erstellte Level zu testen.

Zum Testen starten Sie bitte Mad Factory und achten darauf, daß das Programm die Level aus der richtigen Datei lädt. Um direkt in den gewünschten Level zu kommen, können Sie, sobald das Zwischenbild auftaucht (das Ihnen den aktuellen Level anzeigt), mit <CTRL C> unterbrechen und im Di-

Bedeutung der Gegenstände

Glühbirne:	bringt ein Bonusleben
Blitz:	die Spielfigur kann schießen
Schlüssel:	Türen können geöffnet werden
Diamant:	der Score steigt
Bruchboden:	zerbröckelt bei längerer Berührung
Schraubendreher:	bei Berührung Verlust eines Lebens
Wagen:	bei Berührung Verlust eines Lebens
Säureboden:	bei Berührung Verlust eines Lebens
Presse/Stiel	bei Berührung Verlust eines Lebens

rektmodus folgendes eingeben:

such=<neueLevelnummer>
<RETURN>
cont <RETURN>

Durch die Variable »such« wird die Nummer des nächsten Levels bestimmt. Dadurch brauchen Sie nicht alle Level, die vor dem zu testenden liegen, durchzuspielen.

Eingabehinweise

Mad Factory (Listing 1) geben Sie mit dem Checksummer auf Seite 159 ein. Sie benötigen dazu auch die Datei »Level« (Listing 2).

Starten Sie dann Amiga-Basic. Von Basic aus laden Sie Mad Factory und starten das Programm mit »Run«.

Wenn Sie eigene Levels erzeugen wollen, benötigen Sie den LevelEd. Diesen finden Sie in Listing 3. Achten Sie darauf, daß Ihre Level-Dateien im gleichen Verzeichnis liegen, wie das Hauptprogramm (»Mad Factory«).

Falls Ihnen die abgedruckten acht Level nicht reichen und Sie keine eigenen anlegen wollen, finden Sie auf der Programmservice-Diskette 80 fertige Levels. Damit sollte Mad Factory eigentlich nicht so leicht langweilig werden.

(Reinhold Brings/ Charles Röckelrath/so)

Programmname:	Mad Factory
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

Programmautor: Roman Stumm

```
 1 D30 ' *** Version 1.6 von Roman Stumm ***
 2 k1  SCREEN 1,320,200,5,1
 3 p7  WINDOW 1,"              The Mad Factory",,0,1
 4 hQ  DIM feind1(550),feind2(550),px(100),py(100)
 5 RU  DIM peng(105),birne(110),diamand(110),frage(110)
 6 aE  DIM key(110),gift(100),stein(110),ausgang(110),p(110)
 7 QZ  PALETTE 0,0,0,0: PALETTE 1,0,0,0
 8 sr  PALETTE 2,.7,.6,.55: PALETTE 3,.1,.3,.7
 9 Qd  PALETTE 4,1,1,1: PALETTE 5,.86,.86,0
10 zS  PALETTE 6,1,0,0: PALETTE 7,.93,.73,0
11 Da  PALETTE 8,0,.83,.99: PALETTE 9,0,0,0
12 8X  PALETTE 10,0,0,0: PALETTE 11,.72,.45,0
13 Gd  PALETTE 12,.4,.02,0: PALETTE 13,1,0,1
14 J7  PALETTE 14,0,1,.2: PALETTE 15,.36,.36,.36
15 jh  PALETTE 16 ,.95,.8,0: PALETTE 24,.75,.05,.05
16 wy  PALETTE 25,.55,.55,.55: PALETTE 26,.62,.62,.62
17 et  PALETTE 27,.68,.68,.68: PALETTE 28,.75,.75,.75
18 Rq  PALETTE 29,.8,.8,.8: PALETTE 30,.9,.9,.9
19 bW  LINE (9,2)-(5,13),6
20 ek  LINE -(15,11),6: LINE -(11,17),6
21 1p  LINE -(15,17),6: LINE -(19,7),6
22 dR  LINE -(10,9),6: LINE -(12,2),6
23 Bb  LINE -(9,2),6:PAINT(10,10),6
24 Y0  GET (2,0)-(19,19),peng:CLS
25 6L  CIRCLE (10,13),5,26,,,2.15:PAINT(10,13),26
26 1b  CIRCLE (10,6),6,5:PAINT(10,6),5
27 8k  GET(0,0)-(19,19),birne:CLS
28 18  LINE (10,1)-(1,10),31
29 f5  LINE -(10,19),31: LINE -(19,10),31
30 2c  LINE -(10,1),31: LINE -(7,10),31
31 Pj  LINE -(10,19),31: LINE -(13,10),31
32 An  LINE -(10,1),31
33 yT  PAINT(10,10),8,31:PAINT(6,10),8,31:PAINT(16,10),8,31
34 g9  GET(0,0)-(19,19),diamand:CLS
35 nn  LINE (0,0)-(19,19),9,bf
36 RF  CIRCLE (10,7),5,14,4.71,2.57
37 5V  LINE (10,12)-(10,15),14
38 H7  CIRCLE (10,18),1,14:PSET (10,18),14
39 RF  GET(0,0)-(19,19),frage:CLS
40 5x  CIRCLE (10,5),4,7
41 0w  CIRCLE (10,5),3,7
42 3I  LINE (9,9)-(10,17),7,bf
43 f2  LINE (11,17)-(13,17),7
44 Lc  LINE (11,14)-(13,14),7
45 wk  GET (0,0)-(19,19),key:CLS
46 Gy  LINE (0,5)-(18,11),24,bf
47 Si  LINE (0,0)-(19,5),14,bf
48 9z  FOR a=2 TO 18 STEP 4
49 69  CIRCLE (a,0),2,0:PAINT(a,0),0
50 z7  NEXT a
51 Uk  LINE (10,6)-(10,11),1
52 ig  GET (0,0)-(19,11),gift:CLS
53 wp  FOR a=8 TO 2 STEP -1
54 Ab  CIRCLE (10,10),a,a+23
55 y3  NEXT
56 31  PAINT (10,10),a+24
57 ca  GET(0,0)-(19,19),stein:CLS
58 KD  CIRCLE (10,7),7,13,,,1.35:PAINT(10,7),13
59 bD  LINE (5,5)-(0,19),13
60 y9  LINE -(19,19),13
61 IV  LINE -(15,5),13 :PAINT(9,16),13
62 2X  GET (0,0)-(19,19),ausgang:CLS
63 nG  LINE (96,99)-(90,109),3
64 RS  LINE -(109,109),3
65 9B  LINE -(103,99),3
66 f1  LINE -(96,99),3:PAINT (100,102),3
67 dh  CIRCLE (100,94),5,2:PAINT(100,94),2
68 sY  CIRCLE (98,93),1,4:PSET(98,93),1
69 tN  CIRCLE (102,93),1,4:PSET(102,93),1
70 7k  LINE (98,96)-(102,96),24
71 3j  PSET (94,93),24:PSET (94,94),24
72 oL  PSET (94,95),24:PSET (106,93),24
73 Yg  PSET (106,94),24:PSET(106,95),24
74 uM  PSET (93,94),24:PSET (107,94),24
75 hV  PSET (100,101),4:PSET(100,107),4
76 XG  PSET (100,104),4
77 vZ  GET (90,90)-(109,109),p
78 lH  leben=3:such=1
79 aH  Aufbau:
80 Rz  TIMER OFF:CLS:COLOR 6
81 rn  LOCATE 5,5:PRINT "MAD FACTORY by Roman Stumm 1988"
82 ds  COLOR 19
83 Gj  LOCATE 8,15 :PRINT "SCORE: "punkte
84 Fc  LOCATE 11,15 :PRINT "LIVES: "leben
85 nv  LOCATE 14,15 :PRINT "KEYS:  "schl
86 vp  IF such<0 THEN
87 NF  LOCATE 17,7:COLOR 6
88 f8  PRINT "You reached the Exit of the"
89 Zx  LOCATE 20,17 : PRINT "FACTORY"
90 SF  LOCATE 1:COLOR 0:tusch:tusch:END
91 PI  END IF
92 6m  LOCATE 17,15 :PRINT "STAGE: "such
93 CM  IF leben<=0 THEN
94 ed  LOCATE 22,12:COLOR 6
95 SU  PRINT "G A M E  O V E R":COLOR 0:LOCATE 1:END
96 UN  END IF
97 VY  LOCATE 22,10:PRINT "Press FIRE to start..."
98 Df  WHILE STRIG(3)=0:WEND
99 oW  WHILE STRIG(2)<>0:WEND
100 Xp WHILE INKEY$<>"" :WEND
101 0J CLS:a$=""
102 gY OPEN "Level" FOR INPUT AS #1
103 NJ schauen:
104 Mh WHILE NOT a$="Level"
105 QJ INPUT #1,a$:WEND:a$=""
106 H1 INPUT #1,runde
107 oC IF runde<>such THEN schauen
108 AV FOR iy=0 TO 160 STEP 20
109 LP INPUT #1,a$
110 01 FOR ix=0 TO 280 STEP 20
111 FF INPUT #1,a
112 5k IF a=-1 THEN frei
113 RK IF a=1 THEN boden
114 uj IF a=2 THEN mauer
115 4o IF a=3 THEN bruch
116 Q1 IF a=4 THEN PUT (ix,iy),stein
117 IL IF a=5 THEN leiter
118 W1 IF a=6 THEN door
119 Cr IF a=7 THEN PUT (ix,iy),ausgang
120 7C IF a=8 THEN PUT(ix,iy),frage
```

Listing 1. Moderne Zeiten einmal anders: »Mad Factory«

```
121 rr  IF a=9 THEN PUT(ix,iy),birne
122 Ya  IF a=10 THEN PUT(ix,iy),peng
123 5U  IF a=11 THEN PUT(ix,iy),key
124 OI  IF a=12 THEN PUT(ix,iy),diamand
125 OV  IF a=13 THEN feind1
126 Xg  IF a=14 THEN feind2
127 zB  IF a=15 THEN PUT(ix,iy),gift,PSET
128 gg  IF a=16 THEN stamper
129 OL  IF a=17 THEN platte
130 n6  IF a=18 THEN stiehl
131 Kf  weiter:
132 pV  NEXT ix
133 EI  frei:
134 ub  NEXT iy
135 y4  INPUT #1,a$,x,y:PUT (x,y),p
136 xH  INPUT #1,a$,ix,iy,ix2,iy2
137 oc  INPUT #1,a$,a,b,c,d
138 df  GET (a,b)-(c,d),feind1
139 qe  INPUT #1,a$,a,b,c,d
140 il  GET (a,b)-(c,d),feind2
141 48  INPUT #1,a$,sx,sy,zx,zy
142 9l  INPUT #1,a$,sx2,sy2,zx2,zy2
143 2m  INPUT #1,a$,weiter,bonus,quwei
144 YL  CLOSE #1
145 wr  zer=0:no=0:schuss=0:spring=0
146 QM  rich1=5:rich2=5:rich3=5:rich4=5
147 vP  ON TIMER(4) GOSUB neu
148 ap  scrolling:
149 U5  a$=INKEY$:IF a$="" THEN a$="0"
150 iS  IF ASC(a$)=139 THEN tod
151 sx  IF ix=>zx THEN rich1=-5
152 O7  IF ix<=sx THEN rich1=5
153 B9  IF iy=>zy THEN rich2=-5
154 hJ  IF iy<=sy THEN rich2=5
155 qj  IF ix2=>zx2 THEN rich3=-5
156 n1  IF ix2<=sx2 THEN rich3=5
157 6s  IF iy2>=zy2 THEN rich4=-5
158 4B  IF iy2<=sy2 THEN rich4=5
159 rY  IF sx=zx THEN rich1=0
160 7m  IF sy=zy THEN rich2=0
161 Ei  IF sx2=zx2 THEN rich3=0
162 Su  IF sy2=zy2 THEN rich4=0
163 7y  FOR a=1 TO 4
164 hN  PUT (ix,iy),feind1
165 n9  PUT (ix2,iy2),feind2
166 KN  ix=ix+rich1:iy=iy+rich2
167 BL  ix2=ix2+rich3:iy2=iy2+rich4
168 AC  PUT (ix,iy),feind1,PSET
169 sG  PUT (ix2,iy2),feind2,PSET
170 pu  NEXT
171 kH  Abfrage:
172 uQ  a=POINT(x+10,y+30):b=POINT(x+10,y+10):c=POINT(x-10,y+9)
173 xj  d=POINT(x+8,y-10):e=POINT(x+30,y+9):f=POINT(x+10,y+12)
174 Vd  g=POINT(x+10,y+24)
175 Xn  IF f=14 OR f=16 OR f=17 OR b=2 THEN tod
176 ff  IF spring=1 THEN sprung2
177 3G  IF g>13 AND g<18 OR a=-1 THEN tod
178 Ke  IF a<24 AND a<=9 OR a=13 AND NOT a<0 THEN u=1:r=0:GOTO m
        ove
179 xQ  IF a=11 THEN LINE (x,y+20)-(x+20,y+32),0,bf
180 ih  r=STICK(2): u=STICK(3):v=STRIG(3):w=STRIG(2)
181 3S  IF r<>0 AND u=1 AND schuss=1 THEN schies
182 YW  IF r=-1 AND c>=24 THEN r=0
183 jP  IF r=-1 AND c=11 THEN r=0
184 F1  IF r=1 AND e>=24 THEN r=0
185 KC  IF r=1 AND e=11 THEN r=0
186 wd  IF d=12 AND u<0 AND schl=0 THEN u=0
187 HX  IF d=12 AND w=-1 AND schl=0 THEN w=0
188 9N  IF d=12 AND v=-1 AND schl=0 THEN v=0
189 vt  IF r=-1 AND c=12 AND schl<=0 THEN r=0
190 qN  IF r=1 AND e=12 AND schl<=0 THEN r=0
191 Sj  IF w=-1 AND d>=24 OR d=11 THEN w=0
192 vf  IF w=-1 AND d=12 AND schl<=0 THEN w=0
193 1Y  IF v=-1 AND d=12 AND schl<=0 THEN v=0
194 tP  IF w=-1 AND r=0 THEN u=-1 : GOTO move
195 Go  IF v=-1 AND r<>0 AND d<24 AND d<>11 THEN Sprung
196 3E  IF r<>0 THEN u=0:GOTO move
197 ic  IF u<0 AND d>=24 THEN u=0
198 kk  IF u<0 AND d=11 THEN u=0
199 BL  IF u>0 AND a=10 THEN r=0:GOTO move
200 D8  IF u<0 AND b=9 THEN r=0:GOTO move
201 bV  GOTO scrolling
```

```
202 CX  move:
203 xS  FOR be=1 TO 4
204 Ap  PUT (x,y),p
205 xb  x=x+r*5 : y=y+u*5
206 Cr  PUT (x,y),p
207 t9  NEXT be
208 za  control:
209 2n  f=POINT(x+10,y+12):b=POINT(x+10,y+10)
210 9z  IF f=14 THEN CALL tusch:such=weiter:GOTO Aufbau
211 9b  IF b=10 THEN frage.found
212 gH  IF f=5 THEN schuss=1:TIMER ON:SOUND 500,4:PUT (x,y),peng
213 QZ  IF f=13 OR f=18 OR f=19 OR b=2 THEN tod
214 wU  IF f=25 THEN PUT (x,y),birne:leben=leben+1:tusch
215 FS  IF f=11 THEN SOUND 700,4:punkte=punkte+10:PUT (x,y),diamand
216 Gh  IF b=15 THEN schl=schl-1
217 de  IF f=4 THEN PUT(x,y),key:schl=schl+1:SOUND 1000,1:punkte=pu
        nkte+5
218 sm  GOTO scrolling
219 Id  Sprung:
220 ug  IF r=1 THEN f=POINT(x+30,y-9) ELSE f=POINT(x-10,y-9)
221 x7  IF f=12 AND schl<=0 THEN scrolling
222 ed  spring=1
223 Hm  FOR be=1 TO 4
224 U9  PUT (x,y),p
225 jv  y=y-5
226 pT  PUT (x,y),p:NEXT
227 2W  a=POINT (x+10,y+12)
228 kc  IF r=1 THEN f=POINT(x+30,y+9) ELSE f=POINT(x-10,y+9)
229 KF  IF f>=24 OR f=11 THEN spring=0:GOTO control
230 Uc  IF a=18 OR a=19 OR a=13 AND NOT POINT(x+10,y+10)=10 THEN to
        d
231 Pu  FOR be=1 TO 4
232 cH  PUT (x,y),p
233 qX  x=x+r*5
234 xb  PUT (x,y),p:NEXT
235 Sw  IF  r=1 THEN f=POINT(x+30,y+9) ELSE f=POINT(x-10,y+9)
236 Wj  IF f=12 AND schl<=0 THEN spring=0
237 Sc  GOTO control
238 v4  sprung2:
239 8q  spring=0:u=5
240 XH  IF r=1 THEN f=POINT(x+30,y+9):g=POINT(x+30,y+30):a=POINT(x+
        30,y+32) ELSE f=POINT(x-10,y+9):g=POINT(x-10,y+30):a=POINT(x
        -10,y+32)
241 ZO  IF f>=24 OR f=11 THEN GOTO Abfrage
242 G3  IF g>=24 OR g=11 OR g=12 THEN u=0
243 VI  FOR b=1 TO 4
244 oT  PUT (x,y),p
245 aF  x=x+r*5:y=y+u
246 9n  PUT (x,y),p:NEXT
247 cm  GOTO control
248 48  schies:
249 FG  IF r=-1 THEN a=x-10 ELSE a=x+30
250 GK  f=POINT(a,y+30)
251 hR  IF f=24 THEN
252 Uk  LINE (a-10,y+20)-(a+10,y+40),0,bf : zer=zer+1
253 WV  IF zer>100 THEN zer=1
254 Ag  px(zer)=a-10
255 O4  py(zer)=y+20
256 4x  END IF
257 VP  GOTO scrolling
258 bl  neu:
259 6D  SOUND 2000,1,70,2
260 u6  IF no=zer THEN 8989
261 1g  IF no<=100 THEN no=no+1
262 ma  IF no=101 THEN no=1
263 g2  PUT (px(no),py(no)),gift
264 AM  8989 RETURN
265 tZ  frage.found:
266 Py  PUT (x,y),frage
267 Ud  RANDOMIZE TIMER
268 Sa  frage.aktiv:
269 WO  punkte=punkte+20
270 zg  IF bonus=1 THEN
271 G3  PUT (x,y),key :bonus=0
272 9R  ELSEIF bonus=2 THEN
273 E7  PUT (x,y),diamand :bonus=0
274 GZ  ELSEIF bonus=3 THEN
275 Ou  PUT (x,y),birne :bonus=0
276 Nh  ELSEIF bonus=4 THEN
277 ge  such=quwei:tusch:GOTO Aufbau
278 5o  ELSE
279 OC  bonus=CINT(RND*2)+1
```

```
280 41  GOTO frage.aktiv
281 TM  END IF
282 uo  GOTO scrolling
283 rn  boden:
284 ZW  LINE (ix,iy)-(ix+18,iy+11),24,bf
285 BD  LINE (ix-1,iy)-(ix+9,iy+12),1,b
286 vP  LINE (ix+9,iy)-(ix+19,iy+12),1,b
287 GA  GOTO weiter
288 60  mauer:
289 Qr  LINE (ix,iy)-(ix+18,iy+19),24,bf
290 rW  LINE (ix,iy)-(ix+9,iy),1
291 p8  LINE (ix+9,iy)-(ix+9,iy+19),1
292 Hn  LINE (ix,iy+9)-(ix+9,iy+9),1
293 8j  LINE (ix+9,iy+4)-(ix+19,iy+4),1
294 zr  LINE (ix+9,iy+14)-(ix+19,iy+14),1
295 OI  GOTO weiter
296 ZB  bruch:
297 1R  FOR a=ix+2 TO ix+20 STEP 2
298 OY  LINE (a,iy+1)-(a-2,iy+11),11
299 uz  NEXT
300 TN  GOTO weiter
301 qp  feind1:
302 G4  CIRCLE (ix+10,iy+13),5,14,,,2.5
303 5L  PAINT (ix+10,iy+13),14
304 K2  LINE (ix+10,iy+8)-(ix+10,iy+3),15
305 dZ  LINE (ix+8,iy+3)-(ix+12,iy+3),15
306 HL  LINE -(ix+11,iy),15
307 h4  LINE -(ix+9,iy),15
308 RO  LINE -(ix+8,iy+3),15:PAINT (ix+10,iy+2),15
309 cW  GOTO weiter
310 r1  leiter:
311 Tq  LINE (ix,iy)-(ix+19,iy+19),10,bf
312 KJ  LINE (ix+1,iy)-(ix+3,iy+19),19,bf
313 Oe  LINE (ix+18,iy)-(ix+16,iy+19),19,bf
314 u1  LINE (ix+3,iy+4)-(ix+16,iy+6),19,bf
315 OI  LINE (ix+3,iy+14)-(ix+16,iy+16),19,bf
316 jd  GOTO weiter
317 Me  door:
318 Sy  LINE (ix+1,iy)-(ix+18,iy+19),12,bf
319 O2  LINE (ix+3,iy+10)-(ix+7,iy+11),7,bf
320 nh  GOTO weiter
321 CC  feind2:
322 v6  LINE (ix+1,iy+11)-(ix+18,iy+13),17,bf
323 x2  CIRCLE (ix+5,iy+16),2,16:PAINT(ix+5,iy+16),16
324 oY  CIRCLE (ix+14,iy+16),2,16:PAINT(ix+14,iy+16),16
325 Ca  LINE (ix+3,iy+11)-(ix+4,iy+2),17,bf
326 9Q  LINE (ix+16,iy+11)-(ix+15,iy+2),17,bf
327 uo  GOTO weiter
328 3j  stamper:
329 wb  LINE (ix,iy+13)-(ix+19,iy+19),16,bf
330 yu  LINE (ix+8,iy)-(ix+12,iy+13),16,bf
331 ys  GOTO weiter
332 32  stiehl:
333 JL  LINE (ix+8,iy)-(ix+12,iy+19),16,bf
334 1v  GOTO weiter
335 FH  platte:
336 zt  LINE (ix,iy+1)-(ix+18,iy+11),28,bf
337 8A  LINE (ix-1,iy)-(ix+19,iy+12),1,b
338 5z  GOTO weiter
339 Ud  tod:
340 pq  FOR a=y TO y+20 STEP 2
341 GI  LINE (x,a)-(x+19,a),0
342 mk  FOR b=0 TO 200:NEXT b,a
343 Xs  FOR a=y+1 TO y+20 STEP 2
344 JL  LINE (x,a)-(x+19,a),0
345 pn  FOR b=0 TO 200:NEXT b,a
346 pZ  leben=leben-1: such=runde
347 Ka  GOTO Aufbau
348 cK  SUB tusch STATIC
349 2v  RESTORE tusch
350 HB  FOR a=1 TO 7
351 Ra  READ ton,lang
352 FE  SOUND ton,lang/3
353 mr  NEXT
354 61  tusch:
355 JA  DATA 783.99,10,659.26,5,783.99,10,659.26,5
356 vl  DATA 783 .99,10,659.26,5,523.25,25
357 np  END SUB
(C) 1988 M&T
```

Listing 1. (Schluß)

Programmname:	Level
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

```
Programmautor: Roman Stumm
--------------------------

 1 FfO Level
 2 ar1  1
 3 k40 REM, 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 4 f4  REM, 2  0  0  2  0  0  0  0  2  2  12  12  2  0  0
 5 qy  REM, 2  2  12  2  2  5  2  3  6  11  12  12  2  5  3
 6 76  REM, 2  11  12  2  14  5  2  3  6  2  2  2  2  5  3
 7 TZ  REM, 2  5  12  2  2  5  2  3  6  11  6  0  0  5  3
 8 eU  REM, 2  5  12  2  14  5  2  3  6  2  2  0  0  5  3
 9 qo  REM, 2  5  12  2  2  5  2  3  2  2  2  2  15  2  3
10 Dz  REM, 2  5  0  0  0  5  2  14  0  0  0  0  0  14  7
11 sC  REM, 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
12 53  REM,          20          20
13 So  REM,          140         140         260          14
       0
14 XW  REM,          140         140         159          15
       9
15 xs  REM,          260         140         279          15
       9
16 Vr  REM,          140         140         260          14
       0
17 Ws  REM,          140         140         260          14
       0
18 F3  REM,          2           1           2
19 Xx  Level
20 uC1  2
21 AW0 REM, 2  1  1  1  1  1  1  1  1  1  1  1  1  1  2
22 kC  REM, 2  0  0  0  0  12  0  0  0  11  0  0  0  0  2
23 Dg  REM, 2  1  1  0  1  1  1  5  1  1  1  0  1  1  2
24 a3  REM, 2  0  0  0  0  0  0  5  0  0  0  0  0  0  2
25 4F  REM, 2  14  0  0  0  0  0  5  0  12  0  0  0  0  2
26 2j  REM, 2  1  1  1  1  1  1  1  1  1  15  1  2  0  2
27 Uu  REM, 2  0  0  0  0  0  0  0  0  0  0  0  0  2  0  2
28 pa  REM, 7  0  0  0  12  0  0  14  0  0  0  0  0  6  0  2
29 CW  REM, 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
30 NL  REM,          20          20
31 3R  REM,          20          80          140          14
       0
32 Ze  REM,          20          80          39           99
33 qp  REM,          140         140         159          15
       9
34 Lv  REM,          20          80          80           80
35 Ey  REM,          140         140         200          14
       0
36 YS  REM,          3           0           3
37 pF  Level
38 EX1  3
39 zS0 REM, 2  5  0  0  0  0  0  0  0  0  0  0  0  0  2
40 TS  REM, 2  5  0  12  12  0  0  0  0  0  0  0  12  11  2
41 kW  REM, 1  1  0  1  1  0  12  0  12  0  12  0  1  3  1
42 ew  REM, 0  0  0  0  0  0  1  0  1  0  1  0  0  0  0
43 e2  REM, 2  0  0  0  0  0  0  0  0  0  0  0  0  0  2
44 LO  REM, 2  0  9  13  0  0  0  0  8  0  12  0  0  0  2
45 DQ  REM, 2  18  1  2  1  15  1  15  2  15  1  15  1  15  2
46 D1  REM, 2  16  0  6  0  0  0  0  0  6  0  0  0  0  0  7
47 ds  REM, 2  17  3  1  3  3  3  3  1  3  3  3  3  3  2
48 fd  REM,          20          20
49 gp  REM,          20          120         60           10
       0
50 bc  REM,          20          120         39           15
       9
51 gj  REM,          60          100         79           11
       9
52 1e  REM,          20          40          20           12
       0
53 JP  REM,          60          100         140          10
       0
54 3v  REM,          4           1           4
55 7X  Level
56 Ys1  4
57 UE0 REM, 2  8  0  0  0  0  0  0  0  0  0  0  0  12  2
58 5E  REM, 11  12  0  0  0  0  0  12  12  0  0  0  0  12  11
59 4h  REM, 2  1  3  0  0  0  0  0  0  5  4  3  3  1  2
60 PU  REM, 2  0  0  15  1  0  0  0  0  5  2  15  15  15  2
```

Listing 2. Die Level-Dateien für Mad Factory

```
 61 uf  REM, 2  11  2  0  0  15  1  1  1  3  2  5  6  0  4
 62 wP  REM, 2  0  6  0  0  0  0  0  5  8  2  5  2  0  4
 63 HY  REM, 2  1  1  0  0  0  0  0  5  0  2  5  2  0  4
 64 Ra  REM, 2  14  0  0  0  0  12  0  5  0  6  5  2  0  7
 65 am  REM, 2  1  1  3  1  1  1  3  1  1  3  3  1  3  2
 66 XY  REM,          140          60
 67 WB  REM,          20           0            20          14
        0
 68 LY  REM,          20           0            39          19
 69 y1  REM,          20           140          39          15
        9
 70 Fb  REM,          20           0            240         0
 71 Xx  REM,          20           140          100         14
        0
 72 aG  REM,          5            4            3
 73 Pp  Level
 74 sD1  5
 75 wu0 REM, 2  0  0  0  0  11  6 -1
 76 hl  REM, 2  5  3  3  3  3  3  12  0  3  0  12  0  3  0
 77 qI  REM, 2  5  0  12  0  0  0  3  15  0  15  3  15  0  0
 78 9K  REM, 2  3  3  3  3  3  5  2  9  0  9  0  9  0  0
 79 CU  REM, 2  0  0  0  0  0  12  5  2 -1
 80 2O  REM, 2  15  5  1  0  0  0  2 -1
 81 Ew  REM, 2  12  5  0  0  0  0  2  3  0  0  0  0  0  3
 82 mS  REM, 2  4  3  4  18  0  8  7  3  0  3 -1
 83 6r  REM, 2  1  1  1  18  2  2  2  3  0  0  0  3  0  0
 84 3e  REM,          100          140
 85 o5  REM,          60           100          80          14
        0
 86 FI  REM,          60           100          79          11
        9
 87 6N  REM,          80           140          99          17
        9
 88 gu  REM,          60           100          120         10
        0
 89 6R  REM,          80           100          80          16
        0
 90 ub  REM,          6            4            3
 91 h7  Level
 92 CY1  6
 93 aJ0 REM, 12  11  2  12  8  2  12  0  0  0  4  12  0  0  2
 94 0j  REM, 9  12  6  0  0  6  0  0  0  0  0  0  0  14  2
 95 IB  REM, 2  1  1  5  2  1  1  3  1  1  1  1  1  5  2
 96 By  REM, 11  0  0  5  2  0  0  18  0  0  0  0  0  5  2
 97 PU  REM, 2  0  0  5  6  11  0  18  8  0  12  0  0  5  2
 98 BL  REM, 2  5  3  1  2  1  5  16  1  3  1  3  1  1  2
 99 vx  REM, 2  5  15  8  2  0  5  0  0  0  0  0  0  0  7
100 Cl  REM, 2  5  6  12  6  0  5  0  0  0  0  0  0  0  2
101 Kd  REM, 2  2  2  2  2  2  2  0  2  15  1  15  1  15  2
102 ds  REM,          120          20
103 gN  REM,          140          60           260         20
104 FW  REM,          140          60           159         11
        9
105 ds  REM,          260          20           279         39
106 up  REM,          140          40           140         12
        0
107 GB  REM,          120          20           260         20
108 AC  REM,          7            0            7
109 zP  Level
110 Wt1  7
111 IC0 REM, 12  0  0  0  0  0  0  10  0  16  16  16  16  16  16
112 Jb  REM, 0  0  0  0  0  12  0  17  1  6  6  6  6  6  7
113 C2  REM, 5  1  1  1  1  17  1  18  5  1  1  1  1  1  1
114 31  REM, 5  0  0  0  0  12  0  18  5  8  0  0  0  11  0
115 Fl  REM, 5  1  1  1  1  1  1  1  18  17  17  0  0  0  0  0
116 kf  REM, 5  0  0  0  0  12  0  18  11  11  17  17  15  1  17
117 Md  REM, 5  1  1  1  1  1  1  1  18  5  0  0  0  0  5  0
118 9g  REM, 5  0  0  0  0  12  0  18  5  14  0  0  0  5  11
119 NS  REM, 1  1  1  1  1  1  1  1  18  1  1  1  1  1  1  1
120 DD  REM, 120       140
121 N4  REM, 180       140          200          140
122 PX  REM, 180       140          199          159
123 3d  REM, 200       140          200          140
124 YJ  REM, 180       140          240          140
125 5f  REM, 200       140          200          140
126 B8  REM, 8         1            0
127 Hh  Level
128 qE1  8
129 m60 REM, 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
130 XZ  REM, 2  13  0  0  0  0  0  0  0  10  0  0  0  0  6
131 wI  REM, 2  1  1  1  1  1  1  1  1  1  1  1  1  1  2
132 PB  REM, 2  9  0  5  2  0  12  0  12  2  12  0  12  0  0
133 iQ  REM, 0  0  0  5  2  0  0  0  7  2  0  0  0  0  6
```

```
134 d2  REM, 0  15  2  2  2  1  1  1  1  1  1  1  1  2  2
135 yd  REM, 5  2  6  11  16  11  0  0  0  0  0  0  0  2  0
136 a7  REM, 5  0  2  6  0  0  13  0  14  14  0  0  0  6  0
137 wG  REM, 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
138 XJ  REM, 260       20
139 FS  REM, 20        20           180          140
140 q8  REM, 20        20           39           39
141 iq  REM, 180       140          199          159
142 Vd  REM, 20        20           260          20
143 rc  REM, 180       140          240          140
144 uw  REM, -1        0            0
(C) 1988 M&T
```

Listing 2. (Schluß)

Programmname:	levelEd
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

Programmautor: Roman Stumm

```
  1 LG0 ' Level Editor von Roman Stumm 1988
  2 GS  SCREEN 1,320,237,5,1
  3 t0  WINDOW 1,"       Mad Factory Level Editor",,0,1
  4 pW  PALETTE 0,0,0,0: PALETTE 1,.5,.5,.5
  5 po  PALETTE 2,.7,.6,.55: PALETTE 3,.1,.3,.7
  6 Na  PALETTE 4,1,1,1: PALETTE 5,.86,.86,0
  7 wP  PALETTE 6,1,0,0: PALETTE 7,.93,.73,0
  8 AX  PALETTE 8,0,.83,.99: PALETTE 9,0,0,0
  9 5U  PALETTE 10,0,0,0: PALETTE 11,.72,.45,0
 10 Da  PALETTE 12,.4,.02,0: PALETTE 13,1,0,1
 11 G4  PALETTE 14,0,1,.2: PALETTE 15,.36,.36,.36
 12 ge  PALETTE 16 ,.95,.8,0: PALETTE 24,.75,.05,.05
 13 tv  PALETTE 25,.55,.55,.55: PALETTE 26,.62,.62,.62
 14 bq  PALETTE 27,.68,.68,.68: PALETTE 28,.75,.75,.75
 15 On  PALETTE 29,.8,.8,.8: PALETTE 30,.9,.9,.9
 16 5A  CLEAR ,30000&:ON ERROR GOTO fehler
 17 cq  DIM tile(110,20)
 18 fW  MENU 1,0,1," Projekt"
 19 H8  MENU 2,0,1,"  "
 20 KC  MENU 3,0,1,"  "
 21 NG  MENU 4,0,1,"  "
 22 8J  MENU 1,1,1," SAVE"
 23 H6  MENU 1,2,1," LOAD"
 24 Um  MENU 1,3,1," NEW"
 25 P3  MENU 1,4,1," RASTER"
 26 15  MENU 1,5,1," SPECIAL"
 27 ca  MENU 1,6,1," QUIT"
 28 em  ON MENU GOSUB PullDown
 29 LI  now=1:altx=-1:alty=0:levelf=-10:levela=-10
 30 C4  FOR zahl=1 TO 20
 31 w0  ON zahl GOSUB null,boden,mauer,bruch,stein,leiter,door,ausg
        ang,frage,birne,peng,key,diamand,schraube,wagen,gift,stamper
        ,platte,stiehl,du
 32 5N  GET(0,0)-(19,19),tile(110,zahl-1)
 33 tm  CLS:NEXT
 34 Ze  GOSUB Raster
 35 pP  GOSUB aufbau
 36 7D  warte:
 37 pF  MENU ON
 38 BG  WHILE MOUSE(0)=0
 39 Kv1  x=MOUSE(1):y=MOUSE(2)
 40 Wy   IF y<180 THEN
 41 Js2   PUT(x-10,y-5),tile(110,now)
 42 Kt    PUT(x-10,y-5),tile(110,now)
 43 dW1  END IF
 44 9x0 WEND
 45 HO  MENU STOP
 46 U2  IF x>=300 THEN warte
 47 Uy  IF y>=180 THEN GOTO schalter
 48 Cc  puttile:
 49 7a  x=FIX(x/20):y=FIX(y/20)
 50 yp  x=x*20:y=y*20
 51 wR  IF now=19 AND altx<>-1 THEN PUT(altx,alty),tile(110,19)
 52 Kg  IF x=altx AND y=alty THEN altx=-1
 53 NG  IF now<19 THEN PUT(x,y),tile(110,now),PSET ELSE PUT(x,y),t
        ile(110,19): altx=x:alty=y
```

```
 54 Fu  GOTO warte
 55 UQ  schalter:
 56 iv  ix=MOUSE(1)
 57 gB  ix=FIX(ix/20):y=FIX(y/20):y=y*20
 58 UU  IF y=180 THEN now=ix
 59 Ka  IF y=200 AND ix<5 THEN now=15+ix
 60 kr  IF y=200 AND ix>=5 THEN klick
 61 QE  a=MOUSE(0)
 62 N2  GOTO warte
 63 Kb  klick:
 64 q3  ix=MOUSE(1)
 65 pU  IF ix<160 AND ix>120 THEN ClearLastScreen
 66 n7  IF ix<220 AND ix>180 THEN info
 67 Ws  IF ix<280 AND ix>240 THEN move
 68 T8  GOTO warte
 69 iY  ClearLastScreen:
 70 sf  CALL weg3:GOSUB Raster:altx=-1:a=MOUSE(0):GOTO warte
 71 wY  NEU:
 72 4x  CALL weg3:GOSUB Raster
 73 Km  altx=-1:move=0
 74 kh  FOR n=0 TO 1
 75 xS1  xf(n)=0:yf(n)=0:sxf(n)=0:syf(n)=0
 76 LD   zxf(n)=0:zyf(n)=0:axf(n)=0:ayf(n)=0
 77 s5   exf(n)=0:eyf(n)=0:NEXT
 78 ShO RETURN warte
 79 M7  koordinaten:
 80 SJ  zahl%=0:b%=190
 81 we  FOR c=1 TO 3
 82 Ab1  FOR a%=0 TO 90 STEP 20
 83 yu2   CALL cursor (zahl%,b%):WRITE a%
 84 S3    zahl%=zahl%+20
 85 yS0 NEXT a%,c
 86 kM  RETURN
 87 Lg  move:
 88 sI  CALL weg :COLOR 11,0 :GOSUB koordinaten
 89 Jq  COLOR 17,0:LOCATE 25,1
 90 J1  PRINT "Anfangskoordinaten <Ctrl-E = EXIT>"
 91 1y  FOR n=0 TO 1
 92 RJ1  LOCATE 26,1: COLOR 16+n
 93 q0   PRINT "Start Objekt"n+1"(X1,Y1)"
 94 pu2   GOSUB GETZAHL
 95 8Q1  xf(n)=VAL(b$):yf(n)=VAL(c$)
 96 JG   CIRCLE (xf(n),yf(n)),2,16+n:PAINT(xf(n),yf(n)),16+n
 97 AV0 NEXT n
 98 85  FOR n=0 TO 1
 99 ka1  nochmals:
100 9R   CALL weg2:COLOR 16+n:LOCATE 25,1
101 Au   PRINT "Objekt"n+1"getten (X1,Y1,X2,Y2)"
102 1x   COLOR 16
103 y32   GOSUB GETZAHL
104 ca1  axf(n)=VAL(b$):exf(n)=VAL(d$)
105 pr   ayf(n)=VAL(c$):eyf(n)=VAL(e$)
106 rn   IF (exf(n)-axf(n))*(eyf(n)-ayf(n))>2000 THEN nochmals
107 3d   LINE(axf(n),ayf(n))-(exf(n),eyf(n)),16+n,b
108 Lg0 NEXT n
109 JG  FOR n=0 TO 1
110 Jb1  CALL weg2:COLOR 16+n:LOCATE 25,1
111 Le   PRINT "Objekt"n+1"bewegen (X1,Y1,X2,Y2)"
112 v7   COLOR 16
113 8D2   GOSUB GETZAHL
114 2d1  sxf(n)=VAL(b$):zxf(n)=VAL(d$)
115 Fu   syf(n)=VAL(c$):zyf(n)=VAL(e$)
116 eU   LINE(sxf(n),syf(n))-(zxf(n),zyf(n)),16+n,b
117 Up0 NEXT n
118 Ey  move=1:CALL weg:GOSUB aufbau:a=MOUSE(0):GOTO warte
119 Iw  GETZAHL:
120 bb  a$="":b$="":c$="":d$="":e$="":zahl=0:a=0:f$=""
121 1y  WHILE NOT a=13
122 Sp2   a$=INPUT$(1):a=ASC(a$)
123 W9    IF a=5 THEN
124 LJ4      CALL weg :CLOSE
125 Ne       GOSUB aufbau:a=MOUSE(0):RETURN warte
126 yr2   END IF
127 eX    IF a$="," THEN zahl=zahl+1:f$=""
128 Jd    IF a=8 AND f$="" THEN skip
129 i0    IF a=8 THEN
130 It3    f$=LEFT$(f$,((LEN(f$)-1)+ABS(LEN(f$)-1))/2)
131 qP     ELSEIF a$<>"," THEN
132 F42   f$=f$+a$
133 5y    END IF
134 IQ1  IF NOT a=13 THEN PRINT a$;
135 AV   skip:
136 Qr    IF zahl=0 AND a$<>"," THEN
137 Ru2    b$=f$
138 Dr     ELSEIF zahl=1 AND a$<>"," THEN
139 Vz     c$=f$
140 Jy     ELSEIF zahl=2 AND a$<>"," THEN
141 Z4     d$=f$
142 P5     ELSEIF zahl=3 AND a$<>"," THEN
143 d9     e$=f$
144 G91   END IF
145 ma0 WEND
146 e7  CALL weg2
147 jL  RETURN
148 jw  aufbau:
149 z6  FOR a=0 TO 14
150 v71  x=a*20
151 Y5   PUT (x,182),tile(110,a),PSET
152 Xc0 NEXT
153 GP  FOR a=15 TO 19
154 gc1  x=a*20-300
155 AV   PUT(x,202),tile(110,a),PSET
156 cf0 NEXT:COLOR 18,14
157 vm  LINE (120,202)-(160,222),14,bf
158 e2  CALL cursor (125,213):PRINT" CLS"
159 82  LINE (180,202)-(220,222),14,bf
160 NZ  CALL cursor (185,213):PRINT "INFO"
161 LI  LINE (240,202)-(280,222),14,bf
162 ey  CALL cursor (245,213):PRINT "MOVE"
163 zb  RETURN
164 7x  Raster:
165 ce  FOR a=0 TO 280 STEP 20
166 YT1  FOR b=0 TO 160 STEP 20
167 W52   LINE (a,b)-(a+20,b+20),31,b
168 Fx0 NEXT b,a:RETURN
169 nQ  PullDown:
170 o7  a=MENU(1):b=MENU(0):MENU STOP
171 4J  ON a GOTO SPEICHERN,LADEN,NEU,Raster,SPEZIAL,QUIT
172 Hk  QUIT:
173 iv  CALL weg:COLOR 16,0:LOCATE 25
174 Y9  PRINT"Quit: Wirklich(j/n)?"
175 oC  IF UCASE$(INPUT$(1))="J" THEN NEW ELSE CALL weg:GOSUB aufba
        u:RETURN warte
176 oJ  SUB weg3 STATIC
177 0Z  LINE (0,0)-(300,180),0,bf
178 uw  END SUB
179 1u  SPEZIAL:
180 Vc  CALL weg:COLOR 11,0:LOCATE 24
181 j3  GOSUB koordinaten:COLOR 14
182 H2  PRINT"Anfangsposition der Spielfigur eingeben":COLOR 17
183 2V  PRINT"<Crtl-E = EXIT> X-Wert,Y-Wert: "
184 HM1  GOSUB GETZAHL
185 vV0 IF altx<>-1 THEN PUT(altx,alty),tile(110,19)
186 uF  altx=VAL(b$):alty=VAL(c$)
187 1M  PUT(altx,alty),tile(110,19)
188 2z  CALL weg:GOSUB aufbau:RETURN warte
189 ak  info:
190 bI  CALL weg:LOCATE 25
191 da  FOR n=0 TO 1
192 ob1  COLOR 16+n,0
193 Uy   PRINT"Start Objekt"n+1 xf(n) yf(n)
194 tq   CIRCLE (xf(n),yf(n)),2,16+n:PAINT(xf(n),yf(n)),16+n
195 6M0 NEXT:a$=INPUT$(1):CALL weg:LOCATE 25
196 if  FOR n=0 TO 1
197 j41  COLOR 16+n
198 1Z   PRINT"Getten"n+1 axf(n) ayf(n) exf(n) eyf(n)
199 P5   LINE (axf(n),ayf(n))-(exf(n),eyf(n)),16+n,b
200 BR0 NEXT:a$=INPUT$(1):CALL weg:LOCATE 25
201 nk  FOR n=0 TO 1
202 o91  COLOR 16+n
203 ju   PRINT"Objekt"n+1 sxf(n) syf(n)"bis" zxf(n) zyf(n)
204 CR   LINE (sxf(n),syf(n))-(zxf(n),zyf(n)),16+n,b
205 GW0 NEXT:a$=INPUT$(1):CALL weg:LOCATE 25
206 cI  IF altx<>-1 THEN PRINT"Position Spielfigur:"altx alty :LI
        NE (altx,alty)-(altx+19,alty+19),14,b
207 yu  PRINT"Bedeutung Fragezeichen:";
208 bV  IF frage=1 THEN
209 4I1  PRINT"Schlüssel"
210 G3   ELSEIF frage=2 THEN
211 OK   PRINT"Diamand"
212 NB   ELSEIF frage=3 THEN
213 ZH   PRINT"Bonuslife"
```

Listing 3. Mit dem »LevelEd« bauen Sie Ihre eigenen Fabrikhallen

```
214 UJ   ELSEIF frage=4 THEN
215 ji   PRINT"zu Level" levelf
216 5o   ELSE
217 Pn   PRINT"Zufall"
218 SLO END IF
219 tk   a$=INPUT$(1):CALL weg:LOCATE 25
220 HL   PRINT"Nummer nächster Level bei Ausgang:"
221 bK   IF levela>-10 THEN PRINT ,levela ELSE PRINT"Keine Angabe!"
222 vJ   a$=INPUT$(1):CALL weg:GOSUB aufbau:a=MOUSE(0):GOTO warte
223 D5   LADEN:
224 WP   CALL weg3:GOSUB Raster
225 mE   altx=-1:move=0
226 dl   CALL weg:LOCATE 24,1:COLOR 17,0
227 d9   PRINT"Load: <RETURN = EXIT>"
228 bm   PRINT"Dateinamen:"
229 Pu   LINE INPUT a$
230 Ry   IF a$="" THEN CALL weg:GOSUB aufbau:RETURN warte
231 yU   OPEN a$ FOR APPEND AS #1
232 Pk   IF LOF(1)=0 THEN
233 bm1   CALL weg
234 63    LOCATE 25:COLOR 16
235 82    PRINT"Datei nicht gefunden..."
236 1o    CLOSE #1:a$=INPUT$(1)
237 lc    CALL weg:GOSUB aufbau
238 2H    RETURN warte
239 je0 END IF:CLOSE #1
240 OK   OPEN a$ FOR INPUT AS #1
241 Q7   CALL weg:LOCATE 25
242 rs   PRINT "< Crtl-E = EXIT >
243 nP   PRINT "Level Nr: ";
244 HL   GOSUB GETZAHL:such=VAL(b$)
245 fb   schauen:
246 vG   WHILE NOT a$="Level" AND NOT EOF(1)
247 yY   INPUT #1,a$:WEND
248 OG   IF a$<>"Level" THEN
249 fB1   LOCATE 25,10:PRINT"Level nicht enthalten"
250 VD    a$=INPUT$(1)
251 zq    CALL weg:GOSUB aufbau
252 I5    CLOSE #1
253 HW    RETURN warte
254 m00 END IF:a$=""
255 bj   INPUT #1,level
256 wW   IF level<>such THEN schauen
257 Zu   FOR iy=0 TO 160 STEP 20
258 ko1   INPUT #1,a$
259 nA2    FOR ix=0 TO 280 STEP 20
260 Xc3     INPUT #1,a :IF a=-1 THEN freizeile
261 800 IF a<>0 THEN PUT(ix,iy),tile(110,a) ELSE PUT(ix,iy),tile(
        110,0),PSET
262 vb1   NEXT ix
263 G9    freizeile:
264 Oh0 NEXT iy
265 Q2   INPUT #1,a$,altx,alty:PUT(altx,alty),tile(110,19)
266 y7   INPUT #1,a$,xf(0),yf(0),xf(1),yf(1)
267 ro   FOR n=0 TO 1
268 UZ1   INPUT #1,a$,axf(n),ayf(n),exf(n),eyf(n)
269 QV0 NEXT
270 ur   FOR n=0 TO 1
271 921   INPUT #1,a$,sxf(n),syf(n),zxf(n),zyf(n)
272 TY0 NEXT
273 oq   INPUT #1,a$,levela,frage,levelf
274 eR   CLOSE #1
275 G9   move=1 :CALL weg:GOSUB aufbau
276 et   RETURN warte
277 31   SPEICHERN:
278 k2   IF move=0 OR altx=-1 THEN warte
279 RX   CALL weg :COLOR 17,0: LOCATE 24,1
280 Ip   PRINT"Save: <RETURN> = EXIT>"
281 GG   PRINT"Dateinamen:":COLOR 16
282 L1   LINE INPUT datei$
283 7P   IF datei$="" THEN CALL weg:GOSUB aufbau:RETURN warte
284 OX   CALL weg:COLOR 17:LOCATE 25,1:GOSUB Raster
285 aj   PRINT"Ctrl-E = EXIT"
286 qu   PRINT"Level Nr:"
287 wh1   GOSUB GETZAHL:level=VAL(b$)
288 hf0 LOCATE 25,1:PRINT"Nächster Level bei Ausgang:"
289 xp   PRINT "Nr: "; : GOSUB GETZAHL
290 Hh   levela=VAL(b$)
291 eR   nochmal2:
292 7N   LOCATE 25,1:COLOR 16
293 Qn   PRINT"Bedeutung des Fragezeichens (0=Zufall)"
294 Gu   PRINT"1=Schlüssel;2=Diamand;3=Bonus;4=Ausgang"
295 SO1   GOSUB GETZAHL:frage=VAL(b$)
```

```
296 ZC0 IF frage<0 OR frage>4 THEN CALL weg2: GOTO nochmal2
297 K2   IF frage=4 THEN
298 JV1   LOCATE 25,1:COLOR 17
299 kG    PRINT"Nächster Level bei Fragezeichen:"
300 FV2    GOSUB GETZAHL:levelf=VAL(b$)
301 SB1   ELSE
302 pc    levelf=0
303 jw0 END IF:CALL weg
304 eD   OPEN datei$ FOR APPEND AS #1
305 AB   IF LOF(1)=0 THEN NoCheck ELSE CLOSE #1:OPEN datei$ FOR INP
         UT AS #1
306 TB   OPEN "ram:Daten" FOR OUTPUT AS #2
307 Of   such:
308 Uy   a$=""
309 QQ   WHILE NOT EOF(1)
310 4y1   LINE INPUT #1,a$:PRINT #2,a$
311 gW    IF a$="Level" THEN
312 DM2    INPUT #1,a:PRINT #2,a
313 3Y3     IF a=level THEN
314 SQ4      LOCATE 25:COLOR 17
315 hn       PRINT"Level in Datei überschreiben(j/n)"
316 8X       b$=UCASE$(INPUT$(1))
317 Vb       IF b$<>"J" THEN CLOSE #1:CLOSE #2:KILL"ram:Daten":C
             ALL weg:GOSUB aufbau:RETURN warte
318 da       GOSUB weiter:GOTO weiter2
319 5y3     END IF
320 1U2    ELSE
321 mi     GOTO such
322 811    END IF
323 4h0 WEND:CLOSE #1:CLOSE #2:KILL"ram:Daten"
324 2c   OPEN datei$ FOR APPEND AS #2
325 Mq   PRINT #2,"Level"
326 DJ   PRINT #2,level
327 XX   GOSUB weiter:CLOSE #2
328 IF   CALL weg:GOSUB aufbau:RETURN warte
329 Wr   weiter:
330 1m   CALL weg:LOCATE 25: PRINT"Bitte warten..."
331 5g   PUT(altx,alty),tile(110,19)
332 m7   FOR iy=0 TO 160 STEP 20
333 2t   PRINT #2,"REM ,";
334 0N   FOR ix=0 TO 280 STEP 20
335 ax1   a=POINT(ix+10,iy+10)
336 ec    b=POINT(ix+10,iy+8)
337 cK    c=POINT(ix+5,iy+14)
338 Zb    p=FIX((a*100+b*10+c)/10)
339 Kw    RESTORE:a=-1:b=-1
340 5Q2    WHILE a<>p
341 jv     b=b+1 : READ a : WEND
342 Pk1   PRINT #2,b;
343 Wm0 NEXT ix:PRINT #2,"":NEXT iy
344 F1   a$="REM ,"
345 0N   PRINT #2,a$,altx,alty
346 jq   PRINT #2,a$,xf(0),yf(0),xf(1),yf(1)
347 96   FOR n=0 TO 1
348 FI1   PRINT #2,a$,axf(n),ayf(n),exf(n),eyf(n)
349 in0 NEXT
350 C9   FOR n=0 TO 1
351 ul1   PRINT #2,a$,sxf(n),syf(n),zxf(n),zyf(n)
352 lq0 NEXT
353 ZZ   PRINT #2,a$,levela,frage,levelf
354 S3   PUT(altx,alty),tile(110,19)
355 5h   RETURN
356 W9   DATA 0,264,266,110,275,111,133,144,99,55,66,77,91
357 8W   DATA 155,1,11,177,308,176
358 Hu   weiter2:
359 Yi   FOR a=1 TO 16
360 kW1   LINE INPUT #1,a$
361 uz0 NEXT
362 HH   WHILE NOT EOF(1)
363 vp1   LINE INPUT #1,a$:PRINT #2,a$
364 Tm0 WEND:CLOSE #1:CLOSE #2
365 Yu   OPEN "ram:Daten" FOR INPUT AS #1
366 6i   OPEN datei$ FOR OUTPUT AS #2
367 MM   WHILE NOT EOF(1)
368 se1   LINE INPUT #1,a$
369 02    PRINT #2,a$
370 940 WEND:CLOSE #1
371 JP   KILL"ram:Daten":CLOSE #2
372 wn   CALL weg:GOSUB aufbau
373 DS   RETURN warte
374 YY   NoCheck:
375 H4   CLOSE #1
376 Gs   OPEN datei$ FOR OUTPUT AS #2
```

```
377 Cg  PRINT #2,"Level"
378 39  PRINT #2,level
379 NN  GOSUB weiter:CLOSE #2
380 4v  CALL weg:GOSUB aufbau
381 La  RETURN warte
382 4Y  SUB weg2 STATIC
383 iF  LINE (0,191)-(340,240),0,bf
384 EG  END SUB
385 tp  SUB weg STATIC
386 fG  LINE (0,181)-(340,240),0,bf
387 HJ  END SUB
388 j5  ' Das SUB-Programm CURSOR ist aus dem AMIGA MAGAZIN,
389 bz  ' Ausgabe 6/88, Seite 115: "LOCATE: Auf den Punkt gebracht"
        .
390 vz  ' Ich habe es schon oft verwenden können:
391 Rq  SUB cursor(a%,b%) STATIC
392 ek  xadr&=WINDOW(8)+36:yadr&=WINDOW(8)+38
393 US  POKEW xadr&,a%
394 cc  POKEW yadr&,b%
395 PR  END SUB
396 Zy  null:
397 HY  LINE (0,0)-(20,20),31,b
398 mO  RETURN
399 jf  boden:
400 ZC  LINE (0,0)-(18,11),24,bf
401 8n  LINE (-1,0)-(9,12),1,b
402 Ww  LINE (9,0)-(19,12),1,b
403 rT  RETURN
404 ys  mauer:
405 2n  LINE (0,0)-(18,19),24,bf
406 mG  LINE (0,0)-(9,0),1
407 ki  LINE (9,0)-(9,19),1
408 PS  LINE (0,9)-(9,9),1
409 zL  LINE (9,4)-(19,4),1
410 gg  LINE (9,14)-(19,14),1
411 zb  RETURN
412 R3  bruch:
413 NE  FOR a=2 TO 20 STEP 2
414 og  LINE (a,1)-(a-2,11),11
415 mr  NEXT
416 4g  RETURN
417 gM  schraube:
418 XJ  CIRCLE (10,13),5,14,,,2.5
419 Xv  PAINT (10,13),14
420 4t  LINE (10,8)-(10,3),15
421 UK  LINE (8,3)-(12,3),15
422 j4  LINE -(11,0),15
423 mL  LINE -(9,0),15
424 Cf  LINE -(8,3),15:PAINT (10,2),15
425 Dp  RETURN
426 jt  leiter:
427 F1  LINE (0,0)-(19,19),10,bf
428 pL  LINE (1,0)-(3,19),19,bf
429 hM  LINE (18,0)-(16,19),19,bf
430 NO  LINE (3,4)-(16,6),19,bf
431 ev  LINE (3,14)-(16,16),19,bf
432 Kw  RETURN
433 EW  door:
434 OC  LINE (1,0)-(18,19),12,bf
435 wO  LINE (3,10)-(7,11),7,bf
436 OO  RETURN
437 Xd  wagen:
438 HV  LINE (1,11)-(18,13),17,bf
439 XK  CIRCLE (5,16),2,16:PAINT(5,+16),16
440 XM  CIRCLE (14,16),2,16:PAINT(14,16),16
441 8U  LINE (3,11)-(4,2),17,bf
442 YM  LINE (16,11)-(15,2),17,bf
443 V7  RETURN
444 vb  stamper:
445 y4  LINE (0,13)-(19,19),16,bf
446 H4  LINE (8,0)-(12,13),16,bf
447 ZB  RETURN
448 vu  stiehl:
449 cV  LINE (8,0)-(12,19),16,bf
450 cE  RETURN
451 79  platte:
452 cK  LINE (0,1)-(18,11),28,bf
453 2K  LINE (-1,0)-(19,12),1,b
454 gI  RETURN
455 Xf  peng:
456 eZ  LINE (9,2)-(5,13),6
457 hn  LINE -(15,11),6: LINE -(11,17),6
458 4s  LINE -(15,17),6: LINE -(19,7),6
```

```
459 gU  LINE -(10,9),6: LINE -(12,2),6
460 Ee  LINE -(9,2),6:PAINT(10,10),6
461 nP  RETURN
462 DO  birne:
463 AP  CIRCLE (10,13),5,26,,,2.15:PAINT(10,13),26
464 5f  CIRCLE (10,6),6,5:PAINT(10,6),5
465 rT  RETURN
466 QZ  diamand:
467 qD  LINE (10,1)-(1,10),31
468 kA  LINE -(10,19),31: LINE -(19,10),31
469 7h  LINE -(10,1),31: LINE -(7,10),31
470 Uo  LINE -(10,19),31: LINE -(13,10),31
471 Fs  LINE -(10,1),31
472 3Y  PAINT(10,10),8,31:PAINT(6,10),8,31:PAINT(16,10),8,31
473 zb  RETURN
474 Jv  frage:
475 tt  LINE (0,0)-(19,19),9,bf
476 XL  CIRCLE (10,7),5,14,4.71,2.57
477 Bb  LINE (10,12)-(10,15),14
478 ND  CIRCLE (10,18),1,14:PSET (10,18),14
479 5h  RETURN
480 LW  key:
481 C4  CIRCLE (10,5),4,7
482 73  CIRCLE (10,5),3,7
483 AP  LINE (9,9)-(10,17),7,bf
484 m9  LINE (11,17)-(13,17),7
485 Sj  LINE (11,14)-(13,14),7
486 Co  RETURN
487 Uc  gift:
488 O6  LINE (0,5)-(18,11),24,bf
489 aq  LINE (0,0)-(19,5),14,bf
490 H7  FOR a=2 TO 18 STEP 4
491 EH1  CIRCLE (a,0),2,0:PAINT(a,0),0
492 7F0 NEXT a
493 cs  LINE (10,6)-(10,11),1
494 Kw  RETURN
495 Sp  stein:
496 5y  FOR a=8 TO 2 STEP -1
497 Jk1  CIRCLE (10,10),a,a+23
498 7C0 NEXT
499 Cu  PAINT (10,10),a+24
500 Q2  RETURN
501 Ip  ausgang:
502 UN  CIRCLE (10,7),7,13,,,1.35:PAINT(10,7),13
503 1N  LINE (5,5)-(0,19),13
504 8J  LINE -(19,19),13
505 Sf  LINE -(15,5),13 :PAINT(9,16),13
506 W8  RETURN
507 A3  du:
508 XU  LINE (6,9)-(0,19),3
509 nX  LINE -(19,19),3
510 jd  LINE -(13,9),3
511 Jj  LINE -(6,9),3:PAINT (10,12),3
512 Im  CIRCLE (10,4),5,2:PAINT(10,4),2
513 w4  CIRCLE (8,3),1,4:PSET(8,3),1
514 it  CIRCLE (12,3),1,4:PSET(12,3),1
515 KL  LINE (8,6)-(12,6),24
516 8E  PSET (4,3),24:PSET (4,4),24
517 hM  PSET (4,5),24:PSET (16,3),24
518 OG  PSET (16,4),24:PSET(16,5),24
519 jO  PSET (3,4),24:PSET (17,4),24
520 pZ  PSET (10,11),4:PSET(10,17),4
521 Cy  PSET (10,14),4
522 mO  RETURN
523 OJ  fehler:
524 2Q  IF ERR=62 THEN
525 2N  RESUME NEXT
526 vB  ELSEIF ERR=4 THEN
527 VO  CALL weg:LOCATE 25:BEEP
528 R5  PRINT"Fehler in Bildaufbau!!"
529 z8  PRINT"Bitte auch Leveldatei überprüfen!!"
530 fT  CLOSE:a$=INPUT$(1):CALL weg:GOSUB aufbau:RESUME warte
531 gE  ELSEIF ERR=13 THEN
532 aT  CALL weg:LOCATE 25:BEEP
533 du  PRINT"Lesefehler - Leveldatei überprüfen!"
534 jX  CLOSE:a$=INPUT$(1):CALL weg:GOSUB aufbau:RESUME warte
535 Ex  ELSE
536 A7  ERROR ERR
537 bU  END IF
(C) 1988 M&T
```

Listing 3. (Schluß)

Gibber – tödliche Linien

Es ist fast aus. Ich bin eingekreist. Zu viele Linien — und ich bin mittendrin. Es gibt nur noch eine Chance: an der richtigen Stelle durch die feindlichen Linien. Wo nur eine Linie ist, könnte ich es schaffen...

Eigentlich gibt es nicht viel zu sehen. Ein paar Linien, ein kleines sternförmiges Ding, ein Rahmen — mehr nicht. Doch die Linien sind tödlich. Sie sind der »Gibber«. Den Stern steuert der Spieler mit seinem Joystick. Bloß keine Linie damit berühren! Am Anfang ist es noch leicht, dem Gibber zu entkommen. Er wird während eines Versuchs immer kleiner. Wenn er zu einem Punkt geschrumpft ist, läßt er sich fangen, und dann geht's mit dem nächsten Level weiter. Bei jedem Versuch wird es schwieriger. Der Gibber ist nämlich jedes Mal ein bißchen größer. Noch ist er zu schaffen, man kann ihn auf einen Punkt zusammenschrumpfen lassen und ihn fangen. Aber bald wird er so groß sein, daß der Stern eingekreist ist. Und dann heißt es aufpassen. Eine Berührung mit dem Gibber ist erträglich, aber mehrere bedeuten: Ein Leben weniger. »Press fire«. Nach vier Leben ist es dann vorbei: »Game over«.

Gibbers und Stars im Streit

Alles klar? Auf einem fremden Planeten im Universum leben die Gibbers und die Stars. Die Gibbers sind hungrige Lebewesen, die rasch wachsen und die Stars mit Vorliebe aussaugen — und damit vernichten. Wenn ein Gibber versucht, einen Star zu fangen, wird er kleiner. Er verbraucht dabei glücklicherweise eine Menge Energie. Benötigt der Gibber zu lange, um den Star zu fangen, wird der große Gibber zu einem Zwerg und ist des Todes, wenn er von einem Star gefressen wird. Er versucht dann immer noch, den Star zu fangen, hat aber keine Chance diesen letzten Versuch zu überleben.

Der Spieler steuert mit dem Joystick seinen Star. Nachdem er auf den Feuerknopf gedrückt hat, muß er durch geschickte Bewegungen den Gibber möglichst schnell zum Schrumpfen bringen, um ihn endlich fangen zu können, wenn er auf einen einzigen Punkt zusammengeschrumpft ist. Die Gibbers werden aber von Level zu Level größer. Sie können den Star übrigens auch vor dem Drücken des Feuerknopfes bewegen, um eine möglichst gute Startposition zu bekommen. Mit <ESC> beendet man das Spiel; mit <HELP> läßt sich das laufende Spiel anhalten — es wird dann mit <SPACE> wieder fortgeführt. Mit <SPACE> wird natürlich auch ein neues Spiel gestartet, wenn alle Leben der Stars verbraucht sind.

Level, Punktzahl und verbleibende Leben sind am unteren Bildschirmrand zu sehen — man hat ohne Zeitbegrenzung vier Leben, bevor das Spiel zu Ende ist.

Wollen Sie diesen spannenden Wettkampf zwischen Gibbers und Stars auch einmal miterleben? Dann tippen Sie einfach den Basic-Lader für »Gibber« ab und speichern ihn vor dem Start (wichtig) auf Diskette. Nach dem Start mit »RUN« erzeugt der Basic-Lader nun das eigentliche Programm »Gibber«.

Um das Programm zu starten, geben Sie im CLI »Gibber« ein. Sie können ihm aber auch mit

```
copy icon.info gibber.
info
```

ein Icon zuweisen, wobei »icon.info« für den Namen eines beliebigen Programm-Icons steht. Wichtig: Gibber läuft nur mit 512 KByte Speicher. Wer mehr Speicher hat, sollte vor dem Start des Spieles auf jeden Fall »NoFastMem« verwenden, weil es sonst einen Systemabsturz gibt! Um die Tipparbeit zu verringern, wurde das Programm gepackt. Deshalb flackert es auch nach dem Laden kurz — der Entpacker muß das Programm erst auf die volle Länge bringen.

Gibber hat auch eine wirklich gelungene Titelmusik. Leider hätten Sie aber über 80 KByte in Hexadezimalzahlen abtippen müssen, um sie zu hören — und das dürfte selbst dem hartnäckigsten Tipper zuviel sein. Wer das Spiel mit der Titelmusik haben möchte, findet diese Version auf der Leserservice-Diskette.

Und wenn Sie dann eines Abends endlich am Spielen sind: Vergessen Sie nicht, daß auch lange Winternächte einmal zu Ende gehen.

(A. Lietz/rs)

Programmname:	Gibber_Gen
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic
Bemerkung:	Erzeugt lauffähiges Assembler-Programm

Programmautoren: H.J. Nett und Ralf Melles

```
 1 0mO REM Generiert lauffähiges Programm
 2 ag  CLS
 3 XG  OPEN "df2:gibber.basic" FOR OUTPUT
       AS 1
 4 BS  READ anz
 5 oa  FOR i=1 TO anz
 6 3n1  READ h$
 7 yB2   wert1=ASC(LEFT$(h$,1))
 8 bP    IF wert1>64 THEN wert1=wert1-87
          ELSE wert1=wert1-48
 9 FI    wert1=wert1*16
10 7c    wert2=ASC(RIGHT$(h$,1))
11 wp    IF wert2>64 THEN wert2=wert2-87
          ELSE wert2=wert2-48
12 Pi    wert=wert1+wert2
13 9G    PRINT #1,CHR$(wert);
14 J00 NEXT
15 3n  CLOSE 1
16 0v  END
17 yc  Werte:
18 OS  DATA 10328
19 ph  DATA 00,00,03,f3,00,00,00,00,00,00
20 ms  DATA 00,03,00,00,00,00,00,00,00,02
21 H6  DATA 00,00,00,9b,00,00,09,6c,00,00
22 fg  DATA 0f,53,00,00,03,e9,00,00,00,9b
23 xM  DATA 61,06,4e,f9,00,00,00,00,48,e7
24 Fk  DATA ff,fe,2c,78,00,04,4b,fa,01,9a
25 fa  DATA 41,fa,ff,e6,20,50,d1,c8,d1,c8
26 fZ  DATA 22,50,d3,c9,d3,c9,58,88,58,89
27 im  DATA 48,e7,00,c0,61,00,01,92,4c,df
28 vd  DATA 03,00,2f,09,51,88,22,48,20,11
29 fi  DATA 4e,ae,ff,2e,26,57,50,8b,20,1b
30 51  DATA 2e,00,e7,88,22,3c,00,01,00,00
31 r5  DATA 4e,ae,ff,3a,2b,40,00,0c,50,8b
32 77  DATA 20,07,53,80,22,6d,00,0c,22,1b
33 JC  DATA 58,89,54,81,22,c1,51,c8,ff,f6
34 Ix  DATA 2f,0b,2c,07,53,86,26,6d,00,0c
35 wE  DATA 28,4b,4a,9b,22,3c,00,01,00,00
36 xK  DATA 20,1b,58,8b,08,00,00,1e,67,04
37 3v  DATA 08,c1,00,01,e5,88,4e,ae,ff,3a
38 Bs  DATA 28,c0,58,8c,51,ce,ff,e0,26,5f
39 fz  DATA 42,86,42,85,0c,6b,03,e9,00,02
40 x3  DATA 67,00,00,90,0c,6b,03,ea,00,02
```

```
41 8P  DATA 67,00,00,86,0c,6b,03,eb,00,02
42 po  DATA 67,00,00,9e,0c,6b,03,ec,00,02
43 5x  DATA 67,00,00,a6,0c,6b,03,f2,00,02
44 kO  DATA 4e,71,4a,9b,42,85,52,86,be,86
45 XT  DATA 6e,c6,22,5f,51,89,20,11,4e,ae
46 T5  DATA ff,2e,43,fa,ff,12,20,6d,00,0c
47 CM  DATA 20,10,58,80,e4,88,22,80,52,87
48 Nm  DATA 53,87,67,1a,22,50,20,28,00,04
49 W8  DATA e5,88,22,80,20,28,00,08,58,80
50 Tz  DATA e4,88,23,40,00,04,50,88,60,e2
51 pR  DATA 42,a9,00,04,43,fa,fe,e6,20,6d
52 QR  DATA 00,0c,20,50,50,88,22,88,22,6d
53 MG  DATA 00,0c,20,07,e7,88,4e,ae,ff,2e
54 S7  DATA 4c,df,7f,ff,4e,75,61,28,4a,9b
55 eE  DATA 20,1b,e5,88,20,4b,d7,c0,22,6d
56 3W  DATA 00,0c,22,06,e7,89,22,71,18,00
57 z1  DATA 50,89,4e,ae,fd,90,60,00,ff,48
58 R5  DATA 61,06,50,8b,60,00,ff,40,4a,85
59 AR  DATA 67,02,52,86,7a,01,4e,75,4a,9b
60 Bq  DATA 20,6d,00,0c,20,06,e7,88,22,70
61 8Y  DATA 08,00,50,89,20,1b,67,24,22,1b
62 1n  DATA e7,89,20,6d,00,0c,28,70,18,00
63 Nn  DATA 50,8c,53,80,20,4c,24,49,22,1b
64 av  DATA d5,c1,22,12,d1,c1,24,88,51,c8
65 zw  DATA ff,f0,60,d8,60,00,fe,fa,00,00
66 OL  DATA 25,b0,00,00,3d,4c,00,00,00,00
67 uk  DATA 00,00,00,00,00,00,00,00,d1,d5
68 Pm  DATA 2b,48,00,10,24,6d,00,04,d5,c9
69 QE  DATA 20,20,e2,88,66,02,61,6e,65,32
70 ZK  DATA 72,08,76,01,e2,88,66,02,61,62
71 vg  DATA 65,4c,72,03,42,44,61,64,36,02
72 6n  DATA d6,44,72,07,e2,88,66,02,61,4e
73 tm  DATA e3,92,51,c9,ff,f6,15,02,51,cb
74 rd  DATA ff,ee,60,38,72,08,78,08,60,de
75 Io  DATA 72,02,61,40,0c,02,00,02,6d,12
76 6k  DATA 0c,02,00,03,67,ea,72,08,61,30
77 jR  DATA 36,02,32,3c,00,08,60,0a,32,3c
78 qZ  DATA 00,09,d2,42,54,42,36,02,61,1c
79 HW  DATA 53,4a,14,b2,20,00,51,cb,ff,f8
80 7S  DATA b3,ca,6d,00,ff,90,4e,75,20,20
81 vV  DATA 44,fc,00,10,e2,90,4e,75,53,41
82 5p  DATA 42,42,33,c8,00,df,f1,80,e2,88
83 Vw  DATA 66,08,20,20,44,fc,00,10,e2,90
84 Vz  DATA e3,92,51,c9,ff,ea,4e,75,ff,ff
85 SF  DATA 00,00,03,ea,00,00,09,6c,38,1c
86 VY  DATA f0,08,30,34,00,40,50,31,02,98
87 pF  DATA 49,02,71,00,60,4b,a1,42,70,ac
88 ZH  DATA 12,e7,ff,7f,34,9e,e4,04,00,40
89 3i  DATA 08,86,08,03,78,39,10,f0,02,00
90 Mn  DATA 01,01,c9,d7,fd,70,83,be,10,36
91 a9  DATA 50,38,10,37,d0,38,10,33,b0,0e
92 74  DATA 30,30,30,38,10,31,b0,34,70,31
93 zo  DATA 02,06,56,03,8a,06,d8,17,03,83
94 zP  DATA 06,ea,01,02,06,86,06,26,06,46
95 Wy  DATA 03,8a,07,88,1f,06,e2,06,aa,01
96 Qt  DATA 02,06,3a,06,86,06,5a,07,09,80
97 eV  DATA 06,0c,15,00,54,04,08,18,18,0a
98 tO  DATA 28,1f,61,ec,0a,08,1b,80,04,08
99 hR  DATA 1a,68,0a,18,19,68,0a,08,1b,90
100 dQ DATA 04,08,1b,68,0a,18,18,e8,05,8a
101 Jn DATA 38,e3,01,02,06,ba,24,05,99,86
102 dj DATA 82,9e,81,a2,81,9f,4a,11,40,24
103 Ng DATA 16,84,58,00,c5,08,93,0a,00,41
104 RW DATA 03,61,3e,04,80,a1,03,8b,03,30
105 S4 DATA 80,26,40,a2,00,01,41,03,c4,0b
106 g7 DATA 81,40,a2,82,1e,a1,c2,6c,65,cc
107 Cx DATA 3f,01,21,65,f6,1e,d2,87,29,06
108 qN DATA f9,39,48,38,00,28,39,10,36,70
109 ng DATA 39,08,36,50,18,10,30,d0,39,17
110 Kr DATA 10,3e,08,45,70,2d,83,91,5c,06
111 1g DATA e0,e5,c0,dc,05,60,c1,38,1d,c1
112 6A DATA c9,41,91,01,c8,81,a0,01,c8,a8
113 h9 DATA 00,30,72,20,71,00,72,28,24,e4
114 y9 DATA 0d,05,54,06,04,0c,c9,80,05,0c
115 Nq DATA 15,00,56,20,1e,18,04,28,00,45
116 Os DATA f0,2e,80,10,83,22,01,80,8f,01
117 xv DATA 78,ab,d4,10,0e,04,32,04,01,81
118 7o DATA 60,11,20,c0,30,00,44,88,2c,2b
119 vU DATA 38,0a,27,f5,ec,44,28,0c,43,ac
120 eG DATA 02,04,5c,05,54,0c,18,57,00,b8
121 tw DATA 5f,00,78,31,09,40,19,0b,80,1c
122 dX DATA 08,7d,86,0b,20,91,51,0d,45,e0
123 pZ DATA 34,81,60,c4,32,b0,30,e1,c2,c4
124 ny DATA 0f,0c,18,4c,18,2c,18,6c,18,72
125 vJ  DATA 40,ec,42,60,b7,03,61,0f,c4,81
126 1t  DATA b5,f5,26,00,50,a6,80,44,2b,18
127 2z  DATA 1a,18,1b,01,46,51,20,1d,3a,0b
128 00  DATA 40,84,61,4f,a0,58,42,01,27,90
129 mj  DATA 50,50,40,1a,15,91,c9,d7,fc,8c
130 ty  DATA 8a,04,0d,90,05,05,60,02,80,80
131 ys  DATA 28,41,82,70,39,30,9d,c1,04,d0
132 Za  DATA f7,05,61,79,03,d5,fe,ea,a8,79
133 00  DATA 88,08,9a,04,b0,1f,62,ac,ff,d7
134 Ry  DATA f3,18,11,61,80,7d,87,8c,e2,0f
135 wo  DATA b0,c2,6e,2c,21,fc,15,85,44,01
136 1g  DATA d1,00,b7,01,4a,60,6a,81,83,03
137 97  DATA 41,01,85,80,27,03,06,07,02,03
138 KP  DATA 08,40,a2,06,02,30,21,14,3c,c8
139 d1  DATA 88,8c,40,a8,c0,c8,40,60,20,84
140 8X  DATA 10,1c,a0,00,e0,57,88,00,a3,31
141 9R  DATA 0f,03,10,13,07,41,71,01,04,20
142 BT  DATA 2b,0c,85,e9,5c,c2,70,03,c0,8a
143 sa  DATA 10,45,f3,30,07,f8,2b,93,18,0a
144 bQ  DATA b1,09,3e,81,0b,00,bc,00,db,09
145 Ix  DATA 8a,0d,8c,03,93,fa,30,22,05,0a
146 c7  DATA c7,17,06,0a,c2,4c,03,84,00,07
147 gb  DATA 02,06,09,3b,02,81,45,cc,3f,01
148 67  DATA 01,00,fb,0f,69,34,80,ec,03,d9
149 kx  DATA 1c,10,36,50,48,1c,0e,f0,73,93
150 DS  DATA af,fb,a1,97,df,f7,fc,07,58,81
151 vj  DATA 0f,a0,c1,34,3d,a2,4f,61,82,88
152 yd  DATA 55,34,3c,c0,24,a0,0a,90,37,14
153 TK  DATA e7,fe,7e,e5,5c,25,cf,fe,ff,41
154 yH  DATA c8,05,c8,04,11,00,7c,0b,2e,01
155 Dw  DATA d8,24,30,38,10,24,04,34,10,0d
156 Bx  DATA 40,02,10,6e,a0,63,0c,0e,84,03
157 ga  DATA 02,f4,00,4a,81,7a,05,02,35,2a
158 vo  DATA 0b,a7,03,e6,07,04,e0,06,81,44
159 89  DATA 20,6d,10,00,91,97,df,f7,fd,1d
160 0g  DATA 0a,41,02,30,32,10,0b,52,0e,e6
161 Ut  DATA 08,0e,8c,22,81,0e,bf,c1,e0,d0
162 bC  DATA 2e,44,0c,59,81,e0,81,c0,81,bb
163 Ws  DATA 81,c3,81,b9,4f,c1,c2,c0,1f,82
164 FO  DATA 20,b7,4a,83,88,02,21,e4,58,80
165 qy  DATA 80,82,80,42,e0,63,80,80,92,80
166 cx  DATA 42,c0,00,c4,21,32,80,42,00,02
167 1G  DATA 80,80,8a,e6,1f,a0,c0,00,7d,87
168 vx  DATA b4,80,77,c8,80,4f,00,8c,2c,00
169 OF  DATA c0,0f,ed,22,14,6c,b8,00,3f,41
170 mo  DATA e0,19,a9,41,32,00,41,81,d0,e4
171 IQ  DATA 40,a0,51,0a,8d,40,d5,0f,20,c1
172 Iu  DATA 03,48,37,2a,e1,2e,70,70,33,06
173 IV  DATA 0c,5e,00,a0,82,18,79,91,41,3c
174 qz  DATA 82,41,68,61,1f,91,10,d8,55,50
175 He  DATA 04,10,30,b0,04,10,34,90,04,08
176 VI  DATA 88,02,08,18,d8,02,08,1b,48,02
177 2n  DATA 04,78,12,0c,05,2b,24,81,11,56
178 ou  DATA 40,d9,00,0b,41,22,81,ae,1e,22
179 05  DATA 03,04,11,c0,41,0a,09,3d,01,42
180 Jz  DATA c5,80,b0,7c,01,51,0f,b0,c1,12
181 kX  DATA 22,2a,1e,60,50,4b,41,26,20,46
182 DA  DATA 62,ac,ff,d7,c0,39,05,00,69,07
183 w7  DATA 89,80,04,d1,0c,03,10,ca,18,86
184 KB  DATA 02,82,a6,4e,21,82,70,45,cc,41
185 OP  DATA 85,04,31,0a,70,24,b8,1d,72,10
186 SU  DATA 71,78,02,e2,08,61,32,fb,c0,e0
187 vN  DATA a0,34,d0,56,09,a0,b8,02,f0,68
188 3N  DATA a0,67,02,64,03,82,06,4c,03,82
189 CB  DATA 06,4c,03,87,06,18,03,89,04,0c
190 pq  DATA 02,c9,00,c6,86,09,40,27,05,01
191 gt  DATA 06,0c,a1,10,01,0f,06,52,c3,02
192 6F  DATA 0a,0d,0b,0c,0f,08,80,0a,d0,97
193 S1  DATA 38,78,1e,5a,1c,5c,1f,1d,28,1a
194 kT  DATA 25,03,2c,d8,6a,80,60,20,62,80
195 3W  DATA 26,28,06,5c,18,06,80,2a,2d,04
196 zT  DATA d0,04,02,56,60,68,d0,00,97,27
197 Mi  DATA a2,10,30,08,53,04,18,0a,80,61
198 OP  DATA 42,80,21,38,66,87,14,ce,65,f7
199 Xc  DATA 81,e0,40,c9,08,05,0a,06,0a,7e
200 4n  DATA 8c,02,a2,11,07,82,88,a4,06,20
201 x9  DATA a2,8d,80,28,42,45,f6,1e,c1,5b
202 kj  DATA 01,91,06,c5,01,11,88,81,10,00
203 B7  DATA 43,a3,94,d0,2b,91,a0,1d,2a,0b
204 JR  DATA 91,10,f9,03,42,76,07,05,06,ba
205 y9  DATA 05,88,70,7f,c0,38,32,20,18,bf
206 DR  DATA cd,4c,20,b0,21,7b,00,20,53,00
207 5X  DATA bd,09,a6,41,c5,39,ff,b5,b9,57
208 XT  DATA 00,71,01,60,06,00,3f,6e,86,a1
209 oo  DATA 84,4a,21,01,6b,80,03,f4,1e,01
210 5F  DATA 9a,9c,13,48,07,18,1d,08,14,08
211 6n  DATA 1e,00,14,04,ac,0a,02,0c,cc,0a
212 9F  DATA 04,0d,60,62,02,f1,10,a0,40,c6
213 7O  DATA 04,6c,02,03,48,21,08,39,08,10
214 iw  DATA a0,00,21,04,40,c0,41,81,b8,80
215 Mx  DATA 30,5a,40,30,21,20,32,2e,40,b4
216 1M  DATA 0c,a4,1e,41,34,0b,28,1b,12,c3
217 2h  DATA 2d,03,17,83,72,ae,81,15,03,04
218 1S  DATA 01,10,c4,1d,c1,4a,81,be,16,20
219 5G  DATA cc,31,04,11,14,41,11,44,11,14
220 Fh  DATA 18,02,2b,30,06,56,80,b9,16,81
221 kd  DATA da,07,4e,0d,ac,d2,06,03,10,ca
222 xc  DATA 88,7d,86,01,16,14,80,70,e4,13
223 KI  DATA 2a,1e,61,18,a1,80,20,80,13,20
224 gg  DATA 41,20,16,40,82,20,6c,8c,1a,60
225 fP  DATA 12,e7,ff,7f,00,ee,02,dc,87,02
226 mf  DATA 28,9b,32,0d,0a,05,70,9f,c5,b2
227 Cd  DATA e7,b0,1c,2a,66,5f,7f,df,f0,d4
228 kK  DATA 84,92,18,28,50,c0,24,04,04,28
229 N5  DATA 44,18,00,0c,5f,ef,b0,66,bf,ce
230 ku  DATA 98,10,85,40,ff,80,08,34,00,48
231 Vx  DATA 1f,e0,02,0a,48,19,07,20,42,3f
232 Mw  DATA 82,10,30,40,e2,c0,90,55,82,c0
233 Y8  DATA 4b,14,0c,20,0d,54,0f,20,fe,06
234 UQ  DATA 94,0c,90,49,8a,49,20,cb,40,48
235 Lc  DATA 20,d3,40,48,c0,c0,80,24,c0,c0
236 Lm  DATA 40,24,2b,00,50,20,63,a0,78,10
237 FZ  DATA 6d,a0,78,28,30,28,23,82,28,3d
238 xz  DATA 08,41,f4,14,08,19,e8,14,04,4f
239 TD  DATA 32,7d,04,14,42,ec,14,26,c3,37
240 Pz  DATA 2a,e6,02,19,50,37,e0,11,0a,00
241 RP  DATA 08,85,f8,03,02,b9,90,7e,88,01
242 o0  DATA 1c,9f,80,78,40,d3,00,10,3e,00
243 pi  DATA 08,d0,66,90,01,04,0d,14,19,28
244 qR  DATA 0f,08,19,28,0f,0a,60,04,c8,28
245 KE  DATA 81,f3,02,82,4c,04,3e,46,85,d8
246 wt  DATA d9,00,8b,c3,fe,01,1a,0c,c0,5b
247 U3  DATA 03,33,05,11,20,88,11,41,c8,58
248 jm  DATA 27,22,06,cb,16,2c,00,80,43,87
249 mI  DATA 07,f4,03,86,06,7d,1c,50,20,38
250 Pq  DATA 82,00,c9,7e,9b,c0,31,01,23,8c
251 tn  DATA 18,70,82,a2,00,50,a8,00,58,48
252 1l  DATA 8b,7c,e5,5c,25,ce,0e,20,29,3c
253 jS  DATA 01,d1,8a,18,06,02,82,a6,41,a8
254 FQ  DATA 5c,11,50,00,71,b1,7f,ff,a1,a1
255 KP  DATA 0a,14,67,19,4b,02,12,02,07,12
256 6n  DATA 57,12,2e,e0,29,00,a0,2c,18,00
257 6q  DATA 50,ef,02,dc,00,6a,00,c0,c1,06
258 DO  DATA 97,80,09,07,84,04,d0,00,89,82
259 Xm  DATA 1c,0b,a0,2e,17,30,28,27,a2,09
260 ZW  DATA d4,19,15,c0,4a,c0,ed,c0,cd,00
261 Zh  DATA 60,56,54,43,e4,30,09,d8,a4,e2
262 qM  DATA 19,8a,a8,79,06,82,86,00,82,00
263 5a  DATA 4c,81,04,80,59,02,08,81,b2,30
264 tv  DATA 69,80,88,83,a0,31,0a,70,c8,d8
265 66  DATA 8f,d8,28,50,c0,24,04,04,28,2c
266 Xl  DATA 29,95,fe,d6,65,f6,11,c0,e5,5d
267 kk  DATA c1,01,40,00,80,98,80,b9,81,a8
268 ag  DATA 80,c1,81,a0,80,78,41,f2,20,0c
269 dP  DATA 40,c2,28,60,a0,20,cc,d8,20,a0
270 77  DATA 53,22,85,80,10,81,cc,01,c0,ab
271 f5  DATA c2,61,89,05,0b,02,18,01,51,5a
272 kh  DATA 05,a4,0d,02,0d,40,0d,0c,0c,10
273 I4  DATA 0d,02,c4,06,82,98,00,e0,40,0a
274 bV  DATA 20,90,5f,60,47,0e,60,1a,a8,00
275 Tp  DATA 30,08,48,24,16,0a,17,17,3a,87
276 P2  DATA 01,1b,06,00,ff,16,12,e7,63,15
277 uo  DATA 04,00,e2,9d,a2,4e,21,58,4c,c8
278 Sf  DATA 20,50,c0,30,00,87,88,78,29,60
279 Hq  DATA 08,23,a8,50,08,51,68,04,20,00
280 PQ  DATA 08,0a,23,68,70,82,01,05,85,61
281 Yn  DATA 80,05,0a,02,41,15,27,fe,2e,45
282 hN  DATA 66,07,c2,65,f7,fc,fe,47,c8,e8
283 ZQ  DATA 7c,00,c1,04,70,c1,60,aa,01,70
284 W2  DATA 42,e0,64,1d,30,42,88,04,c2,1a
285 8r  DATA 14,13,c3,94,18,07,00,42,0a,04
286 LA  DATA 0d,0c,07,82,0c,8c,07,94,0c,20
287 kN  DATA 03,57,e9,bc,03,10,12,38,c1,87
288 w2  DATA 08,20,26,68,50,4a,80,2c,2f,63
289 sz  DATA 64,61,f5,85,d8,02,06,41,fc,07
```

Listing 1. Der Data-Lader »Gibber_Gen« erzeugt auf Diskette das lauffähige Programm »Gibber«. Bitte mit dem Checksummer (Seite 159) eingeben.

```
290 Nc  DATA 80,10,00,88,14,28,10,37,d0,28
291 sd  DATA 09,de,64,15,8a,06,04,89,b0,6e
292 rS  DATA 0b,45,07,07,81,b4,85,02,84,82
293 ki  DATA 08,46,02,b8,22,41,80,ae,44,87
294 Z7  DATA c8,60,87,1c,43,10,d4,40,cd,a8
295 xw  DATA 1b,00,10,04,f3,04,10,03,27,0a
296 HK  DATA 28,19,40,02,38,90,40,1a,40,19
297 99  DATA 40,1b,04,00,07,80,44,1a,44,19
298 GF  DATA 44,1a,02,00,04,00,42,0a,42,09
299 Af  DATA 42,0d,47,26,b8,38,20,4d,09,73
300 of  DATA ff,bf,c1,11,80,62,0e,f0,90,ac
301 ox  DATA 29,07,c2,88,5a,29,9a,10,0a,0e
302 u2  DATA 08,18,28,04,0a,70,20,40,28,10
303 mk  DATA 00,1c,0b,90,1b,28,1a,60,1b,05
304 UU  DATA c1,16,02,05,b6,00,84,14,9b,05
305 Yk  DATA a1,76,06,84,f4,06,2a,06,c0,03
306 7O  DATA 01,e1,20,8a,33,ff,03,86,a4,44
307 CB  DATA 0c,65,f7,fd,fe,e5,5c,c6,8d,06
308 MS  DATA 68,2d,12,0c,00,0a,11,a0,83,00
309 BA  DATA 03,41,03,7b,03,46,83,f9,03,40
310 HY  DATA 83,cc,0d,82,04,a1,3e,00,20,aa
311 qe  DATA 3f,61,c2,08,0e,53,ec,60,b4,17
312 9k  DATA cd,10,c0,56,51,49,c4,16,09,9a
313 K9  DATA a1,e2,1a,0a,18,02,08,f2,20,8f
314 hU  DATA 22,08,f2,8c,01,e5,98,02,8b,f0
315 r0  DATA 5a,0a,00,25,06,22,b8,08,88,11
316 4K  DATA 84,9b,32,0d,0a,05,78,a7,d7,b2
317 O5  DATA d0,cc,22,d0,11,43,00,41,00,26
318 9e  DATA 40,82,40,2c,81,04,40,d8,18,34
319 pF  DATA c1,0d,4a,08,a1,82,85,0c,02,40
320 20  DATA 40,42,84,21,80,01,85,fe,8b,03
321 JJ  DATA 88,4b,9f,fd,fc,b4,0a,8c,02,c4
322 Ju  DATA db,04,cd,8a,08,09,ca,75,0e,11
323 mk  DATA 83,79,97,df,f7,fd,13,05,89,00
324 6g  DATA 05,03,17,ff,fa,1a,05,a1,46,71
325 WK  DATA 94,b0,21,20,20,71,25,71,22,ee
326 r0  DATA 02,90,03,09,e1,0f,00,20,11,0a
327 pH  DATA 7c,d8,88,84,28,0b,d0,c0,5a,02
328 et  DATA d1,80,07,2a,e1,43,ce,40,40,02
329 nP  DATA 82,70,6b,c1,ea,60,4c,f6,43,74
330 9a  DATA 43,6c,28,54,2d,30,52,1c,c0,4f
331 VO  DATA 2d,88,80,8b,08,a4,08,81,12,90
332 Ka  DATA 04,04,89,09,33,01,25,08,d9,02
333 qr  DATA 8f,0b,1e,a0,50,fa,83,be,59,07
334 2h  DATA 0b,05,98,b2,88,c0,2d,20,08,4e
335 Zp  DATA 00,38,11,40,50,19,80,3f,db,c8
336 1i  DATA e0,40,d1,80,e1,1b,10,00,50,60
337 1q  DATA 32,10,20,69,45,30,1a,90,c0,20
338 Z5  DATA 24,c0,48,bf,ff,e0,a0,24,59,93
339 fl  DATA e8,6a,15,30,08,0a,70,15,20,10
340 Bm  DATA 69,41,11,72,00,10,07,80,b8,d3
341 Cm  DATA 1d,2d,05,3c,c0,3c,30,08,16,65
342 py  DATA 1a,18,05,08,0c,04,0c,4b,10,8d
343 40  DATA 02,02,54,01,06,06,0c,00,61,14
344 tl  DATA 01,80,b3,31,c0,6a,02,2a,c2,cc
345 Yi  DATA c3,22,40,8d,19,7d,ff,7f,90,b2
346 pe  DATA 7c,03,04,01,10,62,19,38,01,f6
347 lS  DATA 1e,60,01,c4,06,00,02,02,3c,21
348 Yw  DATA 02,0a,40,0e,02,94,98,00,8f,25
349 kf  DATA 1b,04,c1,01,00,00,80,8a,08,40
350 mf  DATA 00,08,33,53,82,20,00,e3,03,38
351 Rd  DATA 28,e1,51,40,38,a0,ce,05,70,81
352 Zd  DATA a2,80,e0,74,c0,60,55,ca,06,00
353 Fw  DATA a7,01,01,45,72,ae,f9,c3,c3,03
354 vS  DATA 00,c0,81,81,4d,80,f0,40,c0,08
355 2V  DATA 20,f0,c0,ca,7e,e7,90,20,53,09
356 Zt  DATA e8,03,cc,18,17,d8,1c,12,86,61
357 P4  DATA fc,06,e1,04,27,00,50,60,0f,f6
358 ki  DATA ea,10,0a,80,0c,07,c0,00,14,01
359 FM  DATA 54,02,02,02,20,03,c0,1f,38,03
360 px  DATA e7,00,7c,e0,36,98,07,fc,24,0a
361 3o  DATA 40,58,00,40,1f,38,03,e6,f0,a0
362 PD  DATA c8,a4,00,20,0b,48,08,05,4c,04
363 sd  DATA 02,ea,02,01,67,04,02,a3,00,ff
364 8G  DATA ff,41,03,03,80,01,00,e0,02,00
365 uu  DATA a8,1b,cd,03,79,a0,51,30,05,08
366 RT  DATA 08,89,0a,8a,09,8b,0b,88,0e,40
367 q9  DATA 2e,02,68,24,13,80,9a,0c,00,08
368 ia  DATA 23,01,b8,b0,00,40,26,12,00,08
369 1O  DATA 06,02,03,f1,01,02,41,60,40,5e
370 zU  DATA 20,c2,80,00,c4,e6,03,01,73,80
371 Oh  DATA 3c,41,00,a0,00,19,a2,02,00,b4
372 U6  DATA 81,80,90,14,5a,7c,01,01,03,08
373 rS  DATA 84,0a,c4,02,02,06,61,65,00,21
374 5H  DATA 81,98,44,05,08,b5,80,40,3f,40
375 hl  DATA 20,20,66,17,70,08,08,18,84,41
376 8J  DATA 18,58,20,20,20,66,17,88,08,08
377 SX  DATA 19,44,61,10,14,52,e5,01,01,03
378 SG  DATA 3c,b6,40,10,c0,cc,22,02,b5,00
379 va  DATA 45,a0,80,6f,40,40,22,05,1b,00
380 vM  DATA 83,03,37,01,84,00,8f,02,02,07
381 lq  DATA bc,03,01,20,29,45,00,02,02,07
382 h8  DATA 9c,01,5e,82,02,07,bc,02,01,16
383 sD  DATA 80,b0,00,b6,80,68,00,16,e6,80
384 yF  DATA fa,10,70,b4,b6,36,73,a1,b3,0a
385 jl  DATA 24,e8,64,e9,e0,00,0a,3c,30,32
386 AS  DATA a1,6a,60,4e,29,64,64,6a,64,e0
387 7Z  DATA 4b,40,4c,6f,66,0f,0d,13,02,23
388 FF  DATA 4f,02,2e,52,50,c1,a5,43,1b,33
389 gH  DATA 02,59,52,ca,36,01,10,b2,02,09
390 YE  DATA 43,3b,12,f2,52,6f,4e,e6,04,72
391 4s  DATA a6,2e,e3,e5,00,28,e4,28,ab,8a
392 7z  DATA 48,b8,2c,a0,e8,7a,46,d9,c0,92
393 9W  DATA d9,c4,22,a9,b3,8d,10,c5,35,4c
394 HY  DATA 4c,0e,1a,bb,3a,d3,08,56,c5,e8
395 WJ  DATA 3e,c5,22,ae,4e,43,20,c4,4b,d2
396 yV  DATA d4,c2,c6,c5,d4,c0,88,54,d0,1c
397 e4  DATA 32,e6,ae,76,42,30,82,03,9a,4e
398 9z  DATA d6,36,42,c1,5f,05,9a,81,b2,48
399 9K  DATA 35,ea,e8,19,52,ce,d4,c8,4c,81
400 GI  DATA 16,a1,48,3c,c0,e4,94,a9,8b,97
401 G7  DATA b1,d1,5b,57,7d,a1,48,78,25,2a
402 VC  DATA 06,08,06,21,3f,16,4e,40,a5,85
403 Wr  DATA 77,01,e2,48,15,ca,4a,d8,11,da
404 fV  DATA 18,9b,13,37,b1,b7,36,30,20,49
405 7A  DATA d8,10,ca,99,ba,98,d8,13,8f,10
406 0k  DATA b3,78,b1,70,26,74,b6,30,b0,07
407 db  DATA 50,a5,80,c1,71,27,37,fd,d3,02
408 k3  DATA 13,53,67,02,71,4b,23,23,50,79
409 cI  DATA cb,39,d2,02,41,8f,28,0f,55,75
410 HW  DATA 32,75,73,b7,35,34,a8,1a,97,94
411 OJ  DATA d6,51,54,96,16,97,20,0b,1b,24
412 2X  DATA 25,44,64,15,68,09,46,4a,06,54
413 0k  DATA 2d,2c,8e,e8,ba,98,79,04,96,75
414 Jw  DATA a8,40,fe,18,18,25,a6,50,54,16
415 uY  DATA 15,15,a2,e0,07,58,20,69,40,4a
416 5t  DATA 8b,a1,93,90,75,43,02,07,43,57
417 rh  DATA 67,53,13,02,71,43,5b,53,10,77
418 qZ  DATA 29,47,2a,06,0a,32,70,70,73,04
419 Fc  DATA 12,34,f4,34,24,a0,04,56,54,c6
420 Xp  DATA c6,d4,d9,d4,c0,99,01,a1,09,3a
421 B6  DATA 29,3a,02,39,53,17,17,00,78,07
422 mD  DATA 46,08,08,0b,74,10,29,2a,8b,2b
423 Cs  DATA 28,13,28,2a,0b,0a,88,12,83,c0
424 PA  DATA 5a,a0,80,10,60,80,80,72,7f,e0
425 NQ  DATA 0a,b4,78,80,8c,2a,80,32,61,43
426 Dg  DATA 81,cf,00,a9,86,5e,0e,0a,06,70
427 Gc  DATA 00,ac,86,03,e1,00,a6,87,d2,08
428 i8  DATA 01,86,00,1f,0e,0a,07,70,06,84
429 Nh  DATA 86,f0,00,e0,89,8c,86,14,51,cc
430 w9  DATA 87,e1,40,e1,c0,61,c0,40,ee,00
431 Rt  DATA 15,50,c3,00,f0,90,c2,ae,0f,4e
432 XW  DATA c1,11,c0,7d,c8,00,f0,0f,01,10
433 Ti  DATA c2,a8,16,f7,81,8f,80,47,86,05
434 bF  DATA 64,30,d0,0c,fc,1c,03,77,01,5e
435 ZZ  DATA 05,78,3c,08,77,c8,e1,e2,70,cd
436 kd  DATA cb,90,28,0f,b1,10,eb,f0,4e,2e
437 7c  DATA a7,a3,48,ee,39,58,90,ce,03,1c
438 KK  DATA 30,7c,5b,07,0e,67,00,3a,c5,83
439 SC  DATA 0a,80,0f,0c,6b,e1,db,c7,31,ef
440 YO  DATA 40,38,51,43,bc,be,9e,cd,6b,f8
441 DX  DATA a0,51,43,70,cc,f1,ed,7e,3b,8e
442 hr  DATA 01,e4,c1,83,0a,93,a7,ce,39,f6
443 RB  DATA 70,cc,38,36,6c,24,3a,7c,1c,37
444 WW  DATA e9,fd,d6,c7,8e,56,24,37,80,d3
445 jJ  DATA 0c,ef,be,c4,30,32,f0,0e,15,e0
446 JN  DATA f8,1e,0f,0f,e3,9b,e8,15,00,83
447 r7  DATA 61,21,c7,e0,80,7b,48,8e,b0,30
448 OO  DATA 2b,21,9c,06,b8,46,e1,9a,00,03
449 tM  DATA bf,44,c5,83,0a,98,27,9f,63,e7
450 JR  DATA d9,c7,30,e7,40,3c,04,43,00,f7
451 Qs  DATA c1,9c,c0,e1,dc,01,38,00,f5,89
452 Ny  DATA 0e,e0,3a,01,51,77,38,f1,c4,79
453 xm  DATA 02,bc,1e,06,3f,64,49,eb,22,e4
454 SW  DATA 1a,18,07,d8,48,73,f0,48,f0,ce
455 2z  DATA 2f,0c,4f,00,c0,1e,b2,21,bc,08
456 GD  DATA 38,06,46,61,8b,b9,cd,80,79,8b
457 5q  DATA 06,14,3f,c0,6f,fc,03,fc,00,2c
458 zA  DATA 68,b6,22,1f,c0,7f,e0,08,f6,00
459 sG  DATA fe,00,18,07,d4,88,78,0f,f0,00
460 US  DATA fe,00,1f,d0,0f,e0,1c,22,c1,85
461 Jy  DATA 5a,32,48,47,00,03,08,03,26,01
462 Kn  DATA 01,08,07,27,fe,08,03,e7,01,9c
463 2E  DATA c0,3e,21,00,30,c0,0c,20,14,c0
464 e1  DATA fa,12,1c,00,10,21,86,86,3c,04
465 GL  DATA 44,86,10,c0,dc,00,21,40,f0,01
466 dQ  DATA 25,40,e2,0c,1a,05,00,07,00,ae
467 zK  DATA 87,01,d0,c0,3e,00,51,c0,40,f0
468 ed  DATA 00,14,50,dc,86,03,21,80,1f,87
469 Ge  DATA 8f,70,03,90,01,b8,00,78,08,86
470 IE  DATA 14,2e,10,04,00,00,f0,14,0a,28
471 Yx  DATA 60,01,e0,c0,27,10,03,cc,58,08
472 Ca  DATA 7f,11,9e,3c,07,3e,03,b8,c0,60
473 EU  DATA 1f,62,21,f8,00,2a,c3,00,54,79
474 sx  DATA 81,0a,c4,87,80,00,a8,0c,fa,0c
475 sq  DATA d6,0f,3c,16,02,1e,1e,0f,8f,cb
476 SW  DATA 8f,00,3e,37,02,8a,1e,3e,3f,3b
477 lx  DATA 8b,e1,02,aa,1b,94,7c,1d,87,87
478 Wq  DATA de,03,49,87,07,01,81,1f,c7,c2
479 2A  DATA f3,e0,07,f8,01,f6,12,1f,c0,0f
480 n2  DATA e4,3f,19,0b,fc,07,2b,12,1a,54
481 Ml  DATA e5,0e,07,fe,61,98,11,60,21,c0
482 cI  DATA e7,90,f8,38,90,2a,fe,06,c2,43
483 15  DATA ce,bf,c1,ff,b8,5f,c8,15,90,f0
484 zv  DATA 00,14,2c,d7,f8,1f,0a,16,3c,1c
485 21  DATA 07,1c,7f,87,0b,c5,02,ae,20,76
486 dJ  DATA 04,3d,0b,a0,07,8c,07,f8,80,7b
487 82  DATA 03,95,89,0d,00,3e,01,42,0f,c2
488 9F  DATA 81,e1,80,a0,87,ad,7f,60,c1,ca
489 6v  DATA 03,fc,86,c2,43,e0,07,c4,0e,fd
490 ki  DATA 0b,69,9a,ac,88,70,03,f0,01,fe
491 8Z  DATA 00,7f,10,7c,98,88,7f,c0,6f,fc
492 my  DATA 03,fc,00,2c,68,b6,22,1f,c0,7f
493 xt  DATA e0,08,f6,00,fe,00,18,07,d8,88
494 As  DATA 78,0f,f0,00,fe,00,1f,d0,0f,e0
495 2c  DATA 1e,00,f9,c0,1f,38,0a,06,58,3f
496 fu  DATA 43,ff,87,b8,40,b0,0c,31,0e,3a
497 qS  DATA 8d,3f,8c,3f,80,3b,a4,6f,e1,07
498 1I  DATA 22,07,f0,07,f0,80,88,36,80,38
499 hk  DATA 00,31,00,37,00,34,03,a0,17,80
500 XX  DATA 3e,00,3f,05,94,0f,f8,7c,e0,c4
501 H2  DATA 60,87,11,86,81,e7,e0,fd,f0,7b
502 zC  DATA 16,08,19,f8,0b,08,18,3f,f0,dd
503 TC  DATA e1,e8,c1,db,3c,a0,01,48,04,16
504 Vf  DATA 47,e0,2c,bf,8f,fe,10,58,c0,df
505 P6  DATA 87,ce,87,f6,87,29,01,0c,0d,5d
506 iJ  DATA 0a,fc,1f,d1,02,c0,02,04,58,c0
507 xC  DATA fe,80,59,1d,0c,1f,21,c2,f0,14
508 y0  DATA 00,3f,86,3e,a2,c4,83,2c,af,07
509 KE  DATA ef,0e,86,87,a8,e1,1c,0d,9f,1e
510 Ff  DATA 4f,1d,d1,32,2c,e0,7f,c1,ef,87
511 Xy  DATA f7,c6,53,07,03,07,0c,03,84,03
512 Q9  DATA fe,01,fc,1d,02,f8,07,e0,03,c2
513 In  DATA c3,03,1d,fc,39,ec,34,00,30,f2
514 Ft  DATA 18,3c,1a,40,e8,14,f1,ff,e1,fe
515 Go  DATA 16,28,18,b1,77,8e,72,0d,20,0c
516 tt  DATA 03,0f,bc,0c,4f,10,38,30,fe,1b
517 a7  DATA c0,5a,81,a4,50,31,08,82,cc,19
518 6F  DATA a0,02,c1,44,a8,53,49,06,a0,52
519 w2  DATA 0a,c0,1a,03,05,68,b0,72,05,20
520 hR  DATA 90,00,60,11,01,a0,c5,a8,19,e8
521 6f  DATA 10,2f,e3,cf,a3,4d,43,8c,03,cc
522 QH  DATA 00,0e,58,3e,00,72,20,7f,00,7f
523 7f  DATA 08,40,20,7c,01,f4,01,c4,01,d4
524 Bl  DATA 40,28,00,64,00,7e,05,a4,1a,c1
525 au  DATA 4b,0e,fc,87,f1,58,2d,03,f0,7f
526 9q  DATA 02,c2,06,e4,40,e6,0f,fc,0f,f0
527 Jh  DATA 06,e8,81,00,18,40,b0,bf,00,2c
528 Bi  DATA 10,69,92,00,8f,ef,0e,aa,0d,1c
529 bJ  DATA 1c,31,00,46,00,20,58,a0,c7,8f
530 HE  DATA a0,91,18,0d,c0,20,82,31,02,c4
531 iH  DATA 83,2c,2a,c4,3c,3d,fc,0f,ec,29
532 r2  DATA 70,66,30,69,01,d3,22,ce,07,fc
533 gT  DATA 1f,fc,7f,7c,7f,3c,7f,b0,7f,00
534 VT  DATA 37,80,38,00,1c,00,0c,00,0f,c0
535 m1  DATA 07,61,d6,34,1f,e1,b9,e0,fd,d0
536 mH  DATA ad,10,21,80,2c,47,00,58,a0,7f
537 iG  DATA 80,ff,c3,ff,c7,bf,c7,9f,c7,de
538 8j  DATA 06,3e,07,f4,1c,82,c0,30,00,30
539 k1  DATA f0,1f,a3,c7,03,1c,29,04,78,3c
540 o0  DATA c0,07,81,8a,06,69,20,d4,0a,41
541 SO  DATA 38,03,c0,60,ad,11,0c,82,c0,01
542 U1  DATA 80,44,01,f3,64,b0,79,02,7e,0c
543 Qx  DATA 48,1f,f0,7f,83,14,16,f0,85,4f
544 Aq  DATA 80,10,e2,be,00,14,10,c5,21,f0
```

```
545 1n  DATA 18,54,10,7f,f1,ff,86,0c,0a,71
546 QS  DATA 00,80,50,7c,af,f0,77,30,34,41
547 pd  DATA 07,81,ff,01,4b,ff,c7,fe,1f,06
548 bQ  DATA 05,07,00,a0,ff,82,92,36,9f,ff
549 q6  DATA 3f,d1,c9,e0,c5,07,fb,c2,02,85
550 1N  DATA 38,00,65,98,22,85,38,bf,3e,21
551 eK  DATA 3c,40,c9,34,1f,00,79,fa,22,af
552 ac  DATA 10,78,03,fc,06,08,a1,44,2f,c8
553 Df  DATA 13,ff,8f,80,3a,0a,78,35,a8,7c
554 qx  DATA 11,17,c8,34,84,1f,f2,16,90,50
555 1W  DATA 8f,80,50,bf,e0,38,20,6c,86,0c
556 Uk  DATA 52,0f,04,41,03,2c,b1,04,50,64
557 IN  DATA a1,41,7f,07,44,70,34,86,0d,a2
558 H1  DATA 80,ff,0f,f8,c4,45,c1,70,7f,80
559 fi  DATA a0,50,4f,c0,28,3f,e3,fe,1f,f8
560 N9  DATA 14,f0,6d,10,a4,3e,01,4b,06,02
561 KO  DATA fe,7c,81,4a,86,7c,0c,49,82,7f
562 NL  DATA 00,80,a3,03,7c,bf,bf,3c,01,f0
563 YT  DATA 29,e0,d2,28,03,f8,0a,40,30,00
564 VJ  DATA 6f,e7,1e,14,14,90,7e,20,94,f0
565 yW  DATA 62,90,18,05,27,c0,50,58,1f,ef
566 lB  DATA e8,14,50,7e,00,14,83,c0,29,50
567 jt  DATA f0,41,d9,62,40,60,14,10,7c,60
568 Xo  DATA 0a,08,7f,80,38,03,ef,df,9f,ec
569 SU  DATA ec,1c,48,04,18,17,f0,24,2f,80
570 mG  DATA 3f,ee,0c,3f,f3,f6,0a,38,3f,0a
571 S6  DATA 00,1c,73,e3,14,17,f8,7f,87,fe
572 m9  DATA 2c,05,e0,12,17,00,0b,c6,04,0f
573 3V  DATA f8,fb,20,0f,9c,07,43,00,f0,74
574 Xm  DATA 42,c5,00,60,d0,01,00,40,c8,01
575 Fn  DATA 00,33,d0,06,fc,04,85,88,47,c0
576 PW  DATA 41,14,50,02,b1,60,46,93,25,28
577 9H  DATA 3c,16,f8,fc,20,27,0a,7c,12,16
578 GJ  DATA 04,7e,7f,0f,09,02,60,a7,fb,fb
579 CQ  DATA e7,f7,d7,df,d9,01,6f,df,c3,02
580 Wf  DATA 3c,9f,c1,6d,4f,cf,8f,8f,97,97
581 ec  DATA e7,97,b7,b7,8f,cf,e0,71,4b,01
582 Gp  DATA 82,81,82,43,80,43,43,81,41,43
583 Xh  DATA 40,c0,22,21,83,e5,4d,15,10,94
584 wd  DATA 8a,1a,02,0a,12,02,02,0c,14,04
585 XQ  DATA 18,20,20,30,30,18,3d,85,fe,81
586 GD  DATA 0c,09,82,ff,3f,bf,3f,ff,24,04
587 kM  DATA 01,80,41,41,8d,ea,43,41,c3,21
588 Xy  DATA a3,a2,21,20,62,61,a1,a3,23,20
589 Eo  DATA 23,40,42,82,03,ff,7f,bc,be,3d
590 EJ  DATA dd,de,23,f6,46,e2,ea,f6,e1,fe
591 9Q  DATA e9,f3,fc,82,b0,01,3d,64,74,1c
592 e3  DATA 02,22,62,02,62,52,42,22,42,02
593 eL  DATA 1c,74,34,24,18,48,7d,6a,01,fc
594 en  DATA fd,19,92,fb,7d,79,7a,7b,b8,79
595 2p  DATA ba,ba,bc,07,d0,19,0a,02,d7,b7
596 jp  DATA 37,77,37,b7,b7,66,05,1e,9f,9f
597 iV  DATA 5f,85,ea,7c,1c,d1,05,ed,5f,81
598 An  DATA 12,80,87,db,db,fb,e7,f7,e9,31
599 Hp  DATA 6f,83,72,a0,90,b0,44,01,42,2c
600 md  DATA 34,34,38,18,10,18,15,3f,83,72
601 zk  DATA bf,03,e5,79,40,10,70,08,48,48
602 mI  DATA 51,02,e0,a0,81,84,00,0a,8b,83
603 aW  DATA 40,23,c2,c2,c2,41,8e,91,ff,7f
604 GI  DATA 7a,20,2c,f6,f9,fb,fc,1a,5c,0a
605 Dx  DATA 1e,0d,1f,08,84,9c,8a,82,80,83
606 D0  DATA 01,12,18,08,1b,f7,e7,e7,fa,1e
607 Ew  DATA 02,5f,df,7f,ff,c0,10,50,10,60
608 7d  DATA 10,e0,20,00,ff,7f,5f,af,6f,1f
609 Xm  DATA 50,e2,52,01,04,80,80,06,ff,78
610 D0  DATA 7e,b8,bf,3a,b9,be,7d,7e,f9,00
611 uM  DATA 46,c6,a6,e7,17,53,d3,35,34,34
612 R1  DATA d0,50,94,e5,a0,20,40,01,78,b9
613 WH  DATA dc,58,58,9c,9f,1a,98,5b,98,5e
614 4h  DATA 58,db,d8,3a,a7,70,a9,ce,da,cc
615 RO  DATA c8,f7,77,47,57,5f,78,f2,e1,e9
616 UJ  DATA d5,f3,f7,d7,ff,f8,1c,2a,22,02
617 5f  DATA 02,0c,08,2f,e7,e3,e5,f1,c6,ca
618 BI  DATA c2,d4,c4,c4,d8,d4,d2,d6,f1,fd
619 HO  DATA ff,d2,29,1b,38,ae,93,95,3b,e1
620 uG  DATA 83,2e,06,0a,3c,14,48,00,17,4b
621 2L  DATA 3d,a5,49,c0,f0,1e,0c,06,05,27
622 8N  DATA 15,37,f5,2f,ee,3d,ff,42,e3,d3
623 0d  DATA 73,0a,2b,6b,99,da,b9,f8,c3,c0
624 if  DATA 38,3f,e7,e2,e5,60,64,a6,25,cd
625 ad  DATA 40,40,80,02,f1,75,b0,b8,31,0e
626 Dy  DATA 14,20,a3,26,24,38,f2,70,74,3a
627 JL  DATA 25,2b,38,b6,bd,b8,6e,73,78,ee
628 P2  DATA e7,fc,15,18,95,9f,99,58,d5,c6
629 5U  DATA 1e,02,c1,ff,1f,77,17,47,5b,2b
630 uO  DATA 4b,13,03,1d,25,29,7e,16,0a,0c
631 U7  DATA 48,1f,87,a3,e5,f1,ae,86,92,cc
632 h8  DATA 84,a0,94,78,08,86,03,a3,d2,cb
633 hB  DATA 9b,fb,a7,d7,8f,6f,df,fe,21,11
634 GX  DATA d0,09,c8,19,39,f9,c5,15,d5,75
635 sd  DATA 8d,2c,1d,5d,dc,7c,fc,d1,e0,0b
636 7e  DATA fc,fc,fd,bd,3d,5d,5c,9c,6d,cd
637 ZS  DATA 8c,0d,35,95,65,c4,05,b8,98,29
638 tE  DATA 09,50,21,7f,5e,ae,0e,d6,17,87
639 Fb  DATA 7a,db,1a,ab,cb,8b,0a,71,f0,23
640 Rd  DATA 90,40,40,b0,a4,a2,b2,b6,b1,a3
641 oT  DATA a3,ab,bf,bc,7c,62,7e,75,77,70
642 lm  DATA f8,fa,f1,f5,f5,f3,ef,e1,1f,ea
643 BJ  DATA e0,f0,08,08,c8,58,d8,f8,84,a4
644 I6  DATA e4,54,b4,8c,4c,cc,cc,8c,cc,50
645 QS  DATA e8,13,60,e4,e7,61,62,63,a0,66
646 BX  DATA a5,a4,a0,a5,21,07,da,81,0e,36
647 IC  DATA 0e,06,1a,28,1d,08,60,7f,bf,af
648 jJ  DATA 93,30,43,c0,80,2a,fa,f6,e6,e6
649 sX  DATA ee,e1,e9,e9,e1,e1,ee,fa,e6,fa
650 Cw  DATA f2,e8,f0,fb,63,7a,1f,02,5b,9b
651 Pc  DATA db,87,a7,e7,e7,8f,2f,2f,5f,40
652 mC  DATA f0,03,ff,00,80,00,00,7f,45,a1
653 vc  DATA cf,8f,b7,77,17,17,27,a7,87,47
654 tW  DATA 67,40,e6,d1,be,7c,01,05,43,23
655 GB  DATA a2,62,12,10,93,13,92,51,50,d0
656 nP  DATA 53,90,92,10,13,63,a1,20,21,43
657 mL  DATA 81,c3,f7,5b,e3,f3,e3,f8,36,bf
658 Ud  DATA ff,f0,14,02,2c,0a,06,26,2e,21
659 y4  DATA 11,11,31,29,39,29,35,0d,3d,39
660 mM  DATA 15,35,1c,38,74,f0,88,09,31,51
661 88  DATA 50,91,a0,c1,c1,80,7e,3e,41,ff
662 ZJ  DATA 2a,f3,7b,74,02,44,e0,a0,b1,d1
663 Z1  DATA e1,a0,3c,16,00,9d,9d,dd,a3,02
664 73  DATA 87,07,07,c7,87,87,c7,27,e7,97
665 gY  DATA 17,97,c0,f0,16,bf,be,7b,7f,7c
666 E7  DATA fa,f9,ff,25,a9,f3,ff,f4,60,57
667 4H  DATA e7,f7,e8,40,9f,cf,cf,d0,35,40
668 q2  DATA 5d,fb,fb,f7,f7,f0,18,0c,12,a8
669 ny  DATA 00,81,08,68,04,44,62,02,48,a8
670 L2  DATA 98,42,05,30,71,09,89,83,8a,13
671 bJ  DATA 90,52,12,11,92,51,13,84,12,24
672 G5  DATA 24,20,07,62,63,a1,63,60,88,16
673 mu  DATA c0,0c,24,40,30,5b,d6,24,24,23
674 Dd  DATA ab,80,1b,57,0f,00,93,0d,15,09
675 a4  DATA 11,01,0e,1a,02,0c,08,1f,e0,0f
676 lr  DATA f7,eb,eb,fb,f3,ed,eb,fd,f5,ed
677 bT  DATA e5,f9,e9,f5,1e,00,b5,e5,f1,f6
678 1O  DATA f6,ea,f4,04,b1,c1,c1,ea,ca,e2
679 Ty  DATA ec,d9,61,11,83,82,fd,7d,7d,03
680 5S  DATA c3,43,43,13,d3,13,93,33,2b,ab
681 lO  DATA eb,db,47,a7,27,c7,17,d7,57,57
682 dE  DATA 0f,ef,6f,af,ef,28,f1,57,f8,2e
683 NI  DATA a3,aa,00,82,13,d3,84,85,86,40
684 lZ  DATA c5,40,c6,c2,c3,44,c1,c6,07,80
685 QY  DATA 81,21,29,11,11,21,29,01,12,52
686 aE  DATA 12,72,4a,2a,72,33,02,d4,d4,34
687 ed  DATA 34,c5,04,19,18,18,e9,69,a9,29
688 f5  DATA 49,c8,48,88,08,09,70,b1,b1,c1
689 qB  DATA d0,2d,89,81,8e,81,81,89,8e,8e
690 KN  DATA 86,88,0f,c1,3f,80,c2,80,8c,29
691 SO  DATA 29,8a,c0,1d,f5,fd,fb,f7,f2,06
692 FM  DATA 44,1c,10,d8,ac,22,1c,11,51,50
693 2t  DATA 4a,12,10,02,94,00,f9,60,09,40
694 41  DATA 37,2e,c8,20,c2,c0,c0,c2,42,8d
695 sj  DATA 91,f9,7e,bb,38,38,3a,3a,3d,3c
696 G6  DATA bf,bd,7f,fe,02,85,80,47,80,40
697 Kv  DATA 43,81,85,07,fd,fd,f9,f9,fe,ff
698 S1  DATA ff,fb,47,80,a0,08,30,30,10,30
699 UV  DATA 28,08,10,28,38,34,02,22,0a,32
700 pH  DATA 32,22,2c,14,10,3f,c7,db,e7,df
701 e8  DATA e2,3c,04,bf,de,5e,af,8e,d6,67
702 kS  DATA 47,c6,26,17,b7,af,df,41,f1,25
703 5M  DATA 4d,ce,24,a3,ac,64,60,62,a1,2d
704 UA  DATA c4,0e,28,24,e1,60,61,a0,a5,36
705 ri  DATA 2a,3c,22,26,29,3b,22,b9,a8,71
706 3h  DATA 78,ed,fc,1d,1a,9f,9a,5b,5c,d9
707 cd  DATA d3,d7,cf,cb,d5,ce,d0,d5,44,43
708 ew  DATA 82,89,00,06,e0,ea,67,b5,ba,bc
709 nG  DATA b4,b8,ac,ba,b9,af,b2,75,78,ee
710 sz  DATA fd,f8,1a,15,17,08,84,9d,9d,ba
711 wh  DATA 98,28,08,60,ff,6f,57,47,ab,0b
712 3e  DATA e3,03,c0,f0,16,eb,ee,1e,9e,5e
713 fZ  DATA dc,bf,7b,05,45,25,67,e2,54,35
714 ZA  DATA b4,f0,87,80,60,7f,db,c1,d4,cb
715 bJ  DATA 41,40,51,84,8d,04,03,e2,e7,69
716 gm  DATA 62,7b,bd,b9,be,ba,a2,bc,a2,b2
717 H2  DATA a2,aa,a1,a1,ad,bd,b3,b7,bf,b8
718 mY  DATA 7c,6a,71,65,7b,60,e4,e6,f5,eb
719 vg  DATA e0,1c,1e,15,13,00,88,8c,94,8a
720 LD  DATA 9a,91,89,95,93,93,8b,95,9e,02
721 pU  DATA dc,7c,dc,dc,9c,1c,ec,ec,4c,74
722 mf  DATA 34,e4,a4,44,44,04,58,98,18,28
723 8t  DATA c8,88,08,b0,50,50,e0,20,00,7f
724 fD  DATA 1f,af,8f,37,17,67,47,db,1b,2b
725 58  DATA 4b,0b,e3,03,dd,2d,35,a5,45,59
726 Ya  DATA 99,69,89,09,b1,d1,b1,89,c9,e9
727 RP  DATA d9,05,a5,b5,cd,bd,e3,8b,db,e7
728 QD  DATA cf,ff,d0,e8,c4,74,9c,fc,d2,ca
729 lF  DATA 3a,66,b6,8e,ce,6e,de,7e,f2,f0
730 wD  DATA 05,fe,3e,ee,ce,36,66,06,1a,0a
731 tG  DATA e2,7c,2c,94,04,88,50,7f,6f,57
732 Rv  DATA 07,ab,93,43,3d,6d,0d,95,c5,05
733 Vk  DATA 59,e9,69,29,c9,a9,69,19,99,05
734 bY  DATA 05,c5,e5,d5,0d,ed,dd,23,53,cb
735 AI  DATA 9b,07,97,4f,ef,ff,60,88,e8,d8
736 P3  DATA c4,d4,8c,ec,dc,fc,82,e2,92,d2
737 MJ  DATA f2,4a,8a,ca,aa,1a,5a,9a,6a,2a
738 r0  DATA 2a,0a,52,12,22,fc,7c,9c,2c,d4
739 YD  DATA 64,04,98,a8,b0,90,6a,f8,51,fb
740 Ww  DATA 7e,7b,bb,3a,38,3e,dc,df,5e,59
741 rn  DATA 59,5a,5e,5b,07,40,12,ba,be,b1
742 vH  DATA bd,bd,bb,b3,b4,7c,7a,7e,73,78
743 F9  DATA f4,fa,f5,fb,ff,fc,0e,0e,0b,03
744 4q  DATA 0c,80,8f,81,74,02,ea,a0,9f,05
745 NU  DATA 88,68,a0,80,89,0d,09,05,06,02
746 UX  DATA 04,0c,0b,fd,f3,f5,fa,0e,cd,0f
747 EH  DATA f7,cf,d7,d7,e7,97,87,f2,02,b7
748 L9  DATA 01,84,6e,6f,ae,6f,6e,47,c9,5c
749 aT  DATA 5e,5f,5e,de,3c,be,bf,bf,7f,fd
750 hA  DATA 03,03,d9,52,00,96,15,0a,a0,15
751 BF  DATA 17,40,c2,c0,c5,45,41,42,42,44
752 g4  DATA 41,41,46,07,f6,ba,a2,86,8e,8e
753 bo  DATA 8c,23,49,19,01,16,02,12,14,0c
754 HC  DATA 84,95,20,ca,3f,dc,dd,08,04,0e
755 0v  DATA 5d,5d,5f,9e,9e,9d,dc,3f,dc,3c
756 cj  DATA bd,bf,bd,3e,83,c3,08,fd,fa,02
757 A9  DATA 09,ac,06,06,08,0c,31,50,00,0f
758 Du  DATA e8,50,a9,86,02,ef,6f,9f,df,5f
759 OB  DATA df,9f,bf,80,20,80,ee,53,05,84
760 2R  DATA 39,2b,08,88,34,49,16,c2,c4,c0
761 2u  DATA 24,22,22,25,23,26,24,07,80,10
762 pA  DATA 49,86,8a,86,8e,88,88,8e,40,41
763 rx  DATA 06,10,36,b4,0c,14,14,24,08,1d
764 nE  DATA 10,60,20,7f,bf,bf,d1,c2,c0,a8
765 Qr  DATA 23,f1,fe,fd,f3,fe,f5,f5,fb,ff
766 KU  DATA fb,fd,fc,0f,35,01,48,00,42,bf
767 0q  DATA ff,e0,1f,f1,01,1f,df,df,af,e2
768 rh  DATA 02,9f,6f,2f,42,24,5f,6e,ee,9e
769 6o  DATA 9e,4e,08,3d,3e,16,04,0e,dd,5e
770 WB  DATA 1c,02,2f,2e,6e,1f,2f,af,48,09
771 Mi  DATA dc,3e,3f,3e,be,bc,be,7f,bc,7e
772 T9  DATA 03,a0,5b,ef,f0,ea,29,55,01,60
773 x9  DATA 70,08,23,a2,b0,b0,30,30,d0,20
774 qU  DATA e0,f0,50,60,20,72,40,04,e5,e9
775 KH  DATA 50,48,28,28,18,04,64,54,74,42
776 On  DATA 62,22,12,52,0a,4a,72,4a,0a,32
777 8I  DATA 52,62,12,12,02,6c,5c,6c,02,34
778 M9  DATA 24,64,64,04,48,58,04,28,48,48
779 G4  DATA 18,70,42,78,e1,e6,98,08,30,08
780 rN  DATA 20,08,28,28,38,c1,28,40,07,f0
781 qo  DATA 80,bf,ef,d7,ff,c0,09,c5,20,01
782 yV  DATA 0a,7f,03,64,bf,de,5f,9f,9e,1e
783 qG  DATA ef,1e,1f,ee,1f,6e,ee,ae,ae,2f
784 rf  DATA cf,4f,4e,8f,a1,e0,28,70,60,15
785 cw  DATA 1c,1c,1e,1f,1c,9c,9e,9d,9f,9f
786 of  DATA 9d,03,c0,52,ea,fa,fa,fe,f1,e9
787 nd  DATA e9,f5,fd,f3,fd,fb,e7,fb,ec,1f
788 kU  DATA da,56,9f,ff,d0,20,3f,ff,e0,00
789 BV  DATA 38,81,40,60,50,70,30,12,00,a4
790 bs  DATA 14,80,80,87,00,84,82,83,81,85
791 0O  DATA 80,42,44,41,07,84,99,29,85,83
792 Rn  DATA 80,4b,82,70,17,02,44,84,04,84
793 Y3  DATA c4,44,44,64,04,08,b1,b0,b1,85
794 Vi  DATA 0a,20,72,32,12,11,60,35,1d,13
795 ms  DATA 15,05,15,19,05,05,0d,15,0d,18
796 ox  DATA 1c,82,98,18,e8,18,68,88,88,08
797 of  DATA 30,00,e1,d3,03,06,02,01,a0,07
798 If  DATA f3,f4,65,48,16,fc,fc,fd,7e,20
```

Listing 1. (Fortsetzung)

```
799 Db  DATA 20,fd,7b,75,7c,3a,42,e0,e4,1b
800 9m  DATA 09,9c,5e,0a,90,b9,be,bb,b9,bc
801 0e  DATA bd,2e,2e,79,78,6c,0c,00,84,44
802 Mp  DATA 12,38,14,24,09,1c,9e,10,16,3b
803 yG  DATA 3d,3d,3e,28,21,7e,76,76,7e,75
804 UM  DATA 75,7d,73,73,78,f8,f0,f2,0f,01
805 xo  DATA 17,82,ba,1f,9f,7f,5f,3f,bf,a7
806 x1  DATA d5,bf,7f,02,01,00,04,00,a0,5f
807 rl  DATA f0,18,18,14,14,18,02,02,12,02
808 Ct  DATA 01,16,01,11,11,19,05,1d,13,07
809 3y  DATA 17,00,80,98,81,1f,a2,c4,05,84
810 OQ  DATA a1,4a,0a,43,02,94,54,84,04,28
811 YV  DATA 0d,09,91,92,0a,16,05,48,c8,49
812 yA  DATA c8,c9,c9,48,0a,93,12,12,10,12
813 ji  DATA e0,62,60,63,a3,a0,a1,21,c0,20
814 zZ  DATA 21,43,43,41,40,80,c3,e7,40,00
815 CC  DATA 10,80,20,6f,f7,e8,00,07,fb,fb
816 cq  DATA ec,a8,aa,85,bc,1d,13,78,7c,7d
817 t1  DATA 7e,0a,26,f2,fe,f1,f9,f2,51,4c
818 gU  DATA 54,04,3e,5e,5e,9f,88,01,cf,04
819 zf  DATA 04,9e,1e,ee,ef,1f,ee,1e,af,cf
820 lX  DATA 4f,01,d0,22,7c,74,72,7c,30,40
821 Js  DATA f7,6f,68,40,81,c1,00,30,e0,80
822 Cw  DATA b1,d9,f9,f5,d5,f5,dd,d3,e3,f3
823 3w  DATA f3,fb,f3,e3,db,db,eb,c4,3c,05
824 xR  DATA de,de,3e,8c,0b,bf,be,7d,7c,fd
825 mj  DATA fe,96,02,80,81,0b,ab,42,9e,a1
826 6o  DATA 85,fc,02,11,d2,04,06,04,03,01
827 v6  DATA 00,83,00,87,02,83,80,40,46,45
828 Ad  DATA 47,46,c5,42,87,64,a6,16,36,0e
829 FO  DATA 0e,09,c1,40,14,13,62,e1,10,16
830 HJ  DATA c3,c0,26,26,21,21,24,26,20,07
831 0a  DATA 40,91,01,01,02,5c,1c,5c,2c,4c
832 qI  DATA 0c,0c,74,00,72,0a,23,20,a1,0c
833 82  DATA 16,44,42,46,46,41,20,24,80,8b
834 VD  DATA 03,0d,0d,0a,60,44,00,80,b0,00
835 dJ  DATA 0d,7b,50,17,f8,38,25,02,df,80
836 m8  DATA ba,80,bb,d7,e7,e7,f1,01,17,57
837 tt  DATA 53,d5,be,7f,02,6a,7c,7f,bd,bf
838 oH  DATA be,be,88,02,5e,de,9e,1e,ee,49
839 PK  DATA 4a,dd,5f,5d,08,02,6f,af,6e,6e
840 xb  DATA 1f,9f,1e,9f,1e,1f,01,c4,26,71
841 BU  DATA 7e,76,7c,70,52,f2,e2,e2,f2,fc
842 PK  DATA e2,f4,f4,ec,e4,f4,fc,f2,f6,e1
843 H9  DATA e1,f1,e5,f5,f3,e7,e0,00,1e,1e
844 Oo  DATA d2,c0,60,20,60,a0,24,01,58,01
845 3S  DATA 01,70,03,02,04,b0,20,20,10,40
846 lr  DATA 2f,ff,f7,ef,f8,20,8f,d7,f7,d1
847 C7  DATA dd,05,0a,05,7f,bf,07,29,02,01
848 BS  DATA 01,07,96,e4,64,5c,8a,89,8d,8b
849 7V  DATA 82,20,50,80,88,88,c0,9e,01,01
850 kO  DATA 02,1c,5c,02,5c,1c,52,02,20,15
851 dU  DATA 8d,85,8e,8a,82,84,86,66,09,80
852 Cm  DATA 09,2d,0d,09,0a,01,05,02,09,0e
853 vd  DATA 0e,06,01,06,0c,0a,0e,0e,0e,c9
854 EN  DATA 20,5f,df,a2,58,57,ef,e1,14,20
855 lo  DATA 10,14,14,12,1a,16,0c,e0,90,22
856 k5  DATA 86,8c,84,77,14,ba,d0,50,90,90
857 dy  DATA 10,c0,c0,80,ff,7f,80,ec,50,fe
858 jM  DATA 7c,20,20,f9,76,7e,76,7a,76,79
859 15  DATA 71,79,75,7b,7d,7b,7b,76,0f,01
860 3L  DATA 47,e7,97,97,e0,82,08,ab,d3,e1
861 fy  DATA 81,37,d7,e7,a7,e2,d2,ef,af,ef
862 Mi  DATA 45,c4,e1,40,5f,f7,ef,e8,6e,9f
863 a3  DATA c1,a1,0f,d7,c0,e2,28,87,c0,c1
864 Ng  DATA 68,5b,f7,f3,e8,90,9b,c1,e1,4f
865 Kw  DATA 9f,ef,af,cf,f7,cf,b7,f7,af,8f
866 EE  DATA b7,cf,8f,ef,9f,ff,c0,48,76,ca
867 t4  DATA 80,15,52,c4,a1,0c,50,16,84,0d
868 aR  DATA 58,44,38,38,58,18,04,58,38,68
869 s8  DATA 58,20,74,88,40,18,43,00,21,00
870 wf  DATA 80,12,2a,c0,21,41,40,40,88,03
871 s5  DATA 40,81,04,01,1e,84,18,24,11,71
872 wJ  DATA 04,04,24,22,52,e8,28,48,d0,e0
873 I7  DATA e0,60,65,05,c9,20,40,42,08,40
874 Du  DATA 20,20,75,50,08,40,82,90,01,45
875 jm  DATA a0,61,0f,0b,40,08,10,44,44,45
876 QT  DATA 45,46,41,46,42,40,41,86,82,81
877 8N  DATA 05,07,52,b7,eb,d3,c3,dd,f5,c5
878 pk  DATA e9,d9,c9,d9,c5,e5,f5,dd,d3,d3
879 na  DATA db,eb,c7,f4,3a,e4,61,78,4e,16
880 Mw  DATA 05,48,22,12,45,bf,bf,05,eb,12
881 X4  DATA 10,88,22,f2,fa,fa,fe,fe,f3,f6
882 gi  DATA 22,40,00,26,ae,85,00,03,42,00
883 jt  DATA 0e,88,60,81,df,bf,df,ef,af,af
884 Mn  DATA cf,af,9f,df,df,9f,af,ef,d0,78
885 1G  DATA 09,02,7f,04,04,05,7f,ff,e0,50
886 AS  DATA 10,70,28,48,71,12,b0,a8,08,1c
887 s4  DATA 00,84,58,58,68,18,68,68,32,02
888 YN  DATA f0,a4,00,51,30,00,a2,80,a0,a0
889 pO  DATA 45,05,20,0e,e1,26,b8,08,04,34
890 1q  DATA 04,28,30,30,10,10,28,10,3d,65
891 o5  DATA 00,0a,88,ff,7c,7f,be,be,3f,3d
892 gd  DATA 8c,11,bc,b8,ba,3d,23,70,36,5d
893 Kp  DATA fb,ff,e0,18,26,84,14,21,00,53
894 t2  DATA 05,0d,04,36,59,15,10,1e,b6,16
895 Ui  DATA 26,06,06,09,01,4c,01,7a,90,c0
896 U1  DATA 00,43,94,bf,9f,1e,6f,82,01,0f
897 p1  DATA 04,05,ae,6f,6f,1e,1e,1f,1e,5f
898 MP  DATA 5f,5f,de,81,f8,a3,fb,f3,f0,08
899 1C  DATA 02,06,0e,09,0b,08,0e,55,44,03
900 JF  DATA 7a,28,a8,87,b4,90,11,60,a1,c0
901 nJ  DATA 49,28,00,12,13,fb,fa,fe,fe,10
902 K7  DATA 2d,f7,ff,f4,08,58,1f,e4,08,f8
903 IH  DATA 88,00,81,20,03,6a,80,bf,c0,e9
904 m1  DATA 6f,80,d2,5f,9f,9f,44,00,f7,8f
905 nJ  DATA c2,1a,5f,3f,40,c0,c0,a0,e0,d0
906 TN  DATA 70,08,c8,e8,18,e8,a0,eb,54,04
907 W6  DATA 08,f1,30,d1,d0,51,10,cd,a9,41
908 k4  DATA 81,01,fc,fc,fe,82,93,78,11,a8
909 uq  DATA fb,7b,71,71,7a,76,7e,79,7e,7d
910 cA  DATA 7b,74,f2,0e,0d,4f,80,30,68,04
911 DX  DATA 44,54,34,0c,0c,4c,2c,4c,54,24
912 HG  DATA 44,28,48,48,7f,69,7f,bc,be,3e
913 2b  DATA 5d,9e,9d,90,02,81,43,83,fd,dd
914 rB  DATA f3,f3,9b,db,c7,f7,cf,a0,7e,ca
915 IV  DATA 80,43,43,21,20,a2,88,54,05,88
916 XG  DATA 0d,c8,e3,62,63,63,17,91,e4,61
917 ic  DATA 20,21,c1,80,80,81,00,00,ff,be
918 yy  DATA 3e,5f,5f,5e,df,9f,9e,1f,6e,6e
919 0x  DATA af,2f,6e,ef,6e,1e,1e,9f,5f,51
920 YJ  DATA e5,2d,ff,f8,04,0a,0e,01,03,00
921 di  DATA 8f,00,8c,8a,8e,89,89,87,80,40
922 3K  DATA 4d,80,45,8e,8e,86,86,80,8d,06
923 WH  DATA 02,08,08,0f,f1,8f,cc,3b,e0,80
924 2X  DATA bb,c0,84,40,de,a9,aa,03,91,80
925 hJ  DATA 7f,f0,22,02,08,10,00,df,6f,22
926 Up  DATA 04,de,3f,03,c8,82,28,a0,05,62
927 HO  DATA c0,8d,26,4a,50,78,2d,f1,f8,40
928 ZZ  DATA 4e,14,f8,24,2c,08,fc,fe,1e,12
929 9z  DATA 04,c1,4f,f7,f7,cf,ef,af,bf,b2
930 GM  DATA 02,df,bf,86,05,fe,0c,aa,fe,fe
931 L1  DATA 7e,7c,7c,7c,bc,bf,3c,bd,bc,01
932 Vt  DATA c4,d2,04,00,24,32,0e,03,43,c3
933 Ip  DATA 44,80,09,20,03,80,05,20,01,60
934 Py  DATA 1c,b2,90,01,04,00,0d,0a,00,17
935 fn  DATA 51,04,08,02,81,44,01,00,83,80
936 ip  DATA 80,51,c0,40,24,e0,80,80,19,e0
937 1E  DATA 54,28,aa,17,b0,04,bc,0b,d0,0c
938 GC  DATA a4,1c,48,98,1c,40,81,88,00,62
939 jC  DATA 20,40,81,f0,00,12,20,40,81,b0
940 1b  DATA 00,52,20,b9,81,b2,00,99,81,f2
941 Bq  DATA 00,a9,81,8a,00,89,81,ca,00,b1
942 OL  DATA 81,aa,00,d0,81,80,00,92,41,04
943 30  DATA 90,90,d8,5d,a0,5a,e0,b5,80,11
944 5M  DATA a0,d1,80,51,c0,96,80,31,80,e2
945 IM  DATA 80,71,f0,00,ff,fb,d0,e5,d4,0e
946 35  DATA 00,02,04,0d,58,44,08,19,35,80
947 95  DATA 01,02,07,80,00,40,82,e6,07,40
948 dt  DATA 02,66,06,c0,02,a6,07,c0,02,26
949 At  DATA 06,20,03,c2,06,20,00,a0,83,42
950 Uu  DATA a0,a8,5e,c0,10,f0,2f,40,30,90
951 0f  DATA 71,00,24,80,1e,00,14,80,15,a8
952 WB  DATA 71,89,14,18,10,38,08,05,1c,04
953 SI  DATA 02,4e,0a,a1,67,00,94,70,01,02
954 QJ  DATA ad,40,17,10,10,20,7c,00,0f,08
955 ys  DATA 10,20,6c,00,1f,08,10,20,74,00
956 0A  DATA 00,88,10,20,64,00,10,88,10,20
957 Vh  DATA 78,00,08,88,10,20,68,00,18,88
958 DJ  DATA 10,20,6d,11,24,04,04,28,00,3a
959 gH  DATA 0c,05,fe,fe,18,18,f8,41,23,06
960 W2  DATA 30,50,82,88,17,30,64,f0,7a,80
961 0h  DATA 09,80,0f,85,85,64,90,6f,17,66
962 zh  DATA c3,28,e0,62,05,b0,06,70,07,c1
963 Hd  DATA e7,03,1c,e4,79,1c,0c,40,a8,00
964 QX  DATA 38,00,38,40,a0,01,01,88,c0,30
965 wC  DATA 20,1c,30,16,10,80,a0,60,07,10
966 Hf  DATA 20,14,40,10,08,a0,08,04,30,04
967 C2  DATA 02,d8,02,01,22,01,00,8b,08,02
968 Wq  DATA d4,08,04,18,04,02,2c,20,0d,d0
969 aK  DATA 20,13,e0,10,09,08,08,05,24,04
970 io  DATA 02,2c,20,00,b0,20,12,61,00,75
971 m3  DATA 81,00,96,80,80,49,84,01,0e,04
972 S7  DATA 02,26,02,01,73,01,00,b5,80,80
973 TB  DATA 4e,84,00,00,c2,00,7f,02,01,08
974 OL  DATA 01,00,b4,00,80,49,00,40,2a,80
975 9L  DATA 20,12,e1,80,01,70,c0,0d,40,80
976 dQ  DATA 5e,80,40,22,c0,20,13,60,10,0a
977 B6  DATA 70,08,04,78,04,02,52,02,01,59
978 Is  DATA 01,00,9b,0c,00,0d,86,01,b6,04
979 tN  DATA 02,ba,02,01,03,01,00,b1,80,80
980 zD  DATA 45,86,01,9e,04,02,6e,02,01,6f
981 0z  DATA 01,05,00,bf,01,01,80,88,26,00
982 nC  DATA 10,0a,40,08,05,90,04,02,18,02
983 P8  DATA 01,06,14,05,c8,10,08,10,08,05
984 AL  DATA 88,04,02,64,02,01,6a,01,00,85
985 s0  DATA 0a,00,74,08,05,78,04,02,02,02
986 lp  DATA 01,21,01,00,b2,80,80,4d,40,40
987 0e  DATA 2f,a0,20,15,30,10,0b,58,08,05
988 17  DATA 1c,04,02,ee,02,01,1f,01,04,80
989 ix  DATA 01,21,c0,30,81,00,b4,00,80,48
990 N6  DATA 87,01,52,04,02,58,02,01,12,01
991 1h  DATA 00,9d,00,80,53,80,40,23,c0,20
992 gM  DATA 17,90,10,08,a8,08,04,34,04,02
993 K9  DATA ba,02,01,43,01,00,a5,80,80,48
994 Vy  DATA 87,01,9e,04,16,01,fc,08,04,40
995 JJ  DATA 04,02,70,02,01,44,01,00,86,00
996 QK  DATA 80,5f,00,40,24,20,20,13,10,10
997 kH  DATA 0a,48,08,05,d4,04,02,3a,02,01
998 n8  DATA 7d,01,00,91,80,80,54,c0,40,25
999 JN  DATA 60,20,15,b0,10,09,38,08,05,5c
1000 KZ  DATA 04,11,01,7c,08,05,80,04,02,90
1001 nO  DATA 02,01,18,1a,02,c8,10,0a,e0,08
1002 5i  DATA 04,20,68,00,74,2c,02,90,20,17
1003 RP  DATA 40,10,09,90,b0,19,40,80,56,80
1004 wa  DATA 40,28,c0,20,10,e0,10,08,a8,08
1005 Wm  DATA 05,34,04,02,7a,02,01,63,01,00
1006 p4  DATA 85,80,80,46,c0,40,20,e0,20,11
1007 Ln  DATA 70,10,08,78,08,26,00,1e,14,08
1008 fd  DATA 78,04,20,40,20,80,20,12,a1,e0
1009 og  DATA 42,81,00,b9,00,80,59,80,40,20
1010 AN  DATA 20,20,16,10,10,09,88,08,04,b4
1011 3q  DATA 04,02,fa,02,01,47,01,00,ab,80
1012 nF  DATA 80,5b,c0,41,70,1f,c0,80,50,00
1013 IX  DATA 40,2a,00,20,12,80,10,09,c0,08
1014 1t  DATA 04,d8,44,1d,20,40,29,80,20,11
1015 Bf  DATA c0,10,09,e0,08,04,48,04,02,0c
1016 RI  DATA 22,0d,50,20,13,a0,10,08,30,08
1017 4m  DATA 05,58,04,02,1c,02,01,2e,01,00
1018 aC  DATA af,00,80,5f,80,40,22,20,20,10
1019 nC  DATA 90,10,09,c8,08,05,14,04,02,9a
1020 3I  DATA 02,01,43,01,00,89,80,80,4a,c0
1021 xt  DATA 40,2f,60,20,17,70,10,42,0f,f0
1022 KM  DATA 20,12,00,10,09,20,08,04,08,04
1023 9I  DATA 02,64,02,01,4a,01,00,8d,00,80
1024 I7  DATA 48,40,40,29,20,20,10,50,10,08
1025 0R  DATA a8,08,05,d4,04,02,5a,02,01,5d
1026 Yb  DATA 01,00,8d,80,82,90,01,50,a8,20
1027 op  DATA 40,80,46,85,41,a2,04,02,08,02
1028 Ta  DATA 01,54,01,00,96,00,80,50,80,40
1029 ak  DATA 27,42,a0,55,02,01,06,01,00,ab
1030 fV  DATA 00,80,5b,80,40,27,c0,20,14,10
1031 2Z  DATA 10,08,48,08,05,a4,04,02,66,02
1032 FC  DATA 01,5b,01,00,8d,0a,83,3c,08,25
1033 Cd  DATA 05,f8,08,0c,04,41,10,00,80,54
1034 wZ  DATA 00,40,2e,00,20,11,40,10,08,90
1035 iz  DATA e8,1e,40,80,40,80,40,2c,40,20
1036 ww  DATA 15,20,10,09,d0,08,04,08,74,00
1037 iY  DATA 04,3a,00,02,1d,02,e8,10,08,68
1038 oZ  DATA 08,05,b4,04,02,7a,02,01,23,01
1039 6p  DATA 05,a0,0f,02,01,68,01,00,82,00
1040 7y  DATA 80,4d,00,40,2d,80,20,12,21,30
1041 Oo  DATA 32,81,00,8d,00,80,51,80,40,22
1042 gb  DATA c0,20,17,60,10,09,88,08,05,14
1043 ze  DATA 04,02,6a,02,08,c0,de,04,02,50
1044 uK  DATA 02,01,24,1b,03,98,10,56,00,1c
1045 KE  DATA 17,06,48,10,0b,a0,08,05,70,04
1046 xJ  DATA 02,84,02,01,52,01,00,b5,00,80
1047 V8  DATA 5e,80,40,2c,c0,20,14,e0,10,09
1048 3Q  DATA f0,08,04,84,04,13,80,4c,20,08
1049 RP  DATA 19,04,9e,01,01,d7,04,04,01,00
1050 be  DATA 00,01,00,18,09,f4,00,00,03,eb
1051 7Y  DATA 00,00,0f,53,00,00,03,f2
(C) 1989 M&T
```

Listing 1. (Schluß)

Die Rätselmaschine

Teuflisch, dieses Spiel. Eine Wüste aus Buchstaben, in denen Wörter versteckt sein sollen. Unmöglich, es sind doch keine zu entdecken. Oder doch? Versuchen Sie Ihr Glück! Erstellen Sie außerdem eigene Rätsel, benutzen Sie dabei Begriffe nach Ihren Wünschen.

Fast jeder kennt sie, die »Textsalat« genannten Suchspiele in diversen Rätselmagazinen. In einem scheinbar sinnlos zusammengewürfelten Feld aus Buchstaben verstecken sich dabei Worte. Aufgabe des leidgeprüften Rätselfuchses ist es, diese an den Tag zu bringen. Das Schwierige daran ist, daß die Begriffe nicht nur waagrecht und senkrecht, sondern auch diagonal versteckt sein können. Ebenso können die Wörter auch von hinten nach vorne geschrieben sein.

Scharfblick ist gefragt

Dem begeisterten Rätsel-Freak bieten wir hier eine Computer-Variante dieses kniffligen Spieles. Aus vorgegebenen Begriffen, die jederzeit erweitert werden können, bastelt der Amiga immer wieder neue, schwierigere Rätsel. Besitzen Sie einen Drucker, können Sie eine private Rätselproduktion für Ihre Freunde und Bekannten eröffnen. Der besondere Reiz an selbstgemachten Rätseln liegt darin, daß Sie selbst bestimmen können, welche Begriffe im Buchstabensalat versteckt werden. So können individuelle und persönliche Rätsel erstellt werden. Ein Beispiel: Sie erstellen ein Rätsel für Ihre Oma und verstecken die Namen aller Familienmitglieder im Salat.

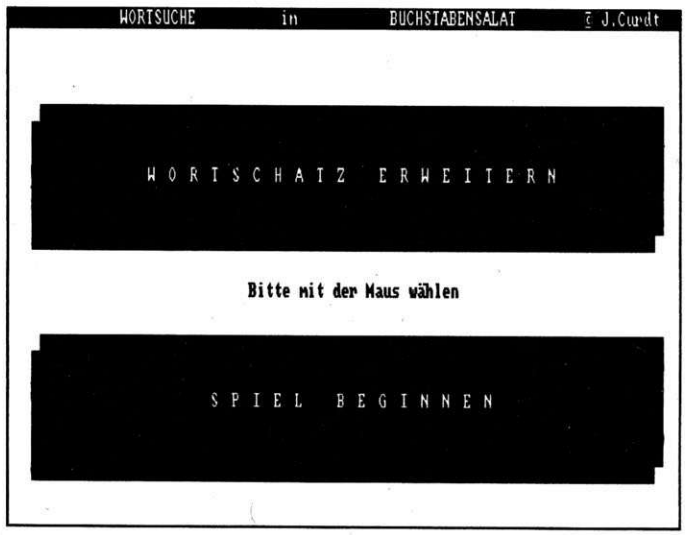


Bild 1. Nach dem Start gibt es zwei Auswahlmöglichkeiten

MONITOR	optisches Ausgabegerät	FREQUENZ	Schwingungshäufigkeit
FESTPLATTE	schneller externer Speicher	SILIZIUM	Halbleiterelement
ANIMATION	bewegte Darstellung	KONDENSATOR	kapazitives Bauelement
BUCHSTABE	alphanumerisches Zeichen	GRAFIK	zeichnerische Darstellung
ZIFFER	numerisches Zeichen	MOTHERBOARD	Hauptplatine
SCHALTER	variabler Kontakt	PERIPHERIE	Zubehör-Umgebung
RESET	Neustart des Systems	KABEL	isoliertes Drahtbündel
TRANSISTOR	Halbleiter-Bauelement	LEUCHTDIODE	Halbleiter-Lichtquelle
MAUS	mechanisches Eingabegerät	STRING	Zeichenkette
CURSOR	Eingabemarke	COLUMN	Druckspalte
CODIEREN	verschlüsseln	SCROLLEN	Bildschirminhalt bewegen
IMPEDANZ	Wechselstromwiderstand	MODEM	DFÜ- Sende- und Empfangsgerät
BITMAP	Bildschirmspeicher	DRUCKER	Gerät zur Ausgabe auf Papier
BACKUP	Sicherheitskopie	FORMFEED	Vorschub um Formularlänge
COMPILER	Übersetzungsprogramm	PROZESSOR	Zentraleinheit
DIRECTORY	Inhaltsverzeichnis einer Disk	BLITTER	Spezial-Grafik-Chip
DEBUGGER	Hilfsprogramm zur Fehlersuche	TASK	selbständiger Programmprozeß
EXPANSION	(Speicher-)Erweiterung	DISKETTE	verbreiteter Datenträger
PLOTTER	zeichnendes Ausgabegerät	COPY	Datenduplikat
REQUEST	Anforderung einer Eingabe	DATENFILE	Liste von Daten
TASTATUR	Gerät zur Eingabe	PIXEL	kleinstes Element einer Grafik
LINEFEED	Zeilenvorschub	MODULA	Compiler-Hochsprache
HANDBUCH	Gebrauchsanweisung	LINEFEED	Zeilenvorschub
MESSAGE	Nachricht vom System	SCREEN	Bildschirm
INPUT	Eingabe	MEMORY	Speicher/Gedächtnis
POINTER	Zeiger	ADRESSE	bestimmter Speicherplatz
WINDOW	Fenster für Ein- und Ausgabe	EMULATOR	Hardware-Simulation
BUS	Datenleitung	FLIPFLOP	einfacher elektron. Speicher
INTERFACE	Schnittstelle	NETZTEIL	Stromversorgung
ARRAY	Variablenfeld	BUBBLESORT	bekannte Sortier-Routine
DATEI	geordnete Sammlung von Daten	GENLOCK	Bildmischer
PATTERN	Zeichenmuster	HARDWARE	physikal. Teile eines Rechners
ADVENTURE	Abenteuer-Spiel	KOMPATIBEL	verträglich
SLOT	Platinen-Steckplatz	KERNEL	Kern des Betriebssystems
FUSE	elektrische Sicherung	LINKER	Programm zum Verbinden
AMPLITUDE	Schwingungshöhe	BREAKPOINT	testweise eingefügter Stop
AMPERE	Maß der Stromstärke	PATCH	kleine Programmflickerei

Tabelle 1. Ein Grundwortschatz für »Textsalat«, beliebige Änderungen sind vorzunehmen

Damit Sie in den Genuß von »Textsalat« kommen können, geben Sie bitte Listing 1 mit dem Checksummer ein. Beachten Sie die Eingabehinweise auf Seite 159. Der Speicher wird vom Programm fast bis zum letzten Byte benutzt. Daher empfiehlt es sich für die Amiga-Besitzer, die nur 512 KByte Speicher besitzen, vor dem Programmstart möglichst alle Windows (Fenster) zu schließen. Sollte der Amiga trotzdem mit einer »Out of Heap-Space«-Meldung den Dienst versagen, hilft das Abschalten der zweiten Floppy-Station vor dem Booten. Das Basic-Programm »Textsalat« benötigt zum Arbeiten außerdem die »graphics.bmap« (unter dem Namen »graphics_lib.fd« auf der Extras-Diskette). Konvertieren Sie diese Datei bitte zuerst mit dem Programm »FD1.2«. Die genaue Beschreibung dieser Wandlung finden Sie im Artikel »Amiga-Basic im Höhenflug« auf Seite 28 in diesem Sonderheft. Die in »graphics.bmap« konvertierte Datei muß sich im selben Verzeichnis wie »Textsalat« befinden. Auf der Programmservice-Diskette finden Sie die bereits gewandelte Datei.

Erstellen eigener Wortlisten

Bevor das erste Rätsel vom Computer generiert werden kann, muß eine Wortliste erstellt werden. Schließlich benötigt das Programm Begriffe, die in das Rätsel eingebaut werden sollen. Dies bereitet jedoch keinerlei Schwierigkeiten: Beim ersten Programmstart weist das Programm darauf hin, daß noch keine Wortliste vorhanden ist und fordert Sie auf diese zu erstellen. Daraufhin öffnet sich ein Eingabefenster, über das die Wortliste angelegt (später auch erweitert oder verändert) werden kann. Sie können nun Zeile für Zeile die entsprechenden Begriffe zusammen mit ihrem Synonym oder der Umschrei-

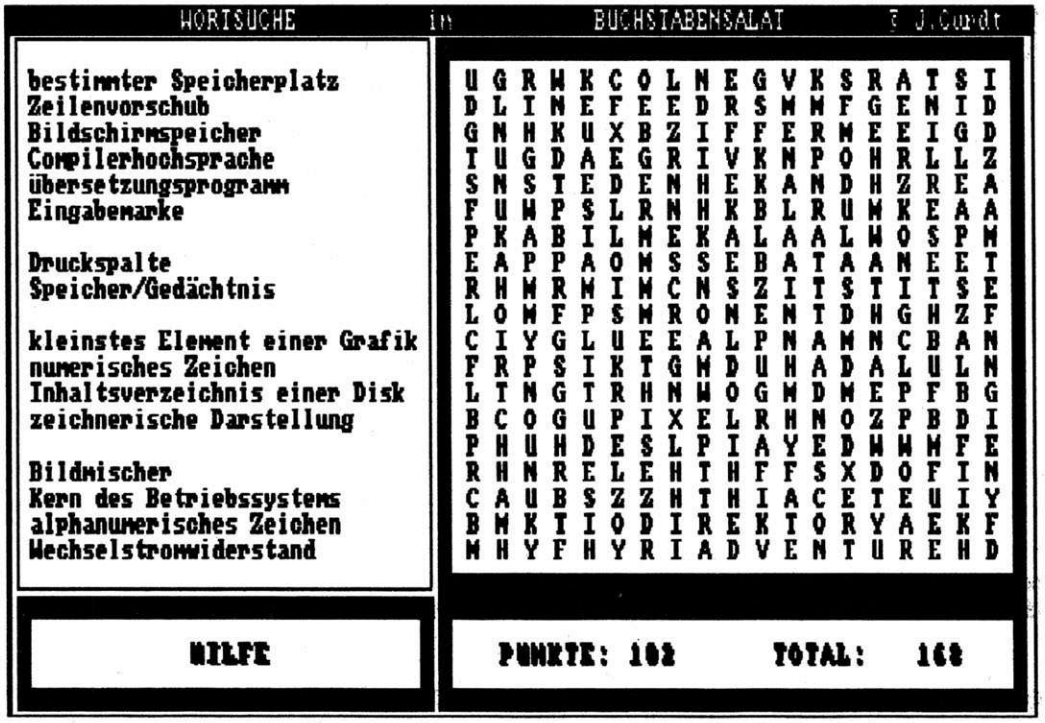


Bild 2. Das Rätsel wartet darauf, gelöst zu werden, meistern Sie die Herausforderung

bung eingeben. Als kleine Anregung finden Sie in Tabelle 1 eine kleine Zusammenstellung von Begriffen aus der Computerwelt (zusammen mit ihrer Umschreibung).

Zu beachten ist, daß der Wortschatz mindestens 19 verschiedene Begriffe umfassen muß. Interessant und abwechslungsreich werden die Rätsel aber erst mit einem Bestand ab etwa 50 Worten. Mit den Cursor-Tasten kann die Wortliste auf- und abwärts gescrollt werden. Haben Sie die Liste fertiggestellt, drücken Sie die ESC-Taste. Dann wird automatisch eine Datei mit dem Namen »Wortschatz« auf Diskette erzeugt. Dabei ist zu beachten, daß nur bis zur aktuellen Eingabezeile gespeichert wird.

Bei jedem weiteren Start des Programms wird nun die Wortliste automatisch geladen und es erscheint ein Auswahlmenü (Bild 1). Wählen Sie mit der Maus »Wortschatz erweitern«,

Wer ist schneller

kann der Wortschatz ergänzt oder verändert werden. Bei »Spiel beginnen« wird ein neues Rätsel generiert. Trotz der intensiven Rechenarbeit erscheint das erste Rätsel schon nach wenigen Augenblicken auf dem Bildschirm (Bild 2). Wenn überdurchschnittlich viele lange Worte unterzubringen sind, müssen Sie etwas mehr Geduld aufbringen. In jedem Fall muß anschließend der Amiga sehr viel länger warten. Denn nun ist es Ihre Aufgabe, das Rätsel zu lösen.

In der linken Bildschirmhälfte sehen Sie eine Liste der Umschreibungen aller Begriffe, die auf der rechten Seite im »Salat« versteckt sind. Haben Sie einen Begriff entdeckt, so klicken Sie nacheinander den Anfangs- und Endbuchstaben dieses Wortes an, worauf dieses markiert wird. Nun suchen Sie die zum Begriff passende Umschreibung auf der linken Seite und klicken diese ebenfalls an. War die Lösung richtig, verschwindet die entsprechende Umschreibung aus der Liste und Sie erhalten wertvolle Punkte. Die Anzahl der Punkte hängt von der Länge des Wortes und der Zeit ab, die zum Suchen benötigt wurde. Sie wird rechts unten neben den Gesamtpunkten angezeigt. Punktabzüge gibt es, wenn Sie von der Funktion »HILFE« Gebrauch machen.

Bei deren Anwendung werden kurzzeitig alle versteckten Begriffe im »Buchstabensalat« farblich hervorgehoben.

Ein Pull-Down-Menü am linken oberen Rand bietet noch verschiedene Auswahlmöglichkeiten:

Spielanleitung zeigen
Als Hilfestellung wird eine kurze Erklärung zum Spiel angezeigt.

Wortliste erweitern
Sie gelangen ins oben beschriebene Editierfenster.

Neues Rätsel geben
Der Computer generiert ein neues Rätsel

Rätsel ausdrucken
Das am Bildschirm angezeigte Rätsel wird auf einem angeschlossenen Drucker ausgegeben. Bild 3 zeigt ein ausgedrucktes Rätsel.

Ende
Beendet das Programm.

Dann also noch viel Spaß mit »Textsalat«, und lassen Sie sich vom Computer nicht in die Enge treiben. Noch etwas: Wenn Sie mehr als 5000 Punkte erreichen, dürfen Sie stolz sein auf Ihren Scharfblick!

(Jürgen Curdt/Martin Jobst/rs)

Wortsuche im Textsalat

```
SEFIZTRMYMTSKGCOGHT      * geordnete Sammlung von Daten
KSJEKEWNFSFBCUPPBAC      * physikal. Teile eines Rechners
RGWATMSRYRIGTNASURE      * Inhaltsverzeichnis einer Disk
PGSTUHOISHEHZITECDN      * Datenduplikat
FNOILJNDLGSQMKCIHWT      * kleine Programmflickerei
HLLUHILZEIUHUTHKSAR      * DFü - Sende- und Empfangsgerät
PCOMPILERMZHOEERTRO      * bestimmter Speicherplatz
OIBFMKGGFEWIHINFAEN      * kleinstes Element einer Grafik
RDNRCERRLQHSUTFZBII      * übersetzungsprogramm
WPAFECSERQDZNMTREAC      * Gerät zur Ausgabe auf Papier
SDLIHAOSTOEQEPALAYS      * Schwingungshäufigkeit
AIIBNIKPAVLEAIFGNAO      * Nachricht vom System
UDTRITBPYGALGAJNGFJ      * testweise eingefügter Stop
EHRNEPEKOTEGENLOCKW      * Bildmischer
DATEIKIRTIRGGEAEADS      * zeichnendes Ausgabegerät
WNZSSNTXFRNDRUCKERZ      * alphanumerisches Zeichen
KUIEWSHOEAHTHFEAFKZ      * Schnittstellen-Norm
IWPOJNENRLCDLLXJZBM      * Schnittstelle
ZOLFHMGOPYTEKHKFIFZ      * Halbleiterelement
```

Bild 3. Die Rätsel können auch auf dem Drucker ausgegeben werden

Programmname:	Textsalat
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic
Bemerkung:	Benötigt graphics.bmap (s.Text)

```
Programmautor: Jürgen Curdt
---------------------------

 1 HOO '*********************************************
 2 4V  '*                                           *
 3 1Q  '*        T E X T S A L A T    1.0           *
 4 6X  '*                                           *
 5 Ie  '*  AMIGA 512K + AmigaBasic + Graphics.bmap  *
 6 8Z  '*                                           *
 7 NU  '*********************************************
```

```
 8 4v  file$="wortschatz"
 9 iI  DECLARE FUNCTION  move& LIBRARY
10 4A  LIBRARY "graphics.library"
11 Ju  DEFINT a-e: DEFINT g-z
12 Ng  WINDOW CLOSE 1: SCREEN 1,640,246,4,2
13 E8  w$=SPACE$(13)+"WORTSUCHE          im"
14 gN  w$=w$+SPACE$(11)+"BUCHSTABENSALAT        (w) J.Curdt"
15 jG  WINDOW 3,w$,,16,1: ziff$="# # # # # #": feld=19
16 GS  abc$= "AAAAAABBBBCCDDDDEEEEEEEFFFFF"
17 bV  abc$=abc$+"GGGGGHHHHHIIIIIIJJKKKKLLL"
18 52  abc$=abc$+"MMMMNNNNNNNNNOOOOPPQRRRRR"
19 By  abc$=abc$+"SSSSSTTTTTUUUVVWWWXYZZ"
20 uM  DIM offset(feld): DIM wort$(feld),bed$(feld),Wert(feld)
21 qr  DIM SHARED farbfeld((feld+1)**2)
22 Gx  FOR i = 1 TO 4: MENU i,0,1,"": NEXT
23 3t  ON BREAK GOSUB finit: BREAK ON
24 oQ  FOR i=0 TO 15
25 rU2   FOR j=1 TO 3: READ farbwert(j): NEXT j
26 TD    PALETTE i,farbwert(1),farbwert(2),farbwert(3)
27 s80 NEXT i
28 VA  DATA  .13 , .13 ,  .33
29 bX  DATA  .30 , .60 ,  .16
30 wg  DATA 0 ,    .53 , 1
31 T7  DATA 0 ,    0   , 0
32 U8  DATA 0 ,    0   , 0
33 cI  DATA 1 ,    .13 , 0
34 ew  DATA .99 , .91  , .28
35 71  DATA .6 , 1     , .93
36 qt  DATA .46 , .86 , 1
37 TP  DATA 0   , .8  , .66
38 pT  DATA 0   , .73 , .6
39 zD  DATA 0   , .66 , .46
40 Vs  DATA 0   , .46 , .33
41 QW  DATA 0   , .33 , .26
42 U2  DATA 0   , .26 , .2
43 hR  DATA .8  , .8  , .8
44 id  startfenster:
45 fi  GOSUB laden
46 Ju  FOR i = 0 TO 8 STEP 8
47 o62   LINE (20+i,43-i)-(610+i,103-i),4+i,bf
48 tE    LINE (20+i,152-i)-(610+i,212-i),4+i,bf
49 fR    LINE (20+i,43-i)-(610+i,103-i),4,b
50 Ee    LINE (20+i,152-i)-(610+i,212-i),4,b
51 uz0 NEXT
52 90  COLOR 5: Abstand -3,12:    PRINT "Bitte mit der Maus wählen
       "
53 M1  s$="WORTSCHATZ ERWEITERN": shadow s$,130,70,20,6,4,4
54 2W  s$="SPIEL BEGINNEN":       shadow s$,190,177,20,6,4,4
55 Nf  GOSUB warten: CLS
56 mr  IF my<125 THEN GOSUB eingabe: GOSUB sichern: WINDOW CLOSE
       4
57 AV  GOSUB info
58 wS  ERASE wfeld$, bfeld$
59 tw  GOSUB laden
60 UY  start:
61 ao  WINDOW OUTPUT 3
62 ub  FOR i = 1 TO 4: MENU i,0,1,"": NEXT
63 cH  LINE (1,1)-(630,232),0,bf
64 Sn  LINE (10,201)-(254,224),15,bf
65 LV  FOR i = 0 TO 7
66 IH2   LINE (262+i,1)-(628-i,192-i),7+i,b
67 TT    LINE (2+i,193+i)-(258-i,232-i),7+i,b
68 oH    LINE (262+i,193+i)-(628-i,232-i),7+i,b
69 CH0 NEXT
70 yo  LINE (2,0)-(257,190),6,b
71 ZR  LINE (270,8)-(620,184),15,bf
72 11  LINE (270,201)-(620,224),15,bf
73 rR  s$="HILFE": shadow s$,108,215,10,3,6,2
74 qi  s$="PUNKTE:          TOTAL:": shadow s$,300,215,10,5,3,1
75 Iw  zaehler=0: Wortwahl=0
76 ld  LINE (5,5)-(250,185),0,bf
77 yG  feld$=SPACE$((feld+1)*(feld))
78 pR  FOR i = feld+1 TO (feld+1)*(feld) STEP feld+1
79 FS2   MID$(feld$,i,1)=CHR$(43)
80 NS0 NEXT
81 E1  feld$=feld$+STRING$(feld+1,CHR$(42))
82 fZ  WHILE zaehler <feld
83 zf2   okayflag=0: RANDOMIZE TIMER
84 lN0 suchneueswort:
85 mT2   Wortwahl=INT(RND(1)*Wortzahl)+1
86 ua    wort$=wfeld$(Wortwahl): i=0
87 tc    WHILE i<zaehler
88 XJ5      IF wort$=wort$(i) THEN suchneueswort
89 RL       i=i+1
```

```
90 th2   WEND
91 sy    wort$(zaehler)=wort$: Wortlaenge=LEN(wort$)
92 80    bed$=bfeld$(Wortwahl): bed$(zaehler)=bed$
93 Eo    Drehflag=INT(RND(1)+.2): IF Drehflag THEN CALL Drehe (wor
         t$)
94 r7    Richtungflag=INT(RND(1)*4)+1: Startrichtung=Richtungflag
95 Ag0 richtung:
96 Ht2   ON  Richtungflag GOTO waagerecht,senkrecht,diauf,diab
97 C4    waagerecht:
98 4a4     Schritt=1: Schritt2=feld+1: GOTO startwahl
99 2h2   senkrecht:
100 pf4    Schritt=feld+1: Schritt2=1: GOTO startwahl
101 1c2  diauf:
102 SZ4    Schritt=feld: Schritt2=1: Drehe wort$: GOTO startwahl
103 Q82  diab:
104 VM4    Schritt=feld+2: Schritt2=feld+1
105 cQ0 startwahl:
106 9u2  exflag=INT(RND(1)*feld): start=exflag+1
107 kI4    WHILE start<>exflag AND okayflag =0
108 Tu     test$=""
109 Yd     FOR i=0 TO feld
110 1v6      t=start*Schritt2+i*Schritt+1
111 kB       IF t>(feld+1)**2 THEN t=t-(feld+1)**2
112 2z       test$=test$+MID$(feld$,t,1)
113 uz4    NEXT
114 sW     offset=0: moeglich=0
115 hD     WHILE offset+Wortlaenge<feld
116 xn6      diff=0: posit=0
117 cH       WHILE diff=0 AND Wortlaenge>posit
118 go8        posit=posit+1
119 kD         t$= MID$(test$,posit+offset,1)
120 A4         IF t$<>" " AND t$<>MID$(wort$,posit,1) THEN dif
               f=1
121 OC6      WEND
122 Cf       IF diff=0 THEN
123 eE8        moeglich=moeglich+1: offset(moeglich)=offset
124 wp6      END IF
125 eu       IF diff THEN offset=offset+posit ELSE offset=offset+1
126 TH4    WEND
127 yj     IF moeglich THEN
128 DM6      moeglich=INT(RND(1)*moeglich)+1
129 tF       anfang=offset(moeglich)
130 vq       j=0
131 7B       FOR i = anfang TO anfang+Wortlaenge
132 EA9         j=j+1
133 OI          t=start*Schritt2+i*Schritt+1
134 7Y          IF t>(feld+1)**2 THEN t=t-(feld+1)**2
135 cS          MID$(feld$,t,1)=MID$(wort$,j,1)
136 HM6      NEXT
137 kX       Abstand -15,zaehler: COLOR 11,0: PRINT bed$;
138 yz       okayflag=1: Wert(zaehler)=(Richtungflag*(Drehflag+1)*
             (feld-Wortlaenge))/10+1
139 oc       zaehler=zaehler+1
140 C54    END IF
141 dl     IF start<feld THEN start=start+1 ELSE start=0
142 jX2  WEND
143 B0   IF Richtungflag<4 THEN Richtungflag=Richtungflag+1 ELSE
         Richtungflag=1
144 1o   IF Richtungflag<>Startrichtung THEN richtung
145 ma0 WEND
146 yr  'raetselfeld ausgeben
147 jR2  WINDOW CLOSE 4
148 6N   FOR i = 0 TO feld-1
149 5p4    FOR j = 1 TO feld
150 4P6      neu=i*(feld+1)+j: farbe=4
151 BD       aus$ = MID$(feld$,neu,1)
152 dX       IF aus$ =CHR$(32) THEN
153 7E8        aus=INT(RND(1)*100)+1: aus$= MID$(abc$,aus,1)
154 XN         MID$(feld$,neu,1)=aus$
155 3w         farbe=3
156 SL6      END IF
157 FG       COLOR farbe,15: CALL Abstand (j-1,i): PRINT  aus$;
158 kS       farbfeld(neu)=farbe
159 ej4    NEXT
160 fk2  NEXT
161 QS   MENU 1,0,1,"      Bitte wählen:      "
162 MX   MENU 1,1,1," Spielanleitung zeigen "
163 2H   MENU 1,2,1,"  Wortliste erweitern  "
164 P5   MENU 1,3,1,"  neues  Rätsel geben  "
165 80   MENU 1,4,1,"   Rätsel ausdrucken   "
166 cJ   MENU 1,5,1,"          ENDE          "
```

Listing 1. Geben Sie »Textsalat« bitte mit dem Checksummer (Seite 159) ein

```
167 vY    ON MENU GOSUB menutest
168 k3    ON TIMER(2) GOSUB zeitzaehler: TIMER ON
169 cP    Zeit=50: Summe=0
170 Ab    my=100: mx=200: second=1: GOSUB auswahl: 'Dummy-Lauf durc
          h Auswahl
171 rl0 mcheck:
172 iX2   IF zaehler<1 THEN TIMER OFF: GOTO fertig
173 VX    MENU ON: GOSUB warten: GOSUB auswahl
174 Kn0 GOTO mcheck
175 3B  zeitzaehler:
176 n02   IF Zeit>1 THEN Zeit=Zeit-1
177 Dp0 RETURN
178 YI  finit:
179 ze2   LIBRARY CLOSE: MENU RESET: WINDOW CLOSE 3: SCREEN CLOSE 1
180 eZ    END
181 Ht0 RETURN
182 UG  auswahl:
183 ae2   IF my>193 THEN
184 dK4     IF mx>262 THEN RETURN
185 kr      GOSUB hilfe: RETURN
186 wp2   END IF
187 zM    IF second AND mx<260 AND my <180 THEN
188 AS4     Erfolg=0
189 bz      y=(my-11)\9:
190 zb      IF Find$=wort$(y) THEN Erfolg=1
191 JY      Drehe Find$: IF Find$=wort$(y) THEN Erfolg=1
192 bm      IF Erfolg THEN
193 Zu7        Abstand -15,y: COLOR 0,0: PRINT  SPACE$(30)
194 7X         Punkte=Wert(y)*Zeit/(Hilfen+1)
195 Vk         Summe=Summe+Punkte
196 oB         LINE(370,202)-(426,220),15,bf: LINE(550,202)-(620,22
               0),15,bf
197 Ee         s$=STR$(Punkte): shadow s$,370,215,10,5,3,1
198 1G         s$=STR$(Summe): shadow s$,550,215,10,5,3,1
199 BQ         Hilfen=0: Zeit=50: zaehler=zaehler-1
200 Uo         Markflag=5
201 qZ4     ELSE
202 2g7        Markflag=0
203 D64     END IF
204 tB      second=0: first=0
205 pI      GOSUB eintragen: RETURN
206 G92   END IF
207 7J    IF mx<280 OR mx>630 THEN RETURN
208 gN    IF my<12  OR my>180 THEN RETURN
209 7o    IF second THEN
210 c94     second=0: first=0: Markflag=0
211 vO      GOSUB eintragen: RETURN
212 MF2   END IF
213 To    IF first THEN
214 lE4     second=1: x2=(mx-278)\18: y2=(my-11)\9
215 zS      GOSUB eintragen: RETURN
216 QJ2   END IF
217 Hu    first=1: Markflag=6
218 f4    x1=(mx-278)\18: y1=(my-11)\9
219 jT    markiere x1,y1: Find$=""
220 uW0 RETURN
221 rM  eintragen:
222 xH  IF x2=x1 THEN
223 Ue2   d=ABS(y2-y1): x=x1: aw=y1: IF aw>y2 THEN aw=y2
224 XQ    FOR i = 0 TO d
225 G44     y=aw+i: markiere x,y
226 Dz2   NEXT: RETURN
227 bU    END IF
228 BX0 IF y2=y1 THEN
229 KW2   d=ABS(x2-x1): y=y1: aw=x1: IF aw>x2 THEN aw=x2
230 dW    FOR i = 0 TO d
231 K74     x=aw+i: markiere x,y
232 J52   NEXT: RETURN
233 ha0 END IF
234 hW  IF ABS(x2-x1)<>ABS(y2-y1) THEN
235 1Y2   second=0: first=0: Markflag=0
236 Vf    markiere x1,y1: markiere x2,y2
237 Bn    RETURN
238 mf0 END IF
239 mP  IF x2<x1 THEN SWAP x1,x2: SWAP y1,y2
240 qr2   steig=1: IF y2<y1 THEN steig=-1
241 jg    d=x2-x1: IF aw>x2 THEN aw=x2
242 pi    FOR i = 0 TO d
243 uL4     x=x1+i: y=y1+(steig*i): markiere x,y
244 162   NEXT
245 Jv0 RETURN
246 F1  fertig:
247 2x2   WINDOW 4,"",(3,1)-(257,230),0,1
248 WT    LOCATE 2,16: COLOR 6,0: PRINT "Die"
```

```
249 dp    s$="Liste der Besten": shadow s$,20,25,14,6,5,1
250 pw    COLOR 11
251 yy    FOR i= 0 TO 9
252 wl4     LOCATE i*2+6,3: PRINT USING  ziff$;TOTAL(i);
253 Qz      LOCATE ,12: PRINT sieg$(i)
254 BG2   NEXT
255 Zj    IF TOTAL(i-1)<Summe THEN
256 cd4     SWAP TOTAL(i), Summe
257 Ba      COLOR 5: LOCATE 26,3
258 Mg      PRINT "Du darfst Dich eintragen...": LINE (17,210)-(240
            ,226),5,b
259 vx      LOCATE 28,4
260 Sg      holestring spieler$,20: sieg$(i)=spieler$
261 NJ      i=i-1
262 uZ      FOR i = 9 TO 0 STEP-1
263 Ok6       IF TOTAL(i) < TOTAL(i+1) THEN
264 Mt8         SWAP TOTAL(i),TOTAL(i+1)
265 oO          SWAP sieg$(i),sieg$(i+1)
266 E76       END IF
267 OT4     NEXT
268 R1      COLOR 11: LINE (0,30)-(254,240),0,bf
269 GG      FOR i= 0 TO 9
270 E36       LOCATE i*2+6,3: PRINT USING  ziff$;TOTAL(i);
271 iH        LOCATE ,12: PRINT sieg$(i)
272 TY4     NEXT
273 7d      GOSUB sichern
274 MF2   END IF
275 Ts    COLOR 5: LOCATE 26,3
276 Sr    PRINT "FÜr ein neues Spiel bitte die": LOCATE 27,3
277 gt    PRINT " linke  Maustaste  drücken!"
278 CZ    GOSUB warten: WINDOW CLOSE 4: GOTO start
279 6t0 menutest:
280 bM2   IF MENU(0)<>1 THEN RETURN
281 ek    ON MENU(1) GOTO 1,2,3,4,5
282 SL0 1 GOSUB info: GOSUB warten: WINDOW CLOSE 4: RETURN
283 9r  2 GOSUB eingabe: GOSUB sichern: WINDOW CLOSE 4: RETURN
284 lR  3 CLS: GOTO start
285 Nr  4 GOSUB ausdruck: RETURN
286 WS  5 GOTO finit
287 zb  RETURN
288 7T  warten:
289 Ao2   WHILE MOUSE(0)<=0: SLEEP: WEND
290 WO    mx=MOUSE(1): my=MOUSE(2)
291 3f0 RETURN
292 FP  info:
293 K32   WINDOW 4,"",(265,11)-(627,230),0,1
294 fo    s$="IN DIESEM FELD WERDEN WOERTER VERSTECKT."
295 As    shadow s$,16,32,8,6,5,1
296 Cr    LOCATE 7,1: COLOR 15
297 Y1    PRINT " Sie können senkrecht, waagerecht oder auch
298 w7    PRINT " diagonal  und  manchmal  sogar  verdreht,
299 hm    PRINT " also  mit dem  letzten  Buchstaben  voran,
300 c3    PRINT " eingetragen sein.": PRINT
301 PZ    PRINT " Ist ein Wort gefunden, bitte ": COLOR 6
302 rU    PRINT " den ersten und den letzten Buchstaben"
303 dl    PRINT " in beliebiger Reihenfolge anklicken,"
304 cg    PRINT " dann den dazu passenden Suchbegriff!": COLOR 15
305 1B    PRINT " IrrtÜmlich markierte Buchstaben werden mit"
306 XV    PRINT " einem dritten Mausklick gelöscht.": PRINT
307 Fu    PRINT " Das Finden verdrehter, diagonaler und sehr"
308 Gd    PRINT " kurzer Wörter wird mit hohen Punktzahlen"
309 KM    PRINT " belohnt."
310 Gv    PRINT " Auch Schnelligkeit bringt Zusatzpunkte.": PRINT:
          PRINT
311 At    PRINT " Die Funktion HILFE zeigt kurz alle Einträge,
312 OK    PRINT " kostet aber wertvolle Punkte!"
313 P10 RETURN
314 ze  hilfe:
315 aU2   PALETTE 4,0,1,0: FOR i=0 TO 500: NEXT: PALETTE 4,0,0,0
316 30    Hilfen=Hilfen+1
317 T50 RETURN
318 2g  ausdruck:
319 PP2   OPEN "prt:" FOR OUTPUT AS #2
320 xr4     PRINT#2, CHR$(13)
321 TH2   CLOSE #2
322 Ct    OPEN "par:" FOR OUTPUT AS #2
323 tJ4     PRINT#2, CHR$(27);CHR$(64):          'Drucker normiere
            n
324 hH      PRINT#2, CHR$(27);CHR$(51);CHR$(36): 'Zeilenabstand  3
            6/216''
325 PS      PRINT#2, CHR$(27);CHR$(71);CHR$(14);:'Doppeldruck und
            Sperrschrift
326 xZ      PRINT#2, "    Wortsuche im Textsalat"
327 NC      PRINT#2, CHR$(27);CHR$(77);CHR$(13): 'Elite ein
```

```
328 aO2   CLOSE #2
329 Lu3    FOR i=0 TO feld-1
330 K14    OPEN "par:" FOR OUTPUT AS #2
331 5K     PRINT#2, CHR$(14);:                    'Sperrschrift ein
332 eS     CLOSE #2
333 dd     OPEN "prt:" FOR OUTPUT AS #2
334 UT6      FOR j= 1 TO feld
335 ZV8        PRINT#2, MID$(feld$,i*(feld+1)+j,1);
336 Va6      NEXT
337 jX4    CLOSE #2
338 S9     OPEN "par:" FOR OUTPUT AS #2
339 Os6      PRINT#2, CHR$(20);:                  'Sperrschrift aus
340 ma4    CLOSE #2
341 ll     OPEN "prt:" FOR OUTPUT AS #2
342 HQ6      PRINT#2, "       * ";bed$(i);CHR$(13)
343 pd4    CLOSE #2
344 di2   NEXT
345 pp    OPEN "prt:" FOR OUTPUT AS #2
346 ik4     PRINT#2, CHR$(13); CHR$(10)
347 th2   CLOSE #2
348 ya0 RETURN
349 ZR  laden:
350 Nm  OPEN file$ FOR APPEND AS #1
351 fm2   IF LOF(1) <= 0 THEN
352 sg4     LOCATE 2,7
353 eG      PRINT "Bevor das erste Rätsel ausgegeben werden kann, m
          uß eine Wörterliste"
354 gE      PRINT SPACE$(17);"mit wenigstens 19 Begriffen angelegt
          werden!"
355 xh      CLOSE #1: DIM wfeld$(100),bfeld$(100)
356 LT      GOSUB eingabe: GOSUB sichern: WINDOW CLOSE 4
357 s4      ERASE wfeld$,bfeld$: CLS: GOTO startfenster
358 ib2   END IF
359 1o0 CLOSE #1
360 yW  OPEN file$ FOR INPUT AS #1
361 hQ2   FOR i= 0 TO 9: INPUT #1,sieg$(i),TOTAL(i): NEXT
362 Mi    INPUT #1,Wortzahl: DIM wfeld$(Wortzahl+100),bfeld$(Wortz
          ahl+100)
363 Q3    FOR i = 0 TO Wortzahl: INPUT #1,wfeld$(i),bfeld$(i): NEX
          T
364 gQ    CLOSE 1
365 Fr0 RETURN
366 1K  sichern:
367 J12   OPEN file$ FOR OUTPUT AS 1
368 rr    FOR i= 0 TO 9
369 ig4     IF sieg$(i)="" THEN sieg$(i)="NN"
370 cX      WRITE #1,sieg$(i),TOTAL(i)
371 492   NEXT
372 d0    WRITE #1,Wortzahl
373 r5    FOR i = 0 TO Wortzahl: WRITE #1,wfeld$(i),bfeld$(i): NEX
          T
374 qa    CLOSE 1
375 P10 RETURN
376 17  eingabe:
377 Bm2   w$=SPACE$(12)+"Wortschatz  ändern  oder  ergänzen"
378 V3    WINDOW 4,w$,(80,50)-(550,172),0,1: COLOR 15,0
379 mi    LOCATE 4,8: PRINT "WORT                        BEDEUTUNG"
380 OI    LOCATE 14,1
381 3X    PRINT "Eingabe beenden und bis zur Eingabezeile speichern
          ...<ESC>";
382 LT    LOCATE 15,12: PRINT "Liste scrollen mit den Pfeiltasten";
383 5U    COLOR 5: AREA (144,38): AREA (160,28): AREA (176,38)
384 bc    AREA (176,80): AREA (160,90): AREA (144,80): AREAFILL
385 oU    LINE (0,38)-(470,80),5,b
386 DX    zaehler=Wortzahl
387 gq0 neu:
388 th2   COLOR 5,0: LOCATE 13,11
389 O0    PRINT "Die Eingabezeile zeigt das "zaehler+1". Wort      "
390 RT    LINE (1,39)-(143,79),7,bf: LINE (161,39)-(468,79),7,bf
391 jL    LINE (8,56)-(461,63),0,bf: p=0
392 3R    FOR i = zaehler-1 TO zaehler+3
393 BI4     COLOR 2,7: IF p=2 THEN COLOR 5,0
394 VL      IF i>=0 THEN
395 uw6       LOCATE 6+p,2: PRINT wfeld$(i)
396 ra        LOCATE 6+p,24: PRINT  bfeld$(i)
397 LE4     END IF
398 6E      p=p+1
399 Wb2   NEXT
400 Ym    LOCATE 8,2
401 Sx    hilf$=wfeld$(zaehler+1): CALL holestring(hilf$,17)
402 Ip    IF hilf$=CHR$(27) THEN Wortzahl=zaehler: RETURN
403 PN    IF hilf$=CHR$(28) THEN
404 LV4     IF zaehler THEN zaehler=zaehler-1
405 YH      GOTO neu
406 UN2   END IF
407 J9    IF hilf$=CHR$(29) THEN zaehler=zaehler+1: GOTO neu
408 pS    wort$=""
409 AV    FOR i= 1 TO LEN(hilf$)
410 3N4     ei$=MID$(hilf$,i,1)
411 K3      IF ei$=CHR$(252) THEN ei$="ue"
412 OM      IF ei$=CHR$(228) THEN ei$="ae"
413 Hd      IF ei$=CHR$(246) THEN ei$="oe"
414 E2      IF ei$=CHR$(223) THEN ei$="ss"
415 OE      wort$=wort$+UCASE$(ei$)
416 ns2   NEXT
417 UV    LOCATE 8,24: hilf$=bfeld$(zaehler+1): CALL holestring(hil
          f$,30)
418 Y5    IF hilf$=CHR$(27) THEN Wortzahl=zaehler: RETURN
419 fd    IF hilf$=CHR$(28) THEN
420 bl4     IF zaehler THEN zaehler=zaehler-1
421 oX      GOTO neu
422 kd2   END IF
423 ZP    IF hilf$=CHR$(29) THEN zaehler=zaehler+1: GOTO neu
424 UJ    zaehler=zaehler+1: wfeld$(zaehler)=wort$: bfeld$(zaehler)
          =hilf$
425 pK    IF zaehler>Wortzahl+99 THEN GOSUB sichern: RUN
426 tc    GOTO neu
427 Fr0 RETURN
428 1J  SUB holestring(ein$,maxlen) STATIC
429 3I2   xa=POS(0): y=CSRLIN: ei=1: x=LEN(ein$)
430 48    WHILE ei><13
431 sI4     COLOR 5,0: LOCATE y,xa: PRINT LEFT$(ein$,x);: PRINT "_"
            ;
432 Ke      IF x<LEN(ein$) THEN PRINT RIGHT$(ein$,LEN(ein$)-x);
433 ow      PRINT "  ";
434 im      abfrage: ei$=INKEY$ : IF ei$=""THEN abfrage
435 me      ei=ASC(ei$)
436 OI      IF ei>26 AND ei<30 THEN ein$=ei$: EXIT SUB
437 9L      IF ei=31 AND x>0 THEN x=x-1
438 kq      IF ei=30 AND x<LEN(ein$) THEN x=x+1
439 kD      IF ei=8 AND x>0 THEN ein$=LEFT$(ein$,x-1)+RIGHT$(ein$,
            LEN(ein$)-x): x=x-1
440 OY      IF ei=127 AND x<LEN(ein$) THEN ein$=LEFT$(ein$,x)+RIGH
            T$(ein$,LEN(ein$)-x-1)
441 hH      IF ei>44 AND ei<123 OR ei=32 OR ei>214 AND ei<253 T
            HEN
442 1I6       IF LEN(ein$)<maxlen THEN
443 sH8         ein$=LEFT$(ein$,x)+ei$+RIGHT$(ein$,LEN(ein$)-x): x=
                x+1
444 lU6       ELSE
445 Co8         GOTO abfrage
446 Oh6       END IF:
447 924     END IF
448 zk2   WEND: LOCATE y,xa: PRINT  ein$;"  "
449 HJ0 END SUB
450 Go  SUB markiere (x,y) STATIC
451 yW2   SHARED Markflag,feld$,feld,Find$,farbfeld
452 ol    neu=y*(feld+1)+x+1
453 4M    t$=MID$(feld$,neu,1): Find$=Find$+t$
454 OI    IF Markflag THEN COLOR Markflag,15 ELSE COLOR farbfeld (n
          eu),15
455 PF    IF Markflag=5 THEN farbfeld (neu)=5
456 iD    POKEW WINDOW(8)+36,x*18+280: POKEW WINDOW(8)+38,y*9+17
457 E1    PRINT t$;
458 QS0 END SUB
459 pD  SUB Abstand (x,y) STATIC
460 mH2   POKEW WINDOW(8)+36,x*18+280: POKEW WINDOW(8)+38,y*9+17
461 TV0 END SUB
462 fZ  SUB Drehe (dreh$) STATIC
463 Q02   hilf$=""
464 xN    FOR i=1 TO LEN(dreh$): hilf$=MID$(dreh$,i,1)+hilf$: NEXT
465 TU    dreh$=hilf$
466 Ya0 END SUB
467 xB  SUB shadow (text$,x%,y%,spacing%,vfarbe%,bfarbe%,tiefe%) ST
        ATIC
468 Rk2   CALL setdrmd&(WINDOW(8),0)
469 Dj    FOR i%=1 TO LEN (text$)
470 Sp4     b$=MID$(text$,i%,1)
471 Td      e&=move&(WINDOW(8),x%+tiefe%,y%+1)
472 Fd      COLOR bfarbe%: PRINT b$;
473 hy      COLOR vfarbe%: e&=move&(WINDOW(8),x%,y%): PRINT b$;
474 Yo      x%=x%+spacing%
475 Lg2   NEXT i%
476 fu    CALL setdrmd&(WINDOW(8),1)
477 j10 END SUB
(C) 1989 M&T
```

Listing 1. »Textsalat« (Schluß)

Bodenlos

Verschlagenheit ist Trumpf. Wer seinem Gegner die besten Fußangeln legt, trägt den Sieg davon. Mit diesem Grundsatz kommen Sie bei »Hinterhalt« am weitesten.

Hinterhalt« (Listing 1) ist buchstäblich ein Spiel mit doppeltem Boden. Öffnen Sie Ihren Gegnern die Falltüren für einen Sturz ins Bodenlose.

Das Spielfeld besteht aus 36 Feldern. Jedes Feld kann bis zu zwei Unterlagen haben. Der weiße Boden liegt über dem orangen; dadurch ergeben sich viele Schwierigkeiten:

Links vom Spielfeld (Bild 1) sehen Sie die Schieber für den weißen Boden. Am oberen Rand sind die für den (dahinterliegenden) orangen. Verändern Sie nun eine der beiden Einstellungen, so werden einige Löcher zugedeckt, andere entstehen an neuen Stellen. Liegt eine Kugel auf einem Feld, das eben noch »sicher« war, so kann es sein, daß ihr jetzt der Boden unter den Füßen weggezogen wird.

Wegen dieser Anordnung können Sie nicht erkennen, ob unter dem weißen noch ein oranges Feld ist oder ob Sie abstürzen, sobald der weiße Boden entfernt ist.

Stehen Sie dagegen schon auf einem orangen Feld, dann können Sie immerhin sicher sein, daß Sie in den Abgrund stürzen, sobald auch dieser verschoben wird. Allerdings hat jeder Spieler nicht nur eine, sondern mindestens fünf Spielkugeln (bei zwei Spielern sogar sieben). Sie sind also nicht gleich mit dem Absturz Ihrer ersten Kugel verloren.

Zu Beginn werden Anzahl und Namen der Spieler abgefragt. Darauf werden die Kontrahenten abwechselnd aufgefordert, ihre Kugeln zu setzen. Sind alle Kugeln im Spiel, so startet der erste Spieler mit einem Versuch, seine Gegner zu versenken. Dazu wird entweder ein weißer oder ein oranger Schieber um eine Position verstellt. Beide können senkrecht zueinander verschoben werden. Dadurch werden die Löcher dieser Schicht verschoben und eventuell Kugeln versenkt.

Hinterhalt läßt nicht zu, daß ein Schieber zweimal nacheinander bewegt wird. Zudem können die Schieber nur in sehr engen Grenzen bewegt werden.

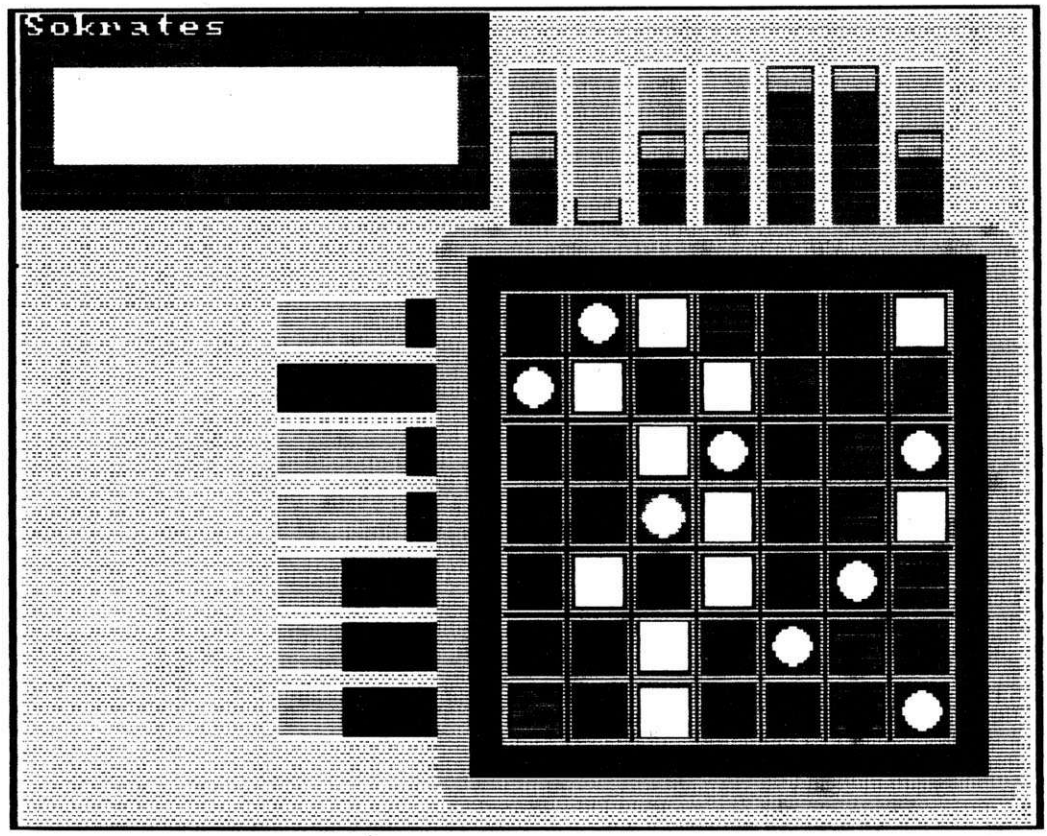


Bild 1. Die Ausgangsstellung von »Hinterhalt«. Die Schieber für das Versenken der Kugeln finden Sie am linken und oberen Spielfeldrand.

Hat ein Spieler keine Kugeln mehr im Feld, so ist er aus dem Rennen. Das Spiel ist beendet, wenn nur noch ein Teilnehmer im Rennen ist.

Wenn die letzten Kugeln zweier Spieler im selben Spielzug versenkt werden, endet das Spiel unentschieden.

Nach dem Ende eines Spiels werden Sie gefragt, ob Sie noch einen Durchgang wünschen. Wenn nicht, drücken Sie <ESC>, ansonsten bestätigen Sie mit <RETURN>.

Hinterhalt geben Sie mit dem Checksummer ein. Beachten Sie dabei die Eingabehinweise auf Seite 159. Starten Sie dann Amiga-Basic und laden Sie Hinterhalt mit »Open«. Sie können das Programm mit »Run« oder mit dem »Start«-Menü starten.

(Christian Buchner/so)

Programmname:	Hinterhalt
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

```
Programmautor: Christian Buchner
-------------------------------

 1 e80 REM ********************************
 2 H7  REM *                              *
 3 tN  REM *     H I N T E R H A L T      *
 4 pV  REM *     --------------------     *
 5 15  REM *      programmiert von        *
 6 FM  REM *      Christian Buchner       *
 7 Vc  REM *      Ganghoferstr. 2         *
 8 VO  REM *      8034 Germering          *
 9 OE  REM *                              *
10 8s  REM *             Viel Spass!      *
11 QG  REM *                              *
12 pJ  REM ********************************
13 Ed1  CLEAR ,20000
14 Pc   LIBRARY "intuition.library"
15 XY   SCREEN 1,320,255,4,1
16 fU   WINDOW 2,"",,16,1
17 Sb   RANDOMIZE TIMER
18 TP   DIM Feld%(1)
```

```
19 wk   Runde=0
20 JV   REM *** Palette festlegen ***
21 fX   FOR Farbe=0 TO 15
22 sz3    READ Rot,Gruen,Blau
23 P1     PALETTE Farbe,Rot/15,Gruen/15,Blau/15
24 8o1  NEXT Farbe
25 gD   DATA 00,00,00, 15,15,15, 00,00,00, 00,00,00
26 GK   DATA 04,04,04, 06,06,06, 08,08,08, 00,00,00
27 WA   DATA 00,00,00, 00,00,00, 00,00,00, 03,03,09
28 Do   DATA 14,12,04, 10,09,03, 15,08,00, 10,04,15
29 zy   REM *** Titelbild ***
30 Zm   WINDOW 1," ",,16,1
31 D4   FOR Sterne=1 TO 300
32 G43    PSET (320*RND,255*RND),3*RND+4
33 Aw1  NEXT Sterne
34 2k   Scr$="E L E C T R O N - S O F T  präsentiert:"
35 9k   CALL Titelscroll (Scr$)
36 pc   A$(1)="
              "
37 rJ   A$(2)="  +   +  +  +   + +++++ ++++ +++  +   +  +++  +   +
        ++++  "
38 jq   A$(3)="  +   +  +  ++  +   +   +    + +  +   + +   + +
         +    "
39 h4   A$(4)="  +++++  +  + + +   +   +++  +++  +++++ +++++ +
         +    "
40 68   A$(5)="  +   +  +  +  ++  +   +    + +  +   + +   + +
         +    "
41 HY   A$(6)="  +   +  +  +   +   +   ++++ +  + +   + +   + ++++
         +    "
42 PI   A$(7)="
              "
43 FF   FOR Spalte=1 TO LEN(A$(1))
44 Wx3    FOR Zeile=1 TO 7
45 h75      IF MID$(A$(Zeile),Spalte,1)="+" THEN Farbe=2 ELSE Farb
            e=3
46 iq       LINE (5*Spalte+8,10*Zeile+20)-(5*Spalte+11,10*Zeile+28
            ),Farbe,bf
47 Ze3    NEXT Zeile
48 3D1  NEXT Spalte
49 OQ   RESTORE Musikdaten
50 D9   PLAYMUSIC
51 bN0 Musikdaten:
52 Mm1  DATA 0,8, 4,8, 7,6, 12,8 ,7,8, 4,6, 5,8, 2,8, 7,6, 0,16, -
        1,-1
53 yk   REM *** Windowtitelscrollroutine ***
54 pE   SUB Titelscroll (Scr$) STATIC
55 64   Scr$=Scr$+CHR$(0)
56 zU   FOR Zeichen=1 TO LEN(Scr$)
57 X13    A$=STRING$(40-Zeichen,32)+LEFT$(Scr$,Zeichen)+CHR$(0)
58 CZ     CALL SetWindowTitles(WINDOW(7),SADD(A$),-1)
59 1c     FOR Warten=1 TO 100:NEXT Warten
60 k9     SOUND ASC(MID$(Scr$,Zeichen,1))*4,.1
61 MT1  NEXT Zeichen
62 24   END SUB
63 Lm   REM *** Subroutine PlayMusic ***
64 ge   SUB PLAYMUSIC STATIC
65 iI   SOUND WAIT
66 yA   Musik:
67 Oi   READ Grundton,Dauer
68 hi   IF Grundton<>-1 THEN
69 wu3    SOUND 2**(Grundton/12)*250,Dauer,127,0
70 aP     SOUND 2**((Grundton+4)/12)*250,Dauer,127,1
71 vZ     SOUND 2**((Grundton+7)/12)*250,Dauer,127,2
72 6Y     SOUND 2**((Grundton+12)/12)*250,Dauer,127,3
73 0z     SOUND RESUME
74 wh1    GOTO Musik
75 92     END IF
76 GI   END SUB
77 wg   REM *** Farb-Effekte ***
78 k1   FOR Warten=1 TO 2000:NEXT Warten
79 20   FOR Farbe=0 TO 1 STEP .01
80 wL3    PALETTE 2,0,Farbe,Farbe/2
81 f8     PALETTE 3,Farbe,0,Farbe/2
82 4k1  NEXT Farbe
83 q1   FOR Durchlauf=1 TO 3
84 753    FOR Farbe=0 TO 1 STEP .01
85 KM5      PALETTE 2,Farbe,1-Farbe,ABS(.5-Farbe)
86 eI       PALETTE 3,1-Farbe,Farbe,ABS(Farbe-.5)
87 9p3    NEXT Farbe
88 oz     FOR Farbe=1 TO 0 STEP-.01
89 OQ5      PALETTE 2,Farbe,1-Farbe,ABS(.5-Farbe)
90 iM       PALETTE 3,1-Farbe,Farbe,ABS(Farbe-.5)
91 Dt3    NEXT Farbe
92 En1  NEXT Durchlauf
93 10   LOCATE 16,4
94 Zx   COLOR 7,0
95 zs   PRINT "Geschrieben von Christian Buchner"
96 pA   FOR Farbe=0 TO 1 STEP .003
97 2h3    PALETTE 7,Farbe,Farbe,Farbe
98 KO1  NEXT Farbe
99 uq   LOCATE 23,3
100 17  COLOR 8,0
101 zG  PRINT "Wieviele Spieler wollen teilnehmen?"
102 5L  LOCATE 27,12
103 oE  COLOR 9,0
104 iA  PRINT "2       3       4"
105 WV  FOR Spieler=0 TO 2
106 jB3   CIRCLE (92+64*Spieler,211),12,10
107 2J    CIRCLE (92+64*Spieler,211),10,8
108 AB1 NEXT Spieler
109 Cc  FOR Farbe=0 TO 1 STEP .008
110 E23   PALETTE 8,Farbe,0,0
111 hT    PALETTE 9,0,Farbe,0
112 Fu    PALETTE 10,0,0,Farbe
113 ZF1 NEXT Farbe
114 CB  REM *** Auswahl der Spieleranzahl ***
115 V4  DEF FNXPos=PEEKW(PEEKL(WINDOW(7)+46)+18)
116 Px  DEF FNYPos=PEEKW(PEEKL(WINDOW(7)+46)+16)
117 Wu0 Ruecksprung:
118 7k1 REM GOSUB Anfeuerung
119 XP  WHILE MOUSE(0)<>0:WEND
120 cP0 AnzahlSpieler:
121 Sx1 IF MOUSE(0)=0 THEN GOTO AnzahlSpieler
122 Cn  IF FNYPos<212 OR FNYPos>232 THEN GOTO AnzahlSpieler
123 CZ  IF FNXPos<84 OR FNXPos>236 THEN GOTO AnzahlSpieler
124 zQ  Spieler=INT((FNXPos-84)/64)
125 H4  IF FNXPos>108+64*Spieler THEN GOTO AnzahlSpieler
126 2E  Spieleranzahl=Spieler+2
127 IU  CIRCLE (92+64*Spieler,211),12,1
128 oB  SOUND 500,2:SOUND 1000,2
129 Wm  FOR Warten=1 TO 1000:NEXT Warten
130 7Z  CIRCLE (92+64*Spieler,211),12,10
131 uW  REM *** Eingabe der Spielernamen ***
132 lz  WINDOW 3," Bitte geben Sie jetzt Ihre Namen ein!",(0,168)-
        (311,241),0,1
133 ky  WINDOW OUTPUT 3
134 Lg  PAINT (9,9),10
135 Cu  FOR Spieler=1 TO Spieleranzahl
136 ar3   COLOR 1,10
137 ZC    PRINT
138 fc    PRINT " Spieler Nr.";Spieler;":";
139 Lk    COLOR 8,0
140 9U    PRINT TAB(20);"                    ";
141 uS0 Eingabeschleife:
142 EU3   Spielername$(Spieler)=""
143 xK    LOCATE CSRLIN,20
144 6L0 Cursor:
145 EX3   PRINT CHR$(127);
146 IL    LOCATE CSRLIN,POS(0)-1
147 q00 Taste:
148 Hb3   Taste$=INKEY$:IF Taste$="" THEN GOTO Taste
149 BO    IF Taste$=CHR$(13) AND Spielername$(Spieler)<>"" THEN
          GOTO Endname
150 kj    IF Taste$=CHR$(8) AND Spielername$(Spieler)<>"" THEN G
          OTO Backspace
151 f0    IF ASC(Taste$)<32 OR ASC(Taste$)>127 AND ASC(Taste$)<
          160 THEN GOTO Taste
152 r6    IF LEN(Spielername$(Spieler))<18 THEN
153 4t5     Spielername$(Spieler)=Spielername$(Spieler)+Taste$:PRI
            NT Taste$;
154 8g      SOUND 800,.5
155 RK3   END IF
156 iW    GOTO Cursor
157 Jh1 Backspace:
158 mj3   PRINT " ";
159 Yc    LOCATE CSRLIN,POS(0)-2
```

Listing 1. »Hinterhalt« ist ein spannendes Spiel für zwei bis vier Fallensteller

```
160 2W      SOUND 400,.5
161 cK      Spielername$(Spieler)=LEFT$(Spielername$(Spieler),LEN(Sp
            ielername$(Spieler))-1)
162 oc      GOTO Cursor
163 4r0  Endname:
164 Gx3     PRINT " "
165 561   NEXT Spieler
166 iu    REM *** Aufbau von Spielfeld im Hintergrund ***
167 Sk    COLOR 1,0
168 GM    CLS
169 rZ    Scr$="Einen kleinen Moment, bitte...     "
170 Kv    CALL Titelscroll (Scr$)
171 GY    WINDOW OUTPUT 2
172 Xp    COLOR 1,0
173 LR    CLS
174 Ao    Feld%(0)=&H5555
175 iB    Feld%(1)=&HAAAA
176 UY    PATTERN &HFFFF,Feld%
177 7T    PAINT (9,9),11
178 AX    Feld%(0)=&HFFFF
179 Ec    Feld%(1)=&HFFFF
180 Yc    PATTERN &HFFFF,Feld%
181 46    Pi=3.1415
182 Om    LINE (128,76)-(128,236),13
183 xj    LINE (308,76)-(308,236),13
184 Te    LINE (138,66)-(298,66),13
185 rW    LINE (138,246)-(298,246),13
186 rY    CIRCLE (138,76),10,13,.5*Pi,1*Pi,1
187 10    CIRCLE (298,76),10,13,0*Pi,.5*Pi,1
188 zF    CIRCLE (138,236),10,13,1*Pi,1.5*Pi,1
189 gQ    CIRCLE (298,236),10,13,1.5*Pi,2*Pi,1
190 1J    LINE (138,76)-(298,236),12,bf
191 Pa    LINE (137,75)-(299,237),13,b
192 Sd    LINE (148,86)-(288,226),13,b
193 vh    PAINT (200,70),13
194 Od    FOR Spalte=0 TO 6
195 tI3     FOR Zeile=0 TO 6
196 XJ5       LINE (149+20*Spalte,87+20*Zeile)-(149+20*Spalte+18,87+
            20*Zeile+18),13,b
197 z43     NEXT Zeile
198 Td1   NEXT Spalte
199 ud    FOR Position=0 TO 6
200 gY3     LINE (150+20*Position,16)-(166+20*Position,65),0,b
201 mP      LINE (78,88+20*Position)-(127,104+20*Position),0,b
202 IE1   NEXT Position
203 UH    REM *** Festlegung der Löcher in den Schiebern ***
204 cl    Orange$(0)="* **  *":Weiss$(0)="**  ** **"
205 rY    Orange$(1)="*  **  ":Weiss$(1)=" ** *  * "
206 A1    Orange$(2)=" **  **":Weiss$(2)="*  ** * *"
207 VO    Orange$(3)="** ** *":Weiss$(3)="**  *  * "
208 MH    Orange$(4)="* * ** ":Weiss$(4)=" * * ** *"
209 rq    Orange$(5)=" **  **":Weiss$(5)=" **  *  *"
210 CN    Orange$(6)=" * ** *":Weiss$(6)="* * ** * "
211 al    Orange$(7)="*  * * ":
212 OC    Orange$(8)=" *  ** ":
213 Wd    REM *** Farben der Spieler festlegen ***
214 RV    Farbe(1)=8
215 bc    Farbe(2)=9
216 PD    Farbe(3)=10
217 dX    Farbe(4)=15
218 vP    REM *** Ausdruck der Schieberreihen ***
219 Ex    FOR Position=0 TO 6
220 3i3     SchieberX(Position)=INT(3*RND)
221 9p      SchieberY(Position)=INT(3*RND)
222 gs      DruckSchieberX(Position)
223 mz      DruckSchieberY(Position)
224 ea1   NEXT Position
225 Nm    FOR Zeile=0 TO 6
226 u93     FOR Spalte=0 TO 6
227 Ay5       CALL DruckLoch(Zeile,Spalte)
228 x73     NEXT Spalte
229 Va1   NEXT Zeile
230 wM    REM *** Spielfeld-Window in den Vordergrund holen ***
231 qr    WINDOW 2
232 1i    WINDOW CLOSE 3
233 ME    REM *** Kugeln auf die Spielfelder legen ***
234 B1    Kugelanzahl=10-Spieleranzahl
235 tn    CLEARWINDOW
236 K2    LINE (10,21)-(134,42),0,bf
237 vK    COLOR 8,0
238 Gl    PIXLOC 13,30:PRINT "Bitte plazieren"
239 xO    PIXLOC 13,38:PRINT "Sie Ihre Kugel."
240 MQ    FOR Kugel=1 TO Kugelanzahl
241 uc3     FOR Spieler=1 TO Spieleranzahl
242 Gx5       LOCATE 1,1:COLOR Farbe(Spieler),1
243 XC        PRINT "                    ":LOCATE 1,1
244 GX        PRINT Spielername$(Spieler)
245 1U        AX=-1:AY=-1:Del=0
246 jx        GOSUB Kugelsetzen
247 PQ3     NEXT Spieler
248 5a1   NEXT Kugel
249 IU    GOTO Hauptprogramm
250 8T    REM *** Hier werden die Kugeln gesetzt ***
251 C60  Kugelsetzen:
252 kF1   X=FNXPos:Y=FNYPos
253 jj    XPos=INT((X-155)/20)
254 c4    YPos=INT((Y-91)/20)
255 cH    IF X-20*XPos>169 OR Y-20*YPos>105 THEN XPos=-1:YPos=-1
256 1Q    IF Del=1 AND MOUSE(0)<>0 THEN GOTO DruckKugel
257 J0    IF AX=XPos AND AY=YPos AND Del=1 THEN GOTO Kugelsetzen
258 79    IF Del=1 THEN LINE (149+20*AX,87+20*AY)-(167+20*AX,105+20*
          AY),13,b
259 RA    IF XPos>=0 AND YPos>=0 AND XPos<7 AND YPos<7 THEN
260 123     Farbe=POINT (158+20*XPos,96+20*YPos)
261 sQ5       IF Farbe=1 OR Farbe=14 THEN
262 Xw        LINE (149+20*XPos,87+20*YPos)-(167+20*XPos,105+20*YPos
            ),Farbe(Spieler),b
263 qG        FOR Frq=400 TO 1000 STEP 100:SOUND Frq,.2:NEXT Frq
264 2L        AX=XPos:AY=YPos:Del=1:GOTO Kugelsetzen
265 D63     END IF
266 E71   END IF
267 bx    Del=0
268 tM    GOTO Kugelsetzen
269 So0  DruckKugel:
270 IR1   FOR Ton=1 TO 20:SOUND 100+900*RND,.2:NEXT Ton
271 i3    CIRCLE (158+20*AX,96+20*AY),6,Farbe(Spieler),,,1
272 Qb    PAINT (158+20*AX,96+20*AY),Farbe(Spieler)
273 cT    LINE (149+20*AX,87+20*AY)-(167+20*AX,105+20*AY),13,b
274 m0    RETURN
275 VG    REM *** Hier folgen einige SUBROUTINEN fuer die grafische
          Ausgabe ***
276 N6    REM *** Ausgaberoutine fuer obere Schieber ***
277 ZD    SUB DruckSchieberX (Position) STATIC
278 rT    SHARED SchieberX()
279 ux    XWert=151+20*Position
280 03    YWert=17+20*(2-SchieberX(Position))
281 VA    LINE (XWert,17)-(XWert+14,YWert),11,bf
282 du    LINE (XWert,YWert)-(XWert+14,65),14,bf
283 C1    LINE (XWert+1,YWert+1)-(XWert+13,YWert+7),13,bf
284 ce    END SUB
285 uA    REM *** Ausgaberoutine fuer linke Schieber ***
286 mR    SUB DruckSchieberY (Position) STATIC
287 2f    SHARED SchieberY()
288 bm    XWert=79+20*(2-SchieberY(Position))
289 FP    YWert=89+20*Position
290 bq    LINE (79,YWert)-(XWert,YWert+14),11,bf
291 hD    LINE (XWert,YWert)-(127,YWert+14),1,bf
292 vM    LINE (XWert+1,YWert+1)-(XWert+7,YWert+13),6,bf
293 ln    END SUB
294 19    REM *** Ausgaberoutine fuer Löcher ***
295 g2    SUB DruckLoch (XPos,YPos) STATIC
296 Qy    SHARED SchieberX(),SchieberY(),Orange$(),Weiss$()
297 sx    Orange$=MID$(Orange$(YPos+SchieberX(XPos)),XPos+1,1)
298 5H    Weiss$=MID$(Weiss$(YPos),1+XPos+SchieberY(YPos),1)
299 yh    IF Weiss$="*" THEN Farbe=1 ELSE Farbe=14
300 j0    IF Orange$=" " AND Weiss$=" " THEN Farbe=0
301 lr    Kugel=POINT (158+20*XPos,96+20*YPos)
302 7u    IF Kugel=0 OR Kugel=1 OR Kugel=14 OR Kugel=12 THEN
303 Sx3     LINE (151+20*XPos,89+20*YPos)-(165+20*XPos,103+20*YPos),
            Farbe,bf
304 VE1   ELSE
305 2U3     IF POINT(151+20*XPos,89+20*YPos)<>Farbe THEN
306 to5       CIRCLE (158+20*XPos,96+20*YPos),6,12,,,1
307 AT        PAINT (151+20*XPos,89+20*YPos),Farbe,12
308 s5        CIRCLE (158+20*XPos,96+20*YPos),6,Kugel,,,1
309 Dx        IF Farbe=0 THEN
310 uR7         SHARED Farbe(),AnzahlKugeln(),Spieleranzahl
311 BC          Faktor=40+80*RND
312 j1          FOR Versenk=7 TO 0 STEP-1
313 Cs9           FOR Frq=Versenk TO Versenk-1 STEP -.1
314 stB             SOUND 200+Faktor*Frq,.4
315 au9           NEXT Frq
```

```
316 CT          CIRCLE (158+20*XPos,96+20*YPos),Versenk,0,,,1
317 5p7       NEXT Versenk
318 Lq        FOR Durchlauf=12 TO 1 STEP -1
319 Ha9         SOUND 2**(Durchlauf/12)*440,.4
320 pM          SOUND 440,.4
321 vU7       NEXT Durchlauf
322 fw        LINE (151+20*XPos,89+20*YPos)-(165+20*XPos,103+20*YP
              os),0,bf
323 ck        FOR Spl=1 TO Spieleranzahl
324 PH9         IF Kugel=Farbe(Spl) THEN AnzahlKugeln(Spl)=AnzahlK
                ugeln(Spl)-1
325 kA7       NEXT Spl
326 C55     END IF
327 D63   END IF
328 E71 END IF
329 LN  END SUB
330 8b  REM *** Subroutine PIXLOC ***
331 vd  SUB PIXLOC (X%,Y%) STATIC
332 Vp  POKEW WINDOW(8)+36,X%
333 k2  POKEW WINDOW(8)+38,Y%
334 QS  END SUB
335 aP  REM *** Subroutine CLEARWINDOW ***
336 Qk  SUB CLEARWINDOW STATIC
337 yI  LINE (0,0)-(144,60),1,b
338 Me  LINE (1,1)-(143,59),10,bf
339 VX  END SUB
340 SO  REM *** Hier beginnt das eigentliche Spiel ***
341 Cp0 Hauptprogramm:
342 XF1 FOR Spieler=1 TO Spieleranzahl
343 Ht3   AnzahlKugeln(Spieler)=Kugelanzahl
344 yz1 NEXT Spieler
345 lS  RESTORE Musik2
346 zv  PLAYMUSIC
347 Vz0 Musik2:
348 Fg1 DATA 7,8, 4,8, 0,6, 5,8, 2,8, 7,6, 0,16, -1,-1
349 Q5  CALL CLEARWINDOW:COLOR 1,1
350 F1  LOCATE 1,1:PRINT "                    "
351 GU  PIXLOC 18,34:COLOR 8,0
352 S0  PRINT " LOS GEHTS! "
353 AW  FOR Warten=1 TO 10000:NEXT Warten
354 lj  LetzterZug$=""
355 t10 Hauptschleife:
356 lT1 FOR Spieler=1 TO Spieleranzahl
357 rl3   CLEARWINDOW
358 IO    LINE (10,21)-(134,42),0,bf
359 9q    LOCATE 1,1:COLOR Farbe(Spieler),1
360 Q5    PRINT "                  ":LOCATE 1,1
361 9Q    PRINT Spielername$(Spieler)
362 wL    COLOR 8,0
363 kS    IF AnzahlKugeln(Spieler)=0 THEN
364 SA5     LINE (14,21)-(130,42),0,bf
365 21      PIXLOC 17,30:PRINT " Leider schon "
366 av      PIXLOC 17,38:PRINT "ausgeschieden!"
367 Pg      FOR Warten=1 TO 2000:NEXT Warten
368 pf      GOTO NaechsterSpieler
369 tm3   END IF
370 ya    LINE (10,17)-(134,46),0,bf
371 tp    PIXLOC 13,26:PRINT "Ihr Zug, bitte!"
372 OF    PIXLOC 13,34:PRINT "Sie haben noch "
373 em    Menge$="":IF AnzahlKugeln(Spieler)>1 THEN Menge$="n"
374 GX    PIXLOC 13,42:PRINT "  ";AnzahlKugeln(Spieler);"Kugel"+Me
          nge$+"."
375 HY    FOR Durchlauf=1 TO 600 STEP 25
376 oP5     SOUND 600+Durchlauf*RND,.4
377 pO3   NEXT Durchlauf
378 o9    DelX=0:DelY=0:AX1=-1:AX2=-1:AY1=-1:AY2=-1
379 Yo    FOR Warten=1 TO 1000:NEXT Warten
380 4q1 REM *** Abfrage der Schieber ***
381 it0 SchieberAbfrage:
382 qL1 X=FNXPos:Y=FNYPos
383 B4  XPos=INT((X-155)/20):YPos=INT((Y-13)/20)
384 1O  IF X-20*XPos>169 OR XPos>6 OR YPos>2 OR YPos<0 THEN XP
        os=-1:YPos=-1
385 aC  IF DelX=1 AND MOUSE(0)<>0 THEN GOTO VerstellX
386 dM  IF AX1=XPos AND AY1=YPos AND DelX=1 THEN GOTO AndereSchieb
        er
387 6C  IF DelX=1 THEN
388 6S3   XWert=151+20*AX1
389 Dw    YWert=17+20*AY1
390 Kj    Farbe=POINT(XWert+2,YWert+2)
391 pS    LINE (XWert+1,YWert+1)-(XWert+13,YWert+7),Farbe,b
392 Zm    DelX=0
393 HA1 END IF
394 ak  IF XPos>=0 AND YPos>=0 THEN
395 3s3   IF 2-SchieberX(XPos)<>YPos THEN
396 tQ5     IF ABS((2-SchieberX(XPos))-YPos)<2 THEN
397 zL7       IF LetzterZug$<>"X"+CHR$(XPos) THEN
398 Mw9         XWert=151+20*XPos
399 sB          YWert=17+20*YPos
400 TN          LINE (XWert+1,YWert+1)-(XWert+13,YWert+7),Farbe(Sp
                ieler),b
401 kk          FOR Frq=-2 TO 2 STEP .3
402 csB           SOUND 800+400*SIN(Frq),.3
403 OK9         NEXT Frq
404 jM          DelX=1:AX1=XPos:AY1=YPos
405 TM7       END IF
406 UN5     END IF
407 VO3   END IF
408 WP1 END IF
409 DK0 AndereSchieber:
410 Si1 XPos=INT((X-77)/20):YPos=INT((Y-91)/20)
411 sC  IF Y-20*YPos>105 OR YPos>6 OR XPos>2 OR XPos<0 THEN XP
        os=-1:YPos=-1
412 Aj  IF DelY=1 AND MOUSE(0)<>0 THEN GOTO VerstellY
413 qF  IF AX2=XPos AND AY2=YPos AND DelY=1 THEN GOTO SchieberAbfr
        age
414 ah  IF DelY=1 THEN
415 2n3   XWert=79+20*AX2
416 D1    YWert=89+20*AY2
417 lA    Farbe=POINT(XWert+2,YWert+2)
418 7w    LINE (XWert+1,YWert+1)-(XWert+7,YWert+13),Farbe,b
419 5J    DelY=0
420 ib1 END IF
421 1B  IF XPos>=0 AND YPos>=0 THEN
422 XN3   IF 2-SchieberY(YPos)<>XPos THEN
423 Nv5     IF ABS((2-SchieberY(YPos))-XPos)<2 THEN
424 Wu7       IF LetzterZug$<>"Y"+CHR$(YPos) THEN
425 c19         XWert=79+20*XPos
426 mE          YWert=89+20*YPos
427 lr          LINE (XWert+1,YWert+1)-(XWert+7,YWert+13),Farbe(Sp
                ieler),b
428 BB          FOR Frq=-2 TO 2 STEP .3
429 3JB           SOUND 800+400*SIN(Frq),.3
430 R19         NEXT Frq
431 Q1          DelY=1:AX2=XPos:AY2=YPos
432 un7       END IF
433 vo5     END IF
434 wp3   END IF
435 xq1 END IF
436 RB  GOTO SchieberAbfrage
437 560 VerstellX:
438 C11 SchieberX(AX1)=2-AY1
439 Dm  CALL DruckSchieberX(AX1)
440 Us  GOSUB Verstellsound
441 Yo  FOR Warten=1 TO 1000:NEXT Warten
442 fA  FOR YPos=0 TO 6
443 Uz3   CALL DruckLoch(AX1,YPos)
444 Ca1 NEXT YPos
445 cs  FOR Warten=1 TO 1000:NEXT Warten
446 YS  LetzterZug$="X"+CHR$(AX1)
447 6w  GOTO NaechsterSpieler
448 LN0 VerstellY:
449 a41 SchieberY(AY2)=2-AX2
450 aC  CALL DruckSchieberY(AY2)
451 f3  GOSUB Verstellsound
452 jz  FOR Warten=1 TO 1000:NEXT Warten
453 kJ  FOR XPos=0 TO 6
454 en3   CALL DruckLoch(XPos,AY2)
455 Li1 NEXT XPos
456 n3  FOR Warten=1 TO 1000:NEXT Warten
457 xu  LetzterZug$="Y"+CHR$(AY2)
458 H7  GOTO NaechsterSpieler
459 kv0 Verstellsound:
460 T81 FOR Durchlauf=1 TO 50 STEP 3
461 Js3   SOUND (Durchlauf+8)**2,.5
462 C11 NEXT Durchlauf
463 pR  RETURN
```

Listing 1. »Hinterhalt« (Fortsetzung)

```
464 yT   REM *** Test, ob schon Spielende ***
465 yF0 NaechsterSpieler:
466 mn1  Kugelbesitzer=0
467 w4   FOR Spl=1 TO Spieleranzahl
468 R13    IF AnzahlKugeln(Spl)<>0 THEN Kugelbesitzer=Kugelbesitz
           er+1
469 4U1  NEXT Spl
470 rY   IF Kugelbesitzer=0 THEN GOTO Unentschieden
471 13   IF Kugelbesitzer=1 THEN GOTO Siegerehrung
472 ma   REM *** Naechster Spieler am Zug ***
473 34   NEXT Spieler
474 O5   GOTO Hauptschleife
475 0c   REM *** Spielende weil Unentschieden ***
476 p50 Unentschieden:
477 BS1  FOR Warten=1 TO 2000:NEXT Warten
478 57   RESTORE Melodie
479 84   PLAYMUSIC
480 Q1   FOR Warten=1 TO 6000:NEXT Warten
481 mR0 Melodie:
482 1B1  DATA 7,10, 6,10, 5,10, 4,15, 3.5,1, 3,1, 2.5,1
483 VD   DATA 2,1, 1.5,1, 1,1, .5,1, 0,10, -1,-1
484 PZ   GOSUB Pokal
485 Hb   CIRCLE (156,60),44,8,,,1
486 DW   CIRCLE (156,60),52,8,,,1
487 I5   PAINT (156,108),8
488 gC   LINE (120,91)-(188,23),8
489 oH   LINE (122,99)-(190,31),8
490 t8   PAINT (156,60),8
491 Pg   FOR Warten=1 TO 2000:NEXT Warten
492 eG   X=3:Y=16:Zeilen=6
493 b6   A$(0)="Kein Spieler hat gewonnen, weil mit"
494 mA   A$(1)="dem letzten Zug beide ihre letzte"
495 hv   A$(2)="Kugel verloren haben."
496 LK   A$(3)=""
497 uN   A$(4)="Noch einmal!     --> RETURN drücken"
498 XY   A$(5)=""
499 5o   A$(6)="Nein Danke!      -->     ESC drücken"
500 hP   GOSUB Textausgabe
501 by   FOR Warten=1 TO 20000:NEXT Warten
502 Vr   WINDOW OUTPUT 1
503 AA   WINDOW 1
504 cm   Runde=Runde+1
505 Yo   WHILE Antwort$<>CHR$(27) AND Antwort$<>CHR$(13)
506 I03    Antwort$=INKEY$
507 cQ1  WEND
508 Ls   IF Antwort$=CHR$(13) THEN Ruecksprung ELSE CleanUp
509 jS   REM *** Jemand hat gewonnen ***
510 MQ0 Siegerehrung:
511 Gy1  FOR Spieler=1 TO Spieleranzahl
512 vt3    IF AnzahlKugeln(Spieler)<>0 THEN Gewinnername$=Spieler
           name$(Spieler)
513 hi1  NEXT Spieler
514 m3   FOR Warten=1 TO 2000:NEXT Warten
515 Hf   RESTORE Siegermelodie
516 jf   PLAYMUSIC
517 Ff0 Siegermelodie:
518 oi1  DATA 0,3, 4,3, 7,3, 4,3, 2,3, 5,3, 9,3, 5,3, 4,3, 7,3, 11,
         3, 7,3
519 wm   DATA 12,6, 11,3, 9,3, 0,3, 4,3, 7,3, 4,3, 2,3, 5,3, 9,3, 5
         ,3
520 Xq   DATA 11,6, 9,3, 11,3, 12,12, -1,-1
521 OA   GOSUB Pokal
522 uB   FOR Warten=1 TO 2000:NEXT Warten
523 91   X=3:Y=16:Zeilen=6
524 Ti   IF Spieleranzahl=2 THEN Anzahl$="n" ELSE Anzahl$=""
525 sl   A$(0)="Ich gratuliere, "+Gewinnername$+"!"
526 fc   A$(1)=""
527 tY   A$(2)="Sie haben ihre"+Anzahl$+" Gegner in den"
528 kS   A$(3)="Hinterhalt geführt."
529 xx   A$(4)=""
530 ud   A$(5)="Noch ein Spiel?  --> RETURN drücken"
531 8o   A$(6)="Nein Danke !!!   -->     ESC drücken"
532 Dv   GOSUB Textausgabe
533 7U   FOR Warten=1 TO 20000:NEXT Warten
534 1N   WINDOW OUTPUT 1
535 gg   WINDOW 1
536 8I   Runde=Runde+1
537 4K   WHILE Antwort$<>CHR$(27) AND Antwort$<>CHR$(13)
538 oW3    Antwort$=INKEY$
539 8w1  WEND
540 rO   IF Antwort$=CHR$(13) THEN Ruecksprung ELSE CleanUp
```

```
541 L30 REM *** Diese Routine gibt den Pokal aus ***
542 OD  Pokal:
543 aI1  COLOR 1,0:CLS
544 fa   FOR Durchlauf=0 TO 8
545 qQ3    IF Durchlauf>3 AND Durchlauf<6 THEN Farbe=9 ELSE Farbe
           =8
546 ZO     LINE (0+Durchlauf,0+Durchlauf)-(311-Durchlauf,250-Durchl
           auf),Farbe,b
547 Z81  NEXT Durchlauf
548 DW   LINE (9,108)-(302,241),10,bf
549 nh   FOR Sterne=1 TO 150
550 VL3    PSET (320*RND,108*RND),3*RND+4
551 WI1  NEXT Sterne
552 ix   LINE (132,94)-(180,99),12,bf
553 3D   CIRCLE (136,79),15,12,1.5*Pi,.5*Pi,1
554 fS   CIRCLE (176,79),15,12,.5*Pi,1.5*Pi,1
555 UR   LINE (136,34)-(136,64),12
556 x2   LINE (176,34)-(176,64),12
557 0E   LINE (136,34)-(150,29),12
558 Sn   LINE (176,34)-(162,29),12
559 21   CIRCLE (156,24),8,12,,,1
560 j9   PAINT (156,24),12
561 wQ   PAINT (156,64),12
562 s7   CIRCLE (156,34),20,13,1*Pi,0*Pi,.1
563 vB   CIRCLE (156,62),20,13,1*Pi,0*Pi,.1
564 cn   FOR XPos=135 TO 177 STEP 6
565 QZ3    CIRCLE (XPos,97),2,8,,,1
566 Kh     PAINT (XPos,97),8
567 9W1  NEXT XPos
568 8P   COLOR 9,12
569 FB   PIXLOC 140,50
570 xA   PRINT "1988"
571 56   FOR Strahl=0 TO 360 STEP 36
572 It3    Bogen=Strahl*Pi/180
573 qL     X1=40*SIN(Bogen)+156:Y1=(45+10*RND)*COS(Bogen)+60
574 5e     X2=50*SIN(Bogen)+156:Y2=(55+10*RND)*COS(Bogen)+60
575 Rh     LINE (X1,Y1)-(X2,Y2),1
576 2F1  NEXT Strahl
577 fH   RETURN
578 Te0 REM *** Diese Routine ist fuer den Text zuständig ***
579 YG  Textausgabe:
580 k11  COLOR 1,10
581 ex   FOR Zeile=0 TO Zeilen
582 mD3    LOCATE Y+2*Zeile,X
583 uT     FOR Zeichen=1 TO LEN(A$(Zeile))
584 9X5      Zeichen$=MID$(A$(Zeile),Zeichen,1)
585 3R       PRINT Zeichen$;
586 pl       IF Zeichen$<>" " THEN SOUND 500,.3
587 X8       FOR Warten=1 TO 100:NEXT Warten
588 ry3    NEXT Zeichen
589 J01  NEXT Zeile
590 sU   RETURN
591 C0   REM *** Anfeuern der Computerfans ***
592 Np0 Anfeuerung:
593 3z1  IF Runde>9 THEN Runde=0
594 r1   IF Runde=0 THEN A$="Los gehts! Auf ins Vergnügen!"
595 Km   IF Runde=1 THEN A$="Na, wie wars? Noch ein Spielchen?"
596 IS   IF Runde=2 THEN A$="Und weils so schön war: Nochmal!"
597 47   IF Runde=3 THEN A$="Sie wollen doch noch nicht aufhören?"
598 ST   IF Runde=4 THEN A$="Jetzt wirds erst richtig spannend!"
599 UT   IF Runde=5 THEN A$="Das Spiel war toll! Noch eins?"
600 jD   IF Runde=6 THEN A$="Ach ist das aufregend!!! Nochmal!"
601 wi   IF Runde=7 THEN A$="Lassen wir die Kugeln tanzen!"
602 3b   IF Runde=8 THEN A$="So ists recht! Auf ein Neues!"
603 5r   IF Runde=9 THEN A$="So Leute, ich gebs noch eins aus!"
604 a0   A$=A$+CHR$(0)
605 AW   WINDOW OUTPUT 1
606 2P   CALL SetWindowTitles(WINDOW(7),SADD(A$),-1)
607 91   RETURN
608 7b0 CleanUp:
609 2M2  FOR i=1 TO 3
610 T24    WINDOW CLOSE i
611 IY2  NEXT i
612 ej   SCREEN CLOSE 1
613 X1   LIBRARY CLOSE
614 eZ   END
(C) 1988 M&T
```

Listing 1. »Hinterhalt« (Schluß)

Lawinengefahr

Eine tolle Spielidee, Computer-Gegner und einen Level-Editor bietet »Crazy Coins«. Ein Taktikspiel bringt viele vergnügliche Stunden. Bis zu vier Spieler nehmen teil.

Lawinen sind eigentlich eine sehr gefährliche Naturerscheinung. Bei diesem Taktikspiel allerdings ist das Auslösen einer Lawine sogar erwünscht. Nur mit Hilfe einer Münzlawine kommen Sie dem Sieg schnell näher.

Jeder der ein bis vier Teilnehmer erhält anfangs eine bestimmte Anzahl Münzen. Das Spielbrett ist in zehn senkrechte Bahnen aufgeteilt (Bild 1). Denken Sie sich das Spielfeld als Kasten, der am oberen Ende zehn Schlitze zum Einwerfen hat. Unten sind entsprechend zehn Auswürfe.

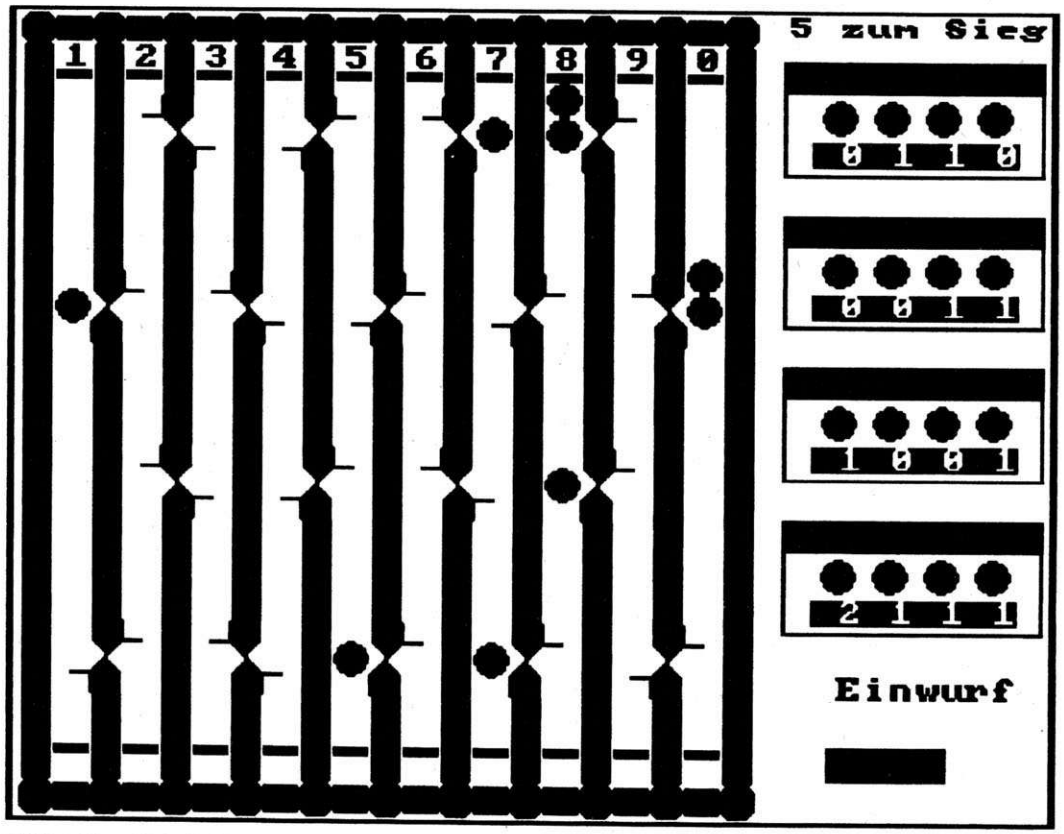


Bild 1. Das Spielfeld von »Crazy Coins«. Sammeln Sie Ihre Kugeln.

Vertrackte Schalter

Das Spielziel ist, alle Münzen seiner eigenen Farbe zu sammeln. Verloren hat auch der Spieler, der keine Münze mehr besitzt.

Nun wäre diese Aufgabe zu einfach, wenn das Spielfeld nicht einige Hindernisse enthielte. Auffälligste Elemente sind dabei die Schalter, die Münzen entweder stoppen oder hindurchlassen. Jede Bahn kann mehrere Schalter hintereinander enthalten. Die Schalter sind durch Doppelpfeile (>—<) dargestellt. Zeigt die Öffnung des Schalters nach oben, darf die Münze passieren. Zeigt die Öffnung nach unten, stoppt der Schalter alle Münzen.

Nun sind die Schalter genau zwischen den Bahnen angebracht. Eine Öffnung zeigt also immer nach oben. Werfen Sie eine Münze in eine Bahn, in der ein Schalter Durchgang signalisiert, der auf der Nachbarbahn eine Münze blockiert, passiert folgendes: Die eingeworfene Münze passiert den Schalter, kippt diesen dabei nach unten. Dadurch wird der Schalter auch auf der Bahn nebenan auf »Frei« umgestellt. Die vorher blockierte Münze wird freigegeben und fällt nach unten. Auf diese Weise sind natürlich Kettenreaktionen möglich. Ihre Aufgabe ist es nun, möglichst mit jeder eingeworfenen so viele Münzen wie möglich bis zum unteren Rand zu bringen. Fallen diese nämlich unten heraus, werden sie Ihrem Konto hinzugezählt.

Jeder Spieler besitzt eine Anzeige für die Münzen mit den vier verschiedenen Farben. Dies ist der persönliche Kontostand.

Haben Sie das Listing 1 mit dem Checksummer (Seite 159) eingegeben, steht einer Partie »Crazy Coins« nichts mehr im Wege. Laden Sie Amiga-Basic und anschließend das Programm. Das Basic-Programm benötigt sehr viel Speicherplatz. Sollten Sie einen Amiga 500 ohne Speichererweiterung besitzen, stoppen Sie bitte eventuell laufende Programme und schließen alle Windows, um genügend Speicher freizugeben.

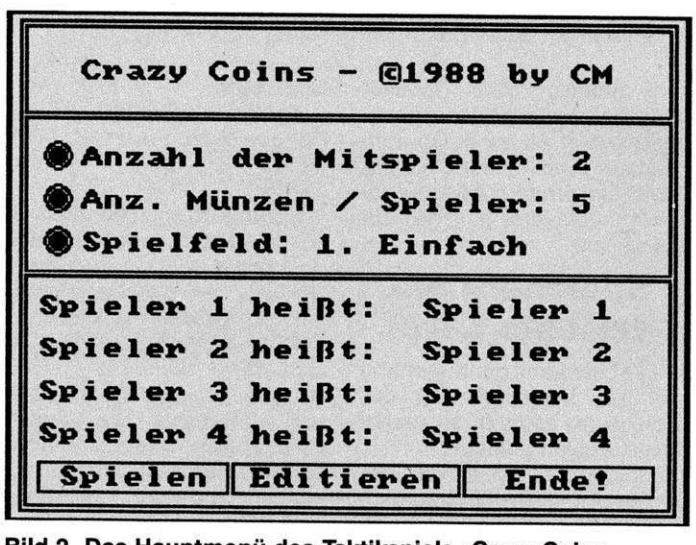


Bild 2. Das Hauptmenü des Taktikspiels »Crazy Coins«. Fordern Sie bis zu drei Computer-Gegner heraus.

Maximal drei Computergegner

Nach dem Programmstart mit RUN erscheint die Titelgrafik, die schon ein Leckerbissen für sich ist. Nach einem beliebigen Tastendruck wird das Hauptmenü geladen (Bild 2). Hier nehmen Sie alle Einstellungen vor. Folgende Parameter stehen zur Wahl:

Anzahl der Mitspieler

Wählen Sie mit der Maus eine Zahl zwischen 2 und 4. Bewegen Sie den Mauszeiger in die Zeile und drücken die linke Maustaste. Der angezeigte Wert verändert sich nun. Bis zu vier Gegner übernimmt bei Bedarf der Amiga. Geben Sie ein-

fach bei »Spieler« als Namen »Amiga« ein — schon spielt der Computer mit.

Anzahl Münzen/Spieler

Vor Spielbeginn legen Sie die Anzahl Münzen fest, die jeder Spieler von seiner Farbe besitzt. Die Auswahl erfolgt wieder mit der Maus.

Spielfeld

Hier bestimmen Sie die Form des Spielfelds. Das Programm stellt fünf Felder: einfach, mittel, schwer, komplex und verrückt. Zusätzlich haben Sie die Möglichkeit, im Editor (siehe unten) Spielfelder zu entwerfen. Diese werden dann auf Diskette gespeichert. Laden Sie das gespeicherte Spielfeld später, können Sie auf diesem Ihre Partien austragen.

Spielernamen

Geben Sie die Namen aller Mitspieler ein. Klicken Sie mit der linken Maustaste das entsprechende Namensfeld an. Geben Sie dann über die Tastatur den Namen ein und schließen mit <RETURN> ab.

Zusätzlich zu den genannten Punkten stehen in der Fußleiste des Menüfensters die Begriffe Spielen, Editieren und Ende. Wählen Sie für das erste Probespiel fünf Münzen und das einfache Brett.

Vorhang auf, das Spiel beginnt

Klicken Sie mit der linken Maustaste »Spielen« an, erscheint ein neues Fenster. In diesem baut das Programm das gewählte Spielbrett auf. Alle Schalter werden zufällig gesetzt und eine Anzahl von Münzen vorab zufällig eingeworfen. Das führt dazu, daß jedes Spiel auf einem gleichen Brett mit einer völlig verschiedenen Ausgangslage beginnt. Ebenfalls zufällig bestimmt das Programm den Startspieler. Nachdem alle »Vorläufer« plaziert sind, blinkt das Feld des Spielers, der beginnt. Dieser wählt nun mit dem Mauszeiger und Druck auf die linke Maustaste die Münze, die er einwerfen will. Dies sollte möglichst keine seiner eigenen Farbe sein, da es diese ja zu horten gilt. Ist die Münze ausgewählt, erscheint das entsprechende Symbol links neben dem Wort »Einwurf«. Anschließend ist die Schachtnummer anzuklicken, in die die Münze eingeworfen werden soll, und schon rutscht diese mehr oder weniger weit nach unten. Erreicht eine der Münzen den unteren Rand, wird diese zum Konto hinzugezählt.

Es wird so lange gespielt, bis einer der Spieler alle Münzen seiner Farbe besitzt, oder bis nur noch ein Spieler eine Münze auf seinem Konto hat. Abbrechen können Sie das Spiel mit Anklicken des Menü-Schalters rechts unten.

Komfortabler Editor

Wählen Sie im Hauptmenü den Punkt »Editieren«, erscheint das Fenster des Editors (Bild 3). Rechts oben steht der Name des bearbeiteten Spielbretts. Darunter sehen Sie eine Tafel mit den Grafiksymbolen aller Gestaltungselemente. Ein Leerfeld, Bahnbegrenzungen, Ecken für Umlenkung der Münzen sind dort auszuwählen. Als besonderer Gag erweisen sich die Elemente in den acht Feldern rechts unten, deren Farben denen der Münzen gleichen. Diese Symbole haben folgende Wirkung: Trifft eine Münze auf ein Symbol mit der schwarzen Öffnung nach oben, gelangt die Münze beim gleichfarbigen Symbol mit Öffnung nach unten wieder ins Spiel. Das ergibt trickreiche Varianten.

Die folgenden Punkte sollten Sie für das Erstellen eines Spielbrettes unbedingt beachten.

1. Unteres Ende des Feldes und Ränder: Hier sollten ausschließlich Mauern angebracht werden. Setzt man ein Zielfeld, erscheint automatisch darunter eine Mauer. Grundsätzlich darf eine Münze das Spielfeld nie verlassen.
2. Einwurffelder: können nicht verändert werden.
3. Schrägen und Schalter nicht am Rand anbringen.
4. Umleitungssteine mit der Öffnung nach oben können in beliebiger Anzahl erscheinen. Die Steine mit der Öffnung nach unten dürfen für jede Farbe nur einmal gesetzt werden.
5. Schalter: Zwei Schalter müssen waagerecht und senkrecht einen Abstand von mindestens zwei Feldern haben. Vom Start- und Endfeld muß jeweils mindestens ein Feld Abstand sein.

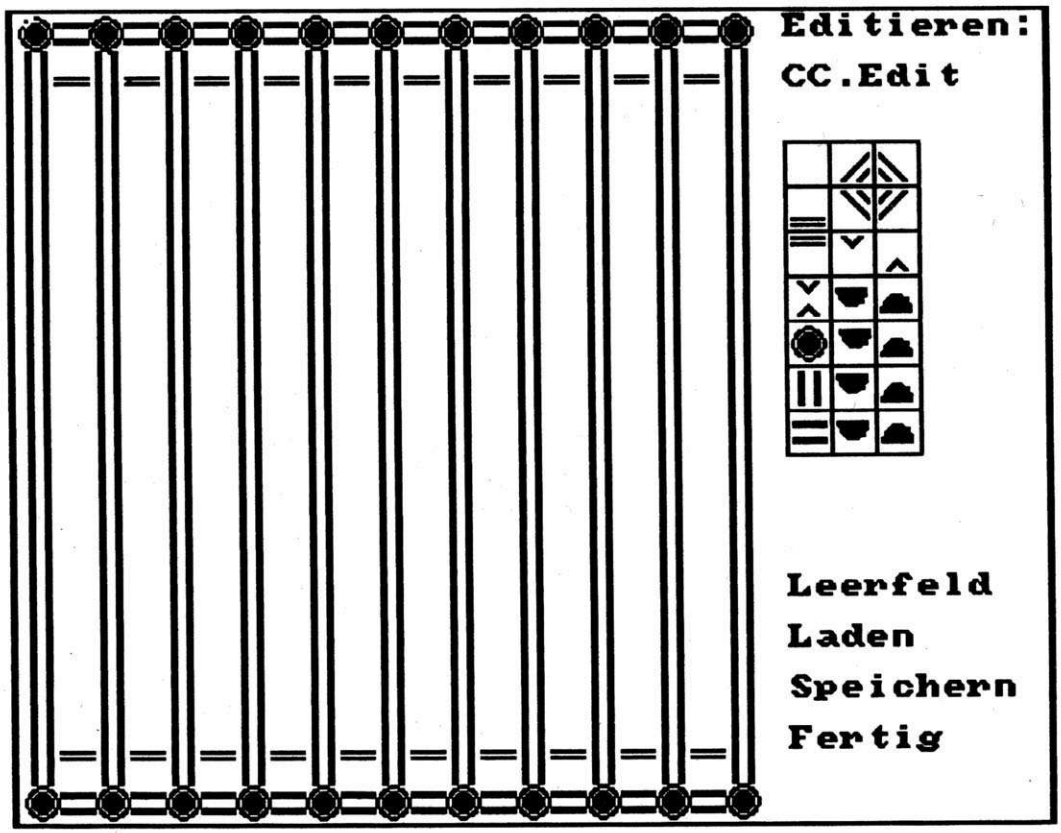


Bild 3. Im Editor erstellen Sie Spielfelder nach eigenem Geschmack. Alle Funktionen werden über die Maus gesteuert. Stellen Sie sicher, daß keine Kugel entwischt.

Speichern Sie erstellte Level

Die Felder mit der Kugel sind nur zur Verschönerung des Spielfeldes gedacht, sie haben keine aktive Funktion.

Links unten stehen vier Begriffe mit folgender Bedeutung:

Leerfeld

Erzeugt ein Feld zum Weiterbearbeiten.

Laden

Hier laden Sie auf Diskette gespeicherte Spielfelder. Klicken Sie das Feld an, erscheint oben in der Namenszeile der Strich-Cursor. Geben Sie über die Tastatur den Namen des zu ladenden Feldes ein und drücken <RETURN>. Findet das Programm das File mit dem eingegebenen Namen (»cc.« wird automatisch zur Kennung angefügt) nicht, erscheint die Meldung »—Error—«. Das Ausgabefenster ist allerdings dann verdeckt, drücken Sie <Amiga-links N> oder <Amiga-links M>, erscheint das Fenster des Editors wieder.

Speichern

Analog zur Funktion Laden, hier speichern Sie erstellte Felder.

Fertig

Sprung zum Hauptmenü. Sie können nun das erstellte Spielfeld testen, indem Sie das Spielfeld entsprechend wählen und »Spielen« anklicken.

Alle Bedienungselemente sind damit erklärt. Das Feld »Ende!« im Hauptmenü erklärt sich von selbst. Sie werden diese Funktion wahrscheinlich sowieso nur nach langen Sitzungen mit »Crazy Coins« wählen. Wir wünschen Ihnen jedenfalls viele Lawinen, die Ihr Konto blitzartig erhöhen. *(rs)*

Programmname:	Crazy Coins
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic
Bemerkung:	Beim A500 ohne Speichererweiterung bitte alle anderen Programme stoppen und alle Windows schließen

```
Programmautor: Christian Marz
-----------------------------
 1 3A0 '****************************************
 2 dF  '*  Crazy Coins - (w)1988 by Chr. Marz  *
 3 5C  '****************************************
 4 GU  'Achtung: Alle unbenutzten Fenster schließen!
 5 Jz  DEFINT a-o,s-z
 6 pu  DIM f(20,22),t(20,22),m(61,2),sch(60,1),s(4,4),g(4),mm(3)
 7 6f  DIM e(51,22),z(51,4),L(51),p(15,2),v(11),d(12),h(3,1)
 8 6m  DIM t$(2,4),n$(4),gw(5),f$(5),b(20),k(5,2),q(9),mn(3)
 9 KV  SCREEN 2,328,257,4,1: WINDOW 2,,,16,2
10 Re  FOR i=0 TO 15: READ p(i,0),p(i,1),p(i,2)
11 xX2   PALETTE i,p(0,0),p(0,1),p(0,2): NEXT
12 Ah0 FOR i=0 TO 11: READ v(i): NEXT
13 fv  FOR i=0 TO 12: READ d(i): NEXT
14 Xk  FOR i=0 TO 20: READ b(i): NEXT
15 fo  FOR i=0 TO 2: FOR j=0 TO 4: READ t$(i,j): NEXT j,i
16 3x  FOR i=0 TO 5: READ f$(i): gw(i)=0: NEXT
17 r5  FOR i=1 TO 4: n$(i)="Spieler "+CHR$(48+i)+" ": NEXT
18 Jm  FOR i=1 TO 10: MENU i,0,1," ": MENU i,0,0: NEXT
19 OJ  RANDOMIZE TIMER: spz=2: stf=1: spf=1
20 Y1  mn(1)=5: mn(2)=6: mn(3)=8: mm(1)=5: mm(2)=9: mm(3)=12
21 q0  MOUSE ON: GOSUB Get.Elemente: COLOR ,0: CLS
22 bN  FOR i=0 TO 15: PALETTE i,p(i,0),p(i,1),p(i,2): NEXT
23 Ln  GOSUB Title
24 dP  FOR i=0 TO 15: PALETTE i,p(i,0),p(i,1),p(i,2): NEXT
25 a0  GOSUB Init.Feld1: GOSUB Menue
26 Bg  Runde:
27 QN  GOSUB Init.Feld: COLOR ,0: CLS: GOSUB Feld.Anz
28 sA  spm=0: msp=spz: GOSUB Spieler.Verteilung
29 8r  GOSUB Spieler.Anz: GOSUB Punkt.Anz
30 OM  GOSUB Start.Einwurf
31 Pz  zzm=0: gwr=0: spl=INT(RND*spz)
32 Jj  WHILE msp>1 AND gwr=0 AND zzm=0
33 mt2   spl=spl+1: IF spl>spz THEN spl=1
34 qB    IF s(spl,0)>0 THEN W0
35 on    IF LEFT$(n$(spl),5)="Amiga" THEN GOSUB Amiga.Zug ELSE GOS
         UB Spieler.Zug
36 cr    IF zzm=1 THEN W0
37 Bv    s(spl,f)=s(spl,f)-1
38 4r    FOR i=0 TO 4: g(i)=0: NEXT: GOSUB Einwurf: a=0
39 zo    FOR i=1 TO 4: s(spl,i)=s(spl,i)+g(i): a=a+s(spl,i): NEXT
40 uu    GOSUB Punkt.Anz
41 wT    IF s(spl,spl)>=mn(stf) THEN gwr=spl
42 Ee    IF a=0 THEN s(spl,0)=1: msp=msp-1:
43 PL0 W0: WEND: IF zzm=1 THEN GOSUB Menue: GOTO Runde
44 Bf  IF gwr=0 THEN
45 Gb2   FOR i=1 TO spz: IF s(i,0)=0 THEN gwr=i
46 pu4     NEXT
47 H30 END IF: gw(gwr)=gw(gwr)+1
48 Xz  GOSUB sieger.anz: GOTO Runde
49 YZ  Ende: BEEP: WINDOW CLOSE 2: SCREEN CLOSE 2: END
50 WL  Einwurf:
51 4X  spm=spm+1: m(spm,0)=f: m(spm,1)=x: m(spm,2)=2
52 LY  PUT(x*11,22),z(0,f),PSET
53 X2  Ein0: mm=0: i=1
54 W9  Ein1:
55 gQ2   x=m(i,1): y=m(i,2): IF f(x,y)<>i+30 THEN f(x,y)=i+30: P
         UT(x*11,y*11),z(0,m(i,0)),PSET
56 s0    IF f(x,y+1)>9 THEN F2
57 yF    ON f(x,y+1) GOTO DLR,DL,DR,SC,ZL,LH,LH,LH,LH
58 aZ    IF f(x+1,y)=14 OR f(x-1,y)=14 THEN
59 w44     f(x,y)=15: f=3: v=y: h=x-1: IF f(x+1,y)=14 THEN h=x+1
60 28      PUT(x*11,y*11),L,PSET: PUT(x*11,y*11+11),z(0,m(i,0)),PS
           ET
61 e3      GOSUB SchM: SOUND 1100,1: f(x,y+1)=i+30: m(i,2)=y+1
62 kx      GOTO FF
63 xq2   END IF
64 rt    IF f(x,y-1)<>15 THEN F0
65 Vk    IF f(x-1,y-1)<>14 AND f(x+1,y-1)<>14 THEN F0
66 tI    IF f(x-1,y-1)=14 THEN
67 S54     f(x-2,y-1)=4: f(x,y-1)=0: f(x,y)=15
68 04      f=0: v=y-1: h=x-1: GOSUB SchM: f=3: GOSUB SchL
69 iR2   ELSE
70 Lw4     f(x+2,y-1)=4: f(x,y-1)=0: f(x,y)=15
71 BJ      f=0: v=y-1: h=x+1: GOSUB SchM: f=3: GOSUB SchR
72 Rc2   END IF: SOUND 1200,1: GOTO F1
73 rv    ZL:
74 xD    g(m(i,0))=g(m(i,0))+1: f(x,y)=0: PUT(x*11,y*11),e(0,5),PS
         ET
75 cK    FOR j=0 TO 2: SOUND 2600,1,255: SOUND 2100,1: NEXT
76 Wg    FOR j=i TO spm-1
77 D84     m(j,0)=m(j+1,0): m(j,1)=m(j+1,1): m(j,2)=m(j+1,2): NEXT
78 zd2   spm=spm-1: i=i-1: GOTO F2
79 J5    LH:
80 hk    a=f(x,y+1)-6: IF f(h(a,0),h(a,1)+1)>0 THEN F2
81 pu    f(x,y)=0: f(h(a,0),h(a,1)+1)=i+30
82 Ws    m(i,1)=h(a,0): m(i,2)=h(a,1)+1
83 bA    SOUND 400,1: SOUND 300,3,255
84 hL    PUT(x*11,y*11),L,PSET: PUT(h(a,0)*11,h(a,1)*11+11),z(0,m(
         i,0)),PSET
85 7K    GOTO FF
86 OW    DLR:
87 2c    IF f(x-1,y+1)<>0 AND f(x+1,y+1)<>0 THEN F2
88 00    IF (RND<.5 AND f(x-1,y+1)=0) OR f(x+1,y+1)<>0 THEN
89 ML4     f(x-1,y+1)=30+i: f(x,y)=0: m(i,1)=x-1: m(i,2)=y+1
90 eI      PUT(x*11,y*11),L,PSET: PUT(x*11-11,y*11+11),z(0,m(i,0))
           ,PSET
91 4n2   ELSE
92 B64     f(x+1,y+1)=30+i: f(x,y)=0: m(i,1)=x+1: m(i,2)=y+1
93 bD      PUT(x*11,y*11),L,PSET: PUT(x*11+11,y*11+11),z(0,m(i,0))
           ,PSET
94 iL2   END IF: SOUND 1900,1: GOTO FF
95 VD    DL:
96 Xq    IF f(x-1,y+1)>0 THEN F2
97 UT    f(x-1,y+1)=30+i: f(x,y)=0: m(i,1)=x-1: m(i,2)=y+1
98 mQ    PUT(x*11,y*11),L,PSET: PUT(x*11-11,y*11+11),z(0,m(i,0)),P
         SET
99 V1    SOUND 2000,1: GOTO FF
100 sg   DR:
101 Wn   IF f(x+1,y+1)>0 THEN F2
102 LG   f(x+1,y+1)=30+i: f(x,y)=0: m(i,1)=x+1: m(i,2)=y+1
103 lN   PUT(x*11,y*11),L,PSET: PUT(x*11+11,y*11+11),z(0,m(i,0)),P
         SET
104 a6   SOUND 2000,1: GOTO FF
105 iW   SC:
106 CV   IF f(x,y+2)>0 THEN F2
107 g2   j=x-2: IF f(x+1,y+1)=14 THEN j=x+2
108 FG   IF f(j,y+1)=0 THEN W1
109 wu   a=f(j,y+1)-30: m(a,2)=y: PUT(j*11,y*11+11),L,PSET
110 EP   IF f(j,y)=0 THEN PUT(j*11,y*11),z(0,m(a,0)),PSET: GOTO W1
111 KM   b=f(j,y)-30
112 kA   IF b<a THEN
113 4j4    SWAP m(a,0),m(b,0): SWAP m(a,1),m(b,1)
114 U3     PUT(j*11,y*11),z(0,m(a,0)),PSET
115 ng2  END IF
116 9r   W1: f(j,y+1)=15: f(j,y+2)=0: f=0: v=y+1: SOUND 1000,1
117 Ne   IF j=x-2 THEN h=x-1: GOSUB SchR ELSE h=x+1: GOSUB SchL
118 PV   F0: f(x,y)=0
119 Ug   F1: f(x,y+1)=i+30: m(i,2)=y+1
120 066      PUT(x*11,y*11),L,PSET: PUT(x*11,y*11+11),z(0,m(i,0)),
             PSET
121 pG2  FF: mm=mm+1
122 u80 F2: i=i+1: IF i<=spm THEN Ein1
123 Nf   IF mm>0 THEN Ein0
124 My   RETURN
125 ci   SchL:
126 VX   a=5+11*h: b=5+11*v: LINE(a-4,b-4)-(a+4,b+4),f
127 CA   LINE(a-11,b-5)-(a-5,b-5),f: LINE(a-5,b-5)-(a-5,b-11),f
128 57   LINE(a+11,b+5)-(a+5,b+5),f: LINE(a+5,b+5)-(a+5,b+11),f
129 R3   RETURN
130 BN   SchR:
131 ac   a=5+11*h: b=5+11*v: LINE(a+4,b-4)-(a-4,b+4),f
132 fp   LINE(a+11,b-5)-(a+5,b-5),f: LINE(a+5,b-5)-(a+5,b-11),f
133 mc   LINE(a-11,b+5)-(a-5,b+5),f: LINE(a-5,b+5)-(a-5,b+11),f
134 W8   RETURN
135 ry   SchM:
136 K5   a=5+11*h: b=5+11*v: LINE(a-4,b)-(a+4,b),f
137 rt   LINE(a-8,b-3)-(a-5,b),f: LINE(a-5,b)-(a-8,b+3),f
138 OS   LINE(a+8,b-3)-(a+5,b),f: LINE(a+5,b)-(a+8,b+3),f
139 bD   RETURN
140 hm   Init.Feld: ERASE f: DIM SHARED f(20,22)
141 Ug   IF spf=0 THEN FOR x=0 TO 20: FOR y=0 TO 22: f(x,y)=t(x,y):
         NEXT: NEXT: RETURN
```

Listing 1. »Crazy Coins« fordert Ihren Scharfsinn heraus. Bitte mit dem Checksummer (Seite 159) eingeben.

```
142 1s  FOR i=0 TO 20 STEP 2: f(i,0)=20: f(i,22)=20
143 O82   FOR j=1 TO 21: f(i,j)=21: NEXT j,i
144 Ik0 FOR i=1 TO 19 STEP 2
145 1n2   f(i,0)=22: f(i,22)=22: f(i,1)=19: f(i,21)=5: NEXT
146 Rq0 ON spf GOTO S1,S2,S3,S4,S5
147 2X  S1: TO 4,3,4: TO 2,8,5: TO 4,13,4: TO 2,18,5: RETURN
148 xj  S2: TO 2,3,5: TO 4,7,4: TO 2,11,5: TO 4,15,4: TO 2,19,5
149 Ry2   T1 2,10: T1 18,10: IF spf=2 THEN CALL T1(10,10)
150 m0    RETURN
151 xt0 S3: GOSUB S2: RESTORE Feld3
152 nz2   FOR i=6 TO 13: READ x,y: f(x,y)=i: NEXT: RETURN
153 ep0 S4: TO 2,7,5: TO 4,11,4: TO 2,15,5: f(10,15)=21
154 D62   RESTORE Feld4: FOR i=0 TO 3: READ x,y: f(x,y)=14: NEXT
155 Of    f(9,2)=13: f(11,2)=11: f(7,21)=7: f(13,21)=9
156 9a    T1 10,6: T1 6,15: T1 10,15: T1 14,15: RETURN
157 sa0 S5: FOR i=0 TO 4: T1 2+i*4,17: NEXT
158 nv2   FOR i=0 TO 3: f(1+i*6,3)=6+i: f(1+i*6,4)=10+i: f(1+i*6,21
          )=9-i
159 m8    NEXT: TO 4,7,4: TO 2,10,5: TO 4,13,4: RETURN
160 Gw0 SUB TO (x,y,a) STATIC
161 Lw2   FOR i=x TO x+a*4-4 STEP 4: f(i,y)=14: NEXT
162 eg0 END SUB
163 tS  SUB T1 (x,y) STATIC
164 uP2   f(x,y-1)=18: f(x-1,y)=3: f(x,y)=0: f(x+1,y)=2
165 Hd    f(x-1,y+1)=17: f(x,y+1)=0: f(x+1,y+1)=16: f(x,y+2)=1
166 ik0 END SUB
167 IK  Init.Feld1: ERASE t: DIM SHARED t(20,22)
168 rT  FOR x=0 TO 20 STEP 2: t(x,0)=20: t(x,22)=20
169 4p2   FOR y=1 TO 21: t(x,y)=21: NEXT y,x
170 Ae0 FOR x=1 TO 19 STEP 2
171 v82   t(x,0)=22: t(x,22)=22: t(x,1)=19: t(x,21)=5: NEXT: RETURN
172 RZ0 Feld.Anz: sn=0
173 qh  FOR y=22 TO 0 STEP -1: FOR x=0 TO 20
174 BW2   a=f(x,y): PUT(x*11,y*11),e(0,a),PSET
175 8s    IF a=5 THEN f(x,y+1)=5: f(x,y)=0
176 Vv    IF a=14 THEN sn=sn+1: sch(sn,0)=x: sch(sn,1)=y
177 aS    IF a>9 AND a<14 THEN h(a-10,0)=x: h(a-10,1)=y
178 240 NEXT x,y: POKEW WINDOW(8)+38,17: COLOR 5,0
179 1a  FOR i=1 TO 10: POKEW WINDOW(8)+36,INT(i)*22-9
180 Th2   PRINT USING"# ";i+(i=10)*10;
181 Jc0 NEXT: f=3
182 Ht  FOR i=1 TO sn
183 Qx2   IF RND<.5 THEN
184 6n4     h=sch(i,0): v=sch(i,1)
185 2v      f(h-1,v)=4: f(h+1,v+1)=15: GOSUB SchL
186 bK2   ELSE
187 9q4     h=sch(i,0): v=sch(i,1)
188 HG      f(h+1,v)=4: f(h-1,v+1)=15: GOSUB SchR
189 zs2   END IF
190 dP0 NEXT: RETURN
191 hE  Spieler.Anz:
192 a6  FOR i=0 TO spz-1
193 d62   COLOR 8+i*2,9+i*2
194 vI    LINE(239,15+i*48)-(319,50+i*48),9+i*2,bf
195 G5    LINE(240,25+i*48)-(318,49+i*48),0,bf
196 sU    LOCATE 3+i*6,31: PRINT n$(i+1)
197 Rg    FOR j=0 TO 3
198 QH4     PUT(251+j*16,27+i*48),z(0,j+1),PSET: NEXT
199 yU0 NEXT: COLOR 3,0: LOCATE 27,33: PRINT "Einwurf"
200 qd  LOCATE 1,30: PRINT mn(stf)"zum Sieg": RETURN
201 yx  Punkt.Anz:
202 Hb  FOR i=0 TO spz-1: FOR j=0 TO 3
203 z82   LOCATE 6+i*6,32+j*2: COLOR 0,j*2+9
204 KD    PRINT USING"# # ";s(i+1,j+1)
205 WC0 NEXT j,i: RETURN
206 1C  Spieler.Zug:
207 RA  ok1=0: ok2=0: j=7+spl*2: PUT(240,206),L,PSET
208 6D  LINE(253,230)-(289,240),5,b: LINE(254,231)-(288,239),6,bf
209 7g  COLOR 3,6: LOCATE 30,33: PRINT "Menü"
210 qt  Zug1:
211 t52   FOR q=0 TO 1.5 STEP .15
212 WB4     PALETTE j,p(j,0)*q,p(j,1)*q,p(j,2)*q
213 uf2   NEXT: PALETTE j,p(j,0),p(j,1),p(j,2)
214 J8    IF MOUSE(0)<0 THEN
215 vk4     a=MOUSE(1): b=MOUSE(2)
216 dj      IF a>250 AND a<315 AND b>spl*48-22 AND b<spl*48-10
            THEN f=1+(a-251)\16: ok1=1: PUT(240,206),z(0,f),PSET
217 YY      IF a>10 AND a<219 AND ((a\11) MOD 2)=1 AND b>10 AND
            b<22 THEN x=a\11: ok2=1
218 rP      IF s(spl,f)=0 AND ok1 THEN ok1=0: BEEP
219 Qr      IF ok2 THEN IF f(x,2)>0 THEN ok2=0: BEEP
220 Hs      IF a>253 AND a<289 AND b>230 AND b<240 THEN zzm=1
221 V02   END IF
222 QK0 IF (ok1<>1 OR ok2<>1) AND zzm=0 THEN Zug1
223 BA  LINE(253,230)-(289,240),0,bf: RETURN
224 71  Amiga.Zug:
225 Qq  LOCATE spl*6-3,31: COLOR 0,7+spl*2: PRINT n$(spl)
226 12  b=0: FOR i=1 TO 4: b=b+s(spl,i): NEXT: zzm=0
227 OB  K9: f=INT(RND*4)+1
228 HM  IF (spl=f AND b<>s(spl,f)) OR s(spl,f)=0 THEN K9
229 9Y  FOR i=0 TO 9
230 p52   q(i)=0: k(1,0)=f: k(1,1)=i*2+1: k(1,2)=2: mm=1: j=0: q=1
231 OQ    K0: j=j+1: IF j>5 OR j>mm THEN K8
232 4P4     x=k(j,1): y=k(j,2): qq=0
233 Hv2   K1: IF f(x,y)>9 THEN K0
234 DL4     ON f(x,y) GOTO K2,K3,K4,K5,K6,K7,K7,K7,K7: y=y+1: GOTO
            K1
235 ho2   K2: qq=qq+1: x=x+1-INT(RND*2)*2: GOTO K1
236 4f    K3: x=x-1: GOTO K1
237 0a    K4: x=x+1: GOTO K1
238 0K    K5: IF f(x+1,y)=14 THEN a=f(x+2,y)-30 ELSE a=f(x-2,y)-30
239 BW4     y=y+1: IF a<1 OR mm>4 THEN K1
240 nZ      mm=mm+1: k(mm,0)=m(a,0): k(mm,1)=m(a,1): k(mm,2)=m(a,2)
            +2
241 mj      GOTO K1
242 vr2   K6: IF k(j,0)=spl THEN q(i)=q(i)+3 ELSE q(i)=q(i)+1
243 Q24     q=q+qq: GOTO K0
244 U62   K7: a=f(x,y)-6: x=h(a,0): y=h(a,1)+1: GOTO K1
245 sT    K8: q(i)=q(i)/q: IF INKEY$=CHR$(27) THEN zzm=1
246 2A0 NEXT: x=1: q=0: FOR i=0 TO 9
247 zU2   IF q(i)=q AND RND>.8 AND f(i*2+1,2)=0 THEN x=i*2+1
248 bE    IF q(i)>q AND f(i*2+1,2)=0 THEN q=q(i): x=i*2+1
249 ta0 NEXT: WHILE f(x,2)>0: x=x+2: WEND: LOCATE spl*6-3,31
250 3i  COLOR 6+spl*2,7+spl*2: PRINT n$(spl): RETURN
251 gh  Start.Einwurf:
252 Ji  IF msp=2 THEN g(1)=stf+3: g(2)=stf+3: g(3)=0: g(4)=0: GOTO
        V0
253 wT  a=4: IF stf=1 THEN a=3
254 a7  FOR i=1 TO 4: g(i)=a: NEXT: IF msp=3 THEN g(4)=1
255 pA  V0: x=INT(RND*10)*2+1: IF f(x,2)>0 THEN V0
256 iq  V1: f=INT(RND*4)+1: IF g(f)=0 THEN V1
257 1t  g(f)=g(f)-1: GOSUB Einwurf: IF g(1)+g(2)+g(3)+g(4)>0 THEN
        V0
258 W8  RETURN
259 3t  Spieler.Verteilung:
260 cQ  FOR i=0 TO 4: FOR j=0 TO 4: s(i,j)=0: NEXT j,i
261 qg  IF msp=2 THEN
262 u12   FOR i=1 TO 4: s(0,i)=1: NEXT
263 5j    IF stf>1 THEN s(0,1)=stf*2: s(0,2)=stf*2
264 ra0 ELSE
265 Jp2   FOR i=1 TO 4: s(0,i)=stf*3-1: NEXT
266 1Z    IF msp=3 THEN s(0,4)=0
267 F80 END IF
268 xn  IF msp=2 THEN
269 Y32   a=stf*2: IF stf=1 THEN a=1
270 oA    s(2,1)=a: s(1,2)=a: s(2,3)=1: s(1,4)=1
271 yh0 ELSE
272 uQ2   a=v(msp*6+stf*2-20): b=v(msp*6+stf*2-19)
273 K9    FOR j=1 TO msp
274 L24     s((j)MOD msp+1,j)=a: s((j+1)MOD msp+1,j)=b
275 92      IF msp=4 THEN s((j+2)MOD msp+1,j)=b
276 Xc2   NEXT
277 SQ0 END IF: RETURN
278 uh  sieger.anz:
279 zC  WINDOW 3,,(30,40)-(290,205),0,2: SOUND 1440,2.5,255: SOUND
        1,1
280 hV  SOUND 1440,2,255: SOUND 1,1: SOUND 1920,7,255
281 k4  FOR i=0 TO 4
282 Dm2   LINE(i,i)-(260-i,175-i),d(7+i),b: NEXT
283 vt0 FOR x=10 TO 250 STEP 3
284 gV2   LINE(x,9)-(x,67),7+gwr*2: NEXT
285 ZT0 FOR y=11 TO 66 STEP 3
286 9E2   LINE(8,y)-(252,y),7+gwr*2: NEXT
287 DX0 FOR x=0 TO 22: FOR y=0 TO 4
288 tP2   PUT(x*11+10,y*11+11),z(0,-gwr*(MID$(t$(2,y),x+1,1)="x")),
          PSET
289 s00 NEXT y,x
290 9m  LINE(37,77)-(226,90),5,b: LINE(38,78)-(225,89),6,bf
291 9y  LOCATE 11,6: COLOR 3,6: PRINT n$(gwr)" hat gewonnen"
292 Le  FOR i=0 TO spz-1: LOCATE 14+i*2,7
293 pU2   COLOR 8+i*2,0: PRINT n$(i+1);: COLOR 3
294 eE    PRINT " gewann"gw(i+1)CHR$(8)"x": NEXT
295 Rk0 SLEEP: SLEEP: WINDOW CLOSE 3: RETURN
296 3c  Title:
```

```
297 Z4  WINDOW 3,,(10,0)-(310,242),0,2
298 e0  a=60: b=0: GOSUB Schrift: a=132: b=1: GOSUB Schrift
299 uE  FOR y=4 TO 10 STEP 2
300 To2   FOR x=0 TO 292 STEP 2
301 6j4     dy=SIN((3.143*x)/292)*y
302 1G      SCROLL(x,124)-(x+1,184+y*4),0,dy
303 Hn      SCROLL(292-x,54-y*4)-(293-x,123),0,-dy
304 3u0 NEXT x,y: COLOR 2,0
305 aW  LOCATE 15,12: PRINT "Copyright 1988"
306 tM  LOCATE 16,18: PRINT "by": LOCATE 17,12
307 zo  PRINT "Christian Marz": SLEEP: SLEEP
308 WO  FOR p=1 TO 0 STEP -.05: FOR i=0 TO 15
309 iG2   PALETTE i,p(i,0)*p,p(i,1)*p,p(i,2)*p
310 2a0 NEXT i,p: WINDOW CLOSE 3: RETURN
311 Eo  Schrift:
312 FZ  FOR i=0 TO 4
313 dZ2   LINE(i,a-9+i)-(293-i,a+62-i),d(11-i),b: NEXT
314 9Y0 FOR y=a-2 TO 55+a STEP 3
315 Mz2   LINE(5,y)-(288,y),4+INT(RND*4): NEXT
316 RyO FOR x=7 TO 287 STEP 3
317 VT2   LINE(x,a-4)-(x,57+a),4+INT(RND*4): NEXT
318 v10 FOR x=0 TO 24: SOUND 2000,.1,255
319 HP2   SOUND 1200-b*200,.5,200: SOUND 1000-b*200,.5
320 k6    FOR y=0 TO 4: f=INT(RND*2)+b*2+1
321 eC4     PUT(x*11+10,y*11+a),z(0,-f*(MID$(t$(b,y),x+1,1)="x")),P
            SET
322 r10 NEXT y,x: RETURN
323 Pq  Menue:
324 Mj  WINDOW 3,,(30,30)-(278,206),0,2
325 2v  FOR i=0 TO 4: LINE(i,i)-(248-i,185-i),d(11-i),b: NEXT
326 Lz  LINE(4,35)-(244,37),5,b: LINE(3,36)-(245,36),6
327 mu  FOR x=6 TO 242 STEP 2: LINE(x,6)-(x,33),4: NEXT
328 85  COLOR 7,4: LOCATE 3,3
329 2Q  PRINT" Crazy Coins - "CHR$(169)"1988 by CM "
330 Co  COLOR 3,0: LOCATE 7,4: PRINT"Anzahl der Mitspieler:"spz
331 aN  LOCATE 9,4: PRINT"Anz. Münzen / Spieler:"mm(stf)
332 En  LOCATE 11,4: PRINT"Spielfeld: "f$(spf)
333 LF  FOR i=0 TO 2: PUT(10,46+i*16),e(0,20): NEXT
334 XV  LINE(4,94)-(244,96),5,b: LINE(3,95)-(245,95),6
335 vx  FOR i=1 TO 4: LOCATE 12+2*i,2: COLOR 6+i*2
336 DX2   PRINT"Spieler"i"heißt:  ";: COLOR 13: PRINT n$(i): NEXT
337 YU0 LOCATE 22,2: COLOR 3,6: PRINT " Spielen  Editieren   Ende!
        "
338 oh  LINE(8,167)-(239,176),6,b: LINE(80,166)-(167,177),0,b
339 8K  LINE(8,166)-(79,177),5,b: LINE(81,166)-(166,177),5,b
340 JC  LINE(168,166)-(239,177),5,b: COLOR 3,0
341 A3  spz=spz-2: stf=stf-1: WHILE MOUSE(0)<>0: WEND
342 gO  Menue1:
343 gC  WHILE MOUSE(0)>=0: WEND: a=MOUSE(1): b=MOUSE(2)
344 Po  IF b>45 AND b<57 THEN spz=(spz+1)MOD 3: LOCATE 7,26: PRIN
        T spz+2: FOR i=1 TO 4: gw(i)=0: NEXT
345 2v  IF b>61 AND b<73 THEN stf=(stf+1)MOD 3: LOCATE 9,26: PRIN
        T mm(stf+1)
346 9q  IF b>77 AND b<89 THEN spf=(spf+1)MOD 6: LOCATE 11,15: PRI
        NT LEFT$(f$(spf)+SPACE$(16),16)
347 VM  IF b>13*8 AND b<20*8 THEN
348 gQ2   b=(b-13*8)\16: LOCATE 14+b*2,20: INPUT"",a$
349 j2    IF a$<>"" THEN n$(b+1)=LEFT$(a$+SPACE$(10),10)
350 HR    LOCATE 14+b*2,20: PRINT n$(b+1)
351 bU0 END IF
352 bx  IF b<166 OR b>177 THEN FOR i=0 TO 999: NEXT: GOTO Menue1
353 Nj  IF a>7 AND a<80 THEN spz=spz+2: stf=stf+1: GOTO Menue2
354 rO  IF a>80 AND a<167 THEN spz=spz+2: stf=stf+1: GOTO Menue3
355 OP  IF a>167 AND a<240 THEN Ende
356 Kb  GOTO Menue1
357 57  Menue2: WINDOW CLOSE 3: RETURN
358 9q  Menue3: WINDOW CLOSE 3: GOSUB Editor: GOTO Menue
359 JB  Editor:
360 wI  j=0: COLOR ,0: CLS: COLOR 3,0: LOCATE 1,31
361 yA  PRINT"Editieren:": LOCATE 3,31: PRINT f$(0)
362 3g  FOR x=0 TO 2: FOR y=0 TO 6
363 Gr2   LINE(240+x*14,40+y*14)-(254+x*14,54+y*14),3,b
364 ZX    PUT(242+x*14,42+y*14),e(0,b(x*7+y))
365 6E0 NEXT y,x
366 1p  LOCATE 23,31: PRINT"Leerfeld": LOCATE 25,31: PRINT"Laden"
367 V3  LOCATE 27,31: PRINT"Speichern": LOCATE 29,31: PRINT"Fertig"
368 oF  Edit0: WHILE MOUSE(0)<>0: WEND
369 0r  FOR y=22 TO 0 STEP -1: FOR x=0 TO 20
370 422   PUT(x*11,y*11),e(0,t(x,y)),PSET: NEXT x,y: BEEP
371 RzO Edit1:
372 9f  WHILE MOUSE(0)>=0: WEND: a=MOUSE(1): b=MOUSE(2)
373 zs  IF a>240 THEN
374 cc2   IF a<282 AND b>40 AND b<138 THEN j=b(((a-240)\14)*7+(b
          -40)\14): PUT(256,150),e(0,j),PSET
375 Ym    IF b>175 AND b<184 THEN GOSUB Init.Feld1: GOTO Edit0
376 ET    IF b>191 AND b<200 THEN
377 Mf4     LOCATE 3,31: INPUT"",a$: IF a$="" THEN Edit1
378 1j      f$(0)=LEFT$(a$,10): OPEN f$(0) AS #1
379 Wx      IF LOF(1)=0 THEN BEEP: CLOSE: KILL f$(0): LOCATE 3,31:
            PRINT"--Error-- ": GOTO Edit1
380 CK      CLOSE: OPEN "I",#1,f$(0)
381 Vm      FOR x=0 TO 20: FOR y=0 TO 22: INPUT #1,t(x,y): NEXT y,
            x
382 GX      CLOSE: GOTO Edit0
383 702   END IF
384 m6    IF b>207 AND b<216 THEN
385 Rn4     LOCATE 3,31: PRINT f$(0);SPACE$(10)
386 Vo      LOCATE 3,31: INPUT"",a$: IF a$="" THEN Edit1
387 TN      f$(0)=LEFT$(a$,10): OPEN "O",#1,f$(0)
388 1F      FOR x=0 TO 20: FOR y=0 TO 22: PRINT #1,t(x,y): NEXT y,
            x
389 Qi      CLOSE: GOTO Edit1
390 E72   END IF
391 1B    IF b>223 AND b<232 THEN RETURN
392 veO ELSE
393 ER2   x=a\11: y=b\11: IF (j=0 OR j>19) AND y=22 THEN Edit3
394 sZ    IF x>20 OR y<2 OR y>21 THEN SOUND 3000,1: GOTO Edit2
395 M4    IF j=5 AND (x<1 OR x>19) THEN SOUND 3000,1: GOTO Edit2
396 dJ    Edit3: t(x,y)=j: PUT(x*11,y*11),e(0,j),PSET
397 sz    IF j=5 THEN t(x,y+1)=22: PUT(x*11,y*11+11),e(0,22),PSET
398 ySO Edit2:
399 Ms  END IF: GOTO Edit1
400 mG  Get.Elemente:
401 j4  FOR i=0 TO 5
402 P62   LINE(1+d(i),64-d(i))-(21,44),d(i+7),bf
403 Qh    AREA(6+d(i),27): AREA(27,6+d(i)): AREA(48-d(i),27)
404 C7    AREA(27,48-d(i)): COLOR d(11-i): AREAFILL
405 ch0 NEXT
406 op  LINE(0,11)-(10,13),5,b: LINE(0,12)-(10,12),6
407 j2  FOR i=0 TO 3
408 SL2   CIRCLE(38+i*11,5),5,9+i*2: PAINT(38+i*11,5),9+i*2
409 JB    CIRCLE(38+i*11,5),5,8+i*2,,,1: CIRCLE(27,27),6-i,4+i,,,1
410 HE    PAINT(27,27),4+i: GET(33+i*11,0)-(43+i*11,10),z(0,i+1)
411 PR    LINE(55+i*11,18)-(65+11*i,25),8+i*2,bf
412 UU    FOR j=17 TO 26 STEP 9
413 Oz4     CIRCLE(60+i*11,j),5,4,,,.5: PAINT(60+i*11,j),4
414 cy      CIRCLE(60+i*11,j),5,8+2*i,,,.5
415 m00 NEXT j,i: RESTORE Elemente
416 zZ  FOR i=0 TO 22
417 XS2   READ x,y: GET(x,y)-(x+10,y+10),e(0,i): NEXT
418 jRO GET(22,6)-(32,10),L: PUT(22,50),L
419 xc  GET(22,44)-(32,54),e(0,14): GET(0,0)-(10,10),L: RETURN
420 Jy  DATA  0, 0,.1,   0,  0,.1,  1, 0,0,  1, 1,1
421 h2  DATA  0, 0, 1, .32,.32, 1, .5,.5,1, .7,.7,1
422 Ki  DATA .8, 0, 0,  .5,  0, 0,  0,.8,0,  0,.5,0
423 Ic  DATA .8,.8,.8,  .5, .5,.5, .8,.8,0, .5,.5,0
424 mm  DATA 1,1,2,3,4,4,0,1,1,2,2,3,0,1,3,6,8,9,0,4,5,6,7,6,0
425 s0  DATA 0,19,5,14,20,21,22,2,16,18,6,7,8,9,3,17,1,10,11,12,13
426 oS  DATA " xxx xxx   xx  xxxxx x  x"
427 fH  DATA "x    x  x x  x    x  x  x"
428 Ec  DATA "x    xxx  xxxx   x    xxx"
429 BT  DATA "x    x  x x  x  x       x"
430 Ok  DATA " xxx x  x x  x xxxxx xxx "
431 Tb  DATA " xxx  xx  xxx x   x  xxxx"
432 UC  DATA "x    x  x  x  xx  x x    "
433 xP  DATA "x    x  x  x  x x x  xxx "
434 we  DATA "x    x  x  x  x  xx     x"
435 x5  DATA " xxx  xx  xxx x   x xxxx "
436 yQ  DATA " xxxx xxx xxxx  xxx  x"
437 bF  DATA "x      x  x    x     x"
438 mk  DATA " xxx   x  xxx  x  xx x"
439 N1  DATA "    x  x  x    x   x  "
440 Su  DATA "xxxx  xxx xxxx  xxx  x"
441 As  DATA CC.Edit,1. Einfach,2. Mittel,3. Schwer,4. Komplex,"5.
        Verrückt"
442 TL  Elemente:
443 Gb  DATA  0, 0,22, 0,11,11,33,11, 0, 0, 0,11,55,11,66,11
444 5c  DATA 77,11,88,11,55,22,66,22,77,22,88,22, 0, 0, 0, 0
445 3T  DATA 11,33,33,33,22,44, 0, 3,22,22, 0,44,11,55
446 E5  Feld3: DATA 1,7,19,7,19,14,1,14,19,15,1,15,1,8,19,8
447 Wl  Feld4: DATA 4,3,4,19,16,3,16,19
(C) 1989 M&T
```

Listing 1. (Schluß)

Vollbremsung im Amiga

Highscore-Jäger und Programmtüftler aufgepaßt: Mit unserem »Soft-Stopper« genügt fortan ein Tastendruck, und schon frieren Sie Ihr Lieblingsspiel oder Ihr selbstgeschriebenes Programm ein. Ein Druck auf die Maustaste genügt, sofort geht der Spielespaß weiter.

Kennen Sie das? Wieder einmal sitzen Sie vor dem brandneuen Spiele-Hit, der Highscore ist zum Greifen nah. Doch gerade im entscheidenden Moment klingelt das Telefon und noch während Sie den Hörer abheben, verlieren Sie Ihr letztes Raumschiff. Hätten Sie das Spiel anhalten können, wäre dies nicht geschehen. Auch beim Testen eigener Programme und bei Bildschirmfotos erweist sich eine Pausenfunktion als überaus nützlich.

Der »Soft-Stopper« ist ein kurzes Programm in Maschinensprache, das, einmal aufgerufen, unauffällig im Hintergrund auf ein Drücken der Enter-Taste (im Zahlenblock ganz rechts) lauert und sodann alle Amiga-Aktivitäten einfriert.

Die Bremse lauert im Hintergrund

Erst ein Druck auf die linke Maustaste setzt den Computer wieder in Gang.

Um bei unserem Beispiel mit dem Spielprogramm zu bleiben: Zum Anhalten des Spiels drücken Sie die Enter-Taste. Nun können Sie in aller Ruhe zu Mittag essen, Gäste empfangen oder tolle Bildschirmfotos schießen. Sie wollen weiterspielen? Ein Klick mit der linken Maustaste und schon geht die Punktehatz weiter.

Den Soft-Stopper geben Sie auf zweierlei Arten ein. Listing 1 ist der Assembler-Quellcode, den Besitzer des Seka-Assemblers unverändert eintippen und assemblieren können. Der Quellcode zeigt auch deutlich die prinzipielle Arbeitsweise des Soft-Stoppers, der alle laufenden Programme durch Sperren der Interrupts anhält. Das Stopper-Programm selbst liegt in einem Speicherbereich, der normalerweise nicht verwendet wird.

Verfügen Sie nicht über diesen Assembler, dann geben Sie einfach Listing 2 mit Amiga-Basic ein und starten das Programm. Das Basic-Programm erzeugt aus den DATA-Werten den Soft-Stopper und schreibt diesen als ausführbares Programm auf Diskette.

Der richtige Aufruf

Nun müssen Sie nur noch das Programm einmal vor dem Start Ihres Spiels aufrufen. Tippen Sie dazu entweder im CLI direkt »Soft-Stopper« ein oder speichern Sie das kurze Programm auf Ihre Spiele-Diskette und fügen Sie den Aufruf (»Soft-Stopper«) am Beginn der Startup-Sequence dieser Diskette ein. Die Startup-Sequence befindet sich im »S«-Ordner.

Der Soft-Stopper funktioniert auch bei »Interceptor«, »World Tour Golf« und anderen kommerziellen Spielen. Versuchen Sie Ihr Glück.

Fallen Ihnen noch Ergänzungen zu diesem Programm ein, so würden wir uns sehr über eine Zusendung freuen. Wie wäre es zum Beispiel mit einer Routine, die den Bildschirm dunkel schaltet, wenn einige Zeit kein Tastendruck und keine Mausbewegung registriert wurde? Oder vielleicht... *(Ralf Dietrich/rs)*

Programmname:	SoftStopper
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Assembler

```
 1  ********************************
 2  *        Soft-Stopper          *
 3  *      von Ralf Dietrich       *
 4  ********************************
 7  START:
 8  EVEN
 9  MOVEM.L D0-D7/A0-A6,-(A7)
10  LEA BSTART(PC),A0
11  LEA ENDE(PC),A1
12  MOVE.L #$100,A2
13  MOVE.L A2,A3
14
15  COPYSCHL:
16  MOVE.W (A0)+,(A2)+
17  CMP.L A0,A1
18  BNE.S COPYSCHL
19  JSR (A3)
20  MOVEM.L (A7)+,D0-D7/A0-A6
21  RTS
22
23  BSTART:
24  LEA.L IRBUFFER+2(PC),A0
25  MOVE.L $68,(A0)
26  LEA NEWIR(PC),A0
27  MOVE.L A0,$68
28  RTS
29
30  NEWIR:
31  MOVEM.L D0-D7/A0-A6,-(A7)
32  MOVE.B $BFEC01,D0
33  CMP.B #121,D0
34  BNE.S NOSPKEY
35  BSR PRGSTOP
36
37  NOSPKEY:
38  MOVEM.L (A7)+,D0-D7/A0-A6
39
40  IRBUFFER:
41  JMP $FFFFFFFF
42
43  PRGSTOP:
44  MOVE.W SR,D0
45  MOVE.W #$2700,SR
46
47  PRGSTOPWAIT:
48  BTST #6,$BFE001
49  BNE.S PRGSTOPWAIT
50  MOVE.B #120,$BFEC01
51
52  ERSTLOSLASSEN:
53  BTST #6,$BFE001
54  BEQ.S ERSTLOSLASSEN
55  MOVE.W D0,SR
56  RTS
57
58  ENDE:
```

Listing 1. Der Seka-Quellcode des Soft-Stoppers

Programmname:	SoftStopper_Gen
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

```
Programmautor: Ralf Dietrich
----------------------------

 1 pB0 REM Generiert lauffaehiges Programm
 2 ag  CLS
 3 Br  OPEN "df2:softstopper" FOR OUTPUT AS 1
 4 BS  READ anz
 5 oa  FOR i=1 TO anz
 6 3n1  READ h$
 7 yB2   wert1=ASC(LEFT$(h$,1))
 8 bP    IF wert1>64 THEN wert1=wert1-87 ELSE wert1=wert1-48
 9 FI    wert1=wert1*16
10 7c    wert2=ASC(RIGHT$(h$,1))
11 wp    IF wert2>64 THEN wert2=wert2-87 ELSE wert2=wert2-48
12 P1    wert=wert1+wert2
13 9G    PRINT #1,CHR$(wert);
14 J00 NEXT
15 3n  CLOSE 1
16 0v  END
17 yc  Werte:
18 Kr  DATA 188
19 ph  DATA 00,00,03,f3,00,00,00,00,00,00
20 bk  DATA 00,02,00,00,00,00,00,00,00,01
21 bk  DATA 00,00,00,20,00,00,00,01,00,00
22 Yn  DATA 03,e9,00,00,00,20,48,e7,ff,fe
23 Mx  DATA 41,fa,00,1c,43,fa,00,72,24,7c
24 6t  DATA 00,00,01,00,26,4a,34,d8,b3,c8
25 4N  DATA 66,fa,4e,93,4c,df,7f,ff,4e,75
26 on  DATA 41,fa,00,2e,20,b9,00,00,00,68
27 jp  DATA 41,fa,00,0a,23,c8,00,00,00,68
28 n0  DATA 4e,75,48,e7,ff,fe,10,39,00,bf
29 dz  DATA ec,01,0c,00,00,79,66,04,61,00
30 IB  DATA 00,0c,4c,df,7f,ff,4e,f9,ff,ff
31 XS  DATA ff,ff,40,c0,46,fc,27,00,08,39
32 0k  DATA 00,06,00,bf,e0,01,66,f6,13,fc
33 1z  DATA 00,78,00,bf,ec,01,08,39,00,06
34 id  DATA 00,bf,e0,01,67,f6,46,c0,4e,75
35 me  DATA 00,00,ff,ff,00,00,03,ec,00,00
36 D8  DATA 00,00,00,00,03,f2,00,00,03,eb
37 ja  DATA 00,00,00,01,00,00,03,f2
(C) 1989 M&T
```

Listing 2. Kurz und bündig: das Stopper-Programm in DATA-Zeilen, einzugeben mit Amiga-Basic

Der Bilderklau

Dieses Bild muß ich haben! Wie oft haben Sie sich schon gefragt, wie man an Grafiken professioneller Programme kommt? »SnipIFF« erledigt diese Aufgabe für Sie. Das Besondere an diesem Programm: Es belegt nur sehr wenig Speicher und läuft unauffällig im Hintergrund.

Egal in welchem Programm Sie sich gerade befinden, »SnipIFF« speichert fast jeden Bildschirminhalt oder einen Ausschnitt davon. Ob Sie dieses Bild ausdrucken, mit einem Malprogramm nachbearbeiten oder einfach nur genauer betrachten wollen, SnipIFF macht's möglich. SnipIFF ist einfach zu installieren, und auch für die Aufrufe brauchen Sie sich nur wenig zu merken: Mit einer einfachen Tastenkombination <Amiga links-Shift links-s> speichern Sie den aktuellen Bildschirminhalt in einer »IFF«-Datei, mit <Amiga links-Shift links-b> einen Ausschnitt.

Sie starten das Programm mit »SnipIFF« vom CLI aus; darauf wird es installiert und bleibt solange resident, bis es mit <Amiga links-Shift links-q> wieder aus dem Speicher entfernt wird oder bis ein Reset ausgelöst wird. Sie brauchen SnipIFF nicht mit »Run« zu starten, denn es arbeitet automatisch im Hintergrund; Sie können nach der Installation normal weiterarbeiten.

Wenn Sie »SnipIFF« ohne Parameter aufrufen, werden alle Dateien unter dem Namen »picxx« in »df1:« gespeichert. Dabei ist »xx« eine aufsteigende Numerierung der Bilder. Wollen Sie die Bilder in einem anderen Laufwerk haben, dann sollten Sie bei der Installation das Verzeichnis als Parameter übergeben: »SnipIFF [Verzeichnis]«.

Standardmäßig werden IFF-Dateien komprimiert und auf Wort-Grenze gespeichert. Falls Ihr Malprogramm eine derartige Datei nicht laden kann, verändern Sie mit den Parametern »-u« (für unkomprimiert) oder »-b« (für Speichern auf Byte-Grenze) diese Einstellungen. Beispielsweise sollten

Weiterbearbeiten mit DPaint

Sie die Option »-b« verwenden, wenn Sie mit »DPaint« arbeiten, da dieses Programm IFF-Dateien auf Byte-Grenze erwartet.

Bevor Sie SnipIFF starten können, geben Sie Listing 1 mit dem Checksummer ein.

Das Programm »SnipIFF_Gen« ist ein Basic-Programm, das Sie nur einmal starten. Bei diesem Durchlauf wird das ablauffähige Programm SnipIFF erzeugt und auf Diskette geschrieben.

Starten Sie nun Ihren »Grafikdieb« mit

```
SnipIFF [Verzeichnis]
 [-u] [-b]
```

Hinterher können Sie eine beliebige Anwendung laden, von der die Grafik gespeichert werden soll. SnipIFF kann alle Bildschirme speichern außer denen, welche den »HAM«-Modus verwenden oder die Tastatur blockieren. Wenn Sie einen Bildschirmausschnitt speichern wollen (mit <Amiga links-Shift links-b>), können Sie mit der Maus einen Rahmen über den Bereich legen, den Sie speichern wollen.

Wenn Sie SnipIFF nicht mehr benötigen, entfernen Sie es mit der Tastenkombination <Amiga links-Shift links-q> aus dem Speicher.

Wir veröffentlichen SnipIFF als Data-Lader, damit auch Leser, die keinen Assembler haben, in den Genuß dieses Programms kommen.

Wer sich für die Programmierung interessiert, findet auf der Programmservice-Diskette ein Assembler-Source-File, das mit dem Public Domain-Assembler A68K von der Fish-Disk 110 assembliert werden kann. Sie benötigen dazu das Include-File »amiga.library« vom Lattice-C-Compiler und den Public Domain-Linker BLink in der Version 7.1.

Für fortgeschrittene Assembler-Programmierer dürfte es kein Problem sein, dieses Programm beispielsweise auf den »DevPac« zu übertragen oder die eine oder andere Routine hinzuzufügen.

Übrigens läßt sich die Routine »ILBMSave« auch in eigenen Programmen verwenden, um IFF-Bilder zu speichern. Sie müssen als Parameter die Adresse des Viewport, die Startkoordinaten sowie Höhe und Breite angeben.

Sie können sich damit Ihre eigene »Grabbit«-Version bauen. Viel Spaß beim Tüfteln!

(so)

SnipIFF-Kurzreferenz

<Amiga links-Shift links-q>	beendet SnipIFF
<Amiga links-Shift links-s>	speichert den Bildschirminhalt
<Amiga links-Shift links-b>	speichert einen Bildschirmausschnitt
Aufruf mit:	
SnipIFF [Verz.][-u][-b]	installiert SnipIFF

Tabelle 1. Der Aufruf und die Tastenkombinationen, mit denen Sie »SnipIFF« steuern

Programmname:	SnipIFF_gen
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic

```
Programmautor: Werner Günther
------------------------------

 1 0m0 REM Generiert lauffähiges Programm
 2 ag  CLS
 3 AJ  OPEN "df1:snipiff/snipiff.bas" FOR OUTPUT AS 1
 4 BS  READ anz
 5 oa  FOR i=1 TO anz
 6 3n1  READ h$
 7 yB2   wert1=ASC(LEFT$(h$,1))
 8 bP    IF wert1>64 THEN wert1=wert1-87 ELSE wert1=wert1-48
 9 FI    wert1=wert1*16
10 7c    wert2=ASC(RIGHT$(h$,1))
11 wp    IF wert2>64 THEN wert2=wert2-87 ELSE wert2=wert2-48
12 Pi    wert=wert1+wert2
13 9G    PRINT #1,CHR$(wert);
14 J00 NEXT
15 3n  CLOSE 1
16 0v  END
17 yc  Werte:
18 Gx  DATA 5204
19 ph  DATA 00,00,03,f3,00,00,00,00,00,00
20 7w  DATA 00,0b,00,00,00,00,00,00,00,0a
21 sw  DATA 00,00,00,63,00,00,01,58,00,00
22 di  DATA 01,3f,00,00,00,29,00,00,00,40
23 5A  DATA 00,00,00,40,00,00,00,05,00,00
24 0r  DATA 00,37,00,00,00,09,00,00,00,1d
25 CD  DATA 00,00,00,25,00,00,03,e9,00,00
26 CU  DATA 00,63,42,81,34,3c,00,ff,43,f9
27 um  DATA 00,00,04,f6,52,02,0c,30,00,20
28 XG  DATA 20,00,67,f6,0c,30,00,2d,20,00
29 Jn  DATA 66,20,52,02,0c,30,00,61,20,00
30 sK  DATA 66,06,42,79,00,00,04,8e,0c,30
31 j4  DATA 00,75,20,00,66,d6,42,39,00,00
32 t1  DATA 05,5f,60,ce,0c,30,00,0a,20,00
33 St  DATA 67,12,0c,30,00,20,20,00,67,be
34 rk  DATA 12,f0,20,00,52,02,52,01,60,e6
35 Xt  DATA 4a,81,66,02,72,08,23,c1,00,00
36 VG  DATA 04,aa,30,3c,00,07,41,f9,00,00
37 i5  DATA 05,10,43,f9,00,00,04,f6,d3,c1
38 bw  DATA 12,d8,51,c8,ff,fc,20,78,00,04
39 qu  DATA 23,c8,00,00,04,92,20,68,01,7a
40 zA  DATA 2f,08,43,f9,00,00,05,18,2c,78
41 ye  DATA 00,04,4e,ae,fe,ec,23,c0,00,00
42 uK  DATA 04,96,43,f9,00,00,05,2a,20,57
43 IV  DATA 2c,78,00,04,4e,ae,fe,ec,23,c0
44 wp  DATA 00,00,04,9a,43,f9,00,00,05,36
45 eT  DATA 20,5f,2c,78,00,04,4e,ae,fe,ec
46 44  DATA 23,c0,00,00,04,a6,43,f9,00,00
47 nq  DATA 05,54,2c,78,00,04,4e,ae,fe,da
48 hr  DATA 4a,80,67,0e,24,3c,00,00,01,5a
49 hv  DATA 26,3c,00,00,00,19,60,4c,2c,78
50 u2  DATA 00,04,4e,ae,ff,7c,41,fa,ff,0a
51 0s  DATA 91,fc,00,00,00,04,26,28,00,00
```

Listing 1. »SnipIFF_Gen« geben Sie mit dem Checksummer auf Seite 159 ein. Nach dem ersten Programmlauf wird das Programm »SnipIFF« erzeugt.

```
52 at  DATA 42,a8,00,00,23,c3,00,00,04,be
53 w3  DATA 22,3c,00,00,05,54,74,32,28,3c
54 v3  DATA 00,00,07,d0,2c,79,00,00,04,9a
55 Hc  DATA 4e,ae,ff,76,2c,78,00,04,4e,ae
56 1P  DATA ff,76,24,3c,00,00,01,73,26,3c
57 qH  DATA 00,00,00,18,2c,79,00,00,04,9a
58 BJ  DATA 4e,ae,ff,c4,22,00,2c,79,00,00
59 Bi  DATA 04,9a,4e,ae,ff,d0,42,81,2c,79
60 ez  DATA 00,00,04,9a,4e,ee,ff,70,53,6e
61 Mp  DATA 69,70,49,66,66,20,61,6c,72,65
62 yk  DATA 61,64,79,20,6d,6f,75,6e,74,65
63 ig  DATA 64,2e,0a,53,6e,69,70,49,66,66
64 4A  DATA 20,56,31,2e,30,20,69,6e,73,74
65 7n  DATA 61,6c,6c,65,64,2e,0a,00,00,00
66 WQ  DATA 03,ec,00,00,00,02,00,00,00,00
67 pU  DATA 00,00,00,e0,00,00,01,2e,00,00
68 AO  DATA 00,13,00,00,01,00,00,00,08
69 xF  DATA 00,00,00,38,00,00,60,00,00
70 Gj  DATA 00,6a,00,00,00,70,00,00,00,82
71 Mv  DATA 00,00,00,8e,00,00,00,9c,00,00
72 ta  DATA 00,a2,00,00,00,b2,00,00,00,b8
73 G2  DATA 00,00,00,c8,00,00,00,ce,00,00
74 sM  DATA 01,08,00,00,01,0e,00,00,01,1c
75 pn  DATA 00,00,01,3a,00,00,01,46,00,00
76 5M  DATA 01,52,00,00,00,01,00,00,00,02
77 1w  DATA 00,00,00,2a,00,00,00,00,00,00
78 Lf  DATA 03,f2,00,00,03,e9,00,00,01,58
79 hj  DATA 48,78,00,00,48,78,00,00,4e,b9
80 Ew  DATA 00,00,00,00,4f,ef,00,08,4a,80
81 GV  DATA 67,00,02,d8,23,c0,00,00,04,9e
82 ez  DATA 2f,00,4e,b9,00,00,00,00,4f,ef
83 32  DATA 00,04,4a,80,67,00,02,c0,23,c0
84 OI  DATA 00,00,04,a2,22,40,41,fa,05,0d
85 NH  DATA 42,80,42,81,2c,7a,04,50,4e,ae
86 gR  DATA fe,44,4a,80,66,00,02,a2,23,c0
87 Pz  DATA 00,00,04,c6,22,7a,04,4c,23,7c
88 lg  DATA 00,00,04,de,00,28,33,7c,00,09
89 sm  DATA 00,1c,2c,7a,04,2a,4e,ae,fe,38
90 37  DATA 4a,80,66,00,02,7c,23,c0,00,00
91 YL  DATA 04,c2,20,7a,04,16,22,68,01,14
92 oo  DATA 23,c9,00,00,04,b6,70,ff,2c,7a
93 eH  DATA 04,06,4e,ae,fe,b6,4a,80,6b,00
94 gf  DATA 02,58,42,81,01,c1,23,c1,00,00
95 oR  DATA 04,ba,43,fa,04,a3,2c,7a,03,ea
96 Me  DATA 4e,ae,fe,da,23,c0,00,00,04,ca
97 hm  DATA 20,3a,04,04,2c,7a,03,d8,4e,ae
98 dn  DATA fe,c2,10,3a,04,32,0c,00,00,10
99 EQ  DATA 67,00,02,24,2c,7a,03,c8,4e,ae
100 oQ DATA fe,7a,28,7a,03,c0,28,6c,00,3c
101 1G DATA 23,cc,00,00,04,ae,4b,ec,00,54
102 xT DATA 26,6c,00,50,23,cd,00,00,04,b2
103 tR DATA 10,3a,04,02,42,39,00,00,04,f4
104 yj DATA 0c,00,00,21,67,00,01,7e,13,fc
105 bo DATA 00,01,00,00,04,f5,70,32,22,7a
106 pa DATA 03,bc,2c,7a,03,80,4e,ae,fe,d4
107 Va DATA 13,ed,00,19,00,00,05,5e,70,01
108 hL DATA 22,4d,2c,7a,03,80,4e,ae,fe,aa
109 uf DATA 13,ed,00,1c,00,00,05,5d,70,02
110 sj DATA 22,4d,2c,7a,03,6c,4e,ae,fe,9e
111 4i DATA 33,ed,00,20,00,00,04,da,3b,7c
112 zu DATA 00,01,00,20,33,ed,00,22,00,00
113 RP DATA 04,dc,3b,7c,ff,ff,00,22,08,39
114 9q DATA 00,06,00,bf,e0,01,66,f6,61,00
115 oH DATA 02,0a,4a,00,66,ee,33,ec,00,12
116 R7 DATA 00,00,04,ce,33,ec,00,10,00,00
117 mO DATA 04,d0,08,39,00,06,00,bf,e0,01
118 VJ DATA 67,f6,20,3c,00,01,86,a0,51,c8
119 GA DATA ff,fe,08,39,00,06,00,bf,e0,01
120 4h DATA 67,e2,33,fa,03,30,00,00,04,d2
121 XP DATA 33,fa,03,2a,00,00,04,d4,3a,2c
122 sM DATA 00,12,3c,2c,00,10,33,c5,00,00
123 ST DATA 04,d6,33,c6,00,00,04,d8,61,00
124 Ym DATA 01,ce,10,39,00,bf,e0,01,02,00
125 jD DATA 00,40,13,c0,00,00,05,5c,67,14
126 BC DATA 30,2c,00,12,b0,7a,02,fa,66,12
127 D2 DATA 30,2c,00,10,b0,7a,02,f2,67,da
128 nr DATA 61,00,01,86,4a,00,66,d2,30,2c
129 qY DATA 00,12,b0,7a,02,d6,6e,06,33,c0
130 1V DATA 00,00,04,ce,30,2c,00,10,b0,7a
131 Z2 DATA 02,c8,6e,06,33,c0,00,00,04,d0
132 1Y DATA 3a,3a,02,c2,3c,3a,02,c0,61,00
133 s6 DATA 01,74,4a,39,00,00,05,5c,66,00
134 ZQ DATA ff,76,42,80,10,3a,03,32,22,4d
135 cp DATA 2c,7a,02,74,4e,ae,fe,aa,42,80
136 wq DATA 10,3a,03,21,22,4d,2c,7a,02,64
137 BX DATA 4e,ae,fe,9e,3b,7a,02,90,00,20
138 PW DATA 3b,7a,02,8c,00,22,26,6c,00,50
139 bF DATA 30,2b,00,08,d0,7a,02,70,32,2b
140 8w DATA 00,0a,d2,7a,02,6a,34,3a,02,6c
141 OM DATA 94,7a,02,60,52,42,36,3a,02,64
142 3O DATA 96,7a,02,58,52,43,60,14,30,2b
143 4K DATA 00,08,32,2b,00,0a,22,7a,02,26
144 YB DATA 34,29,00,0c,36,29,00,0e,22,7a
145 8X DATA 02,1a,43,e9,00,2c,41,fa,02,5a
146 EG DATA 18,3a,02,bf,4a,39,00,00,04,f5
147 IC DATA 67,1c,48,e7,f8,c0,70,14,22,7a
148 4I DATA 02,18,2c,7a,01,dc,4e,ae,fe,d4
149 p5 DATA 42,39,00,00,04,f5,4c,df,03,1f
150 UH DATA 4e,b9,00,00,00,1c,20,3a,01,dc
151 B2 DATA 41,fa,02,24,12,30,00,01,52,01
152 hO DATA 0c,01,00,3a,66,06,72,30,52,30
153 1y DATA 00,00,11,81,00,01,60,00,fd,c8
154 tT DATA 4a,b9,00,00,04,c2,66,1a,22,7a
155 xu DATA 01,aa,23,7c,00,00,04,de,00,28
156 8r DATA 33,7c,00,0a,00,1c,2c,7a,01,88
157 33 DATA 4e,ae,fe,38,4a,b9,00,00,04,c6
158 aG DATA 66,0c,22,7a,01,88,2c,7a,01,74
159 5i DATA 4e,ae,fe,3e,4a,b9,00,00,04,a2
160 Ik DATA 67,10,2f,39,00,00,04,a2,4e,b9
161 wP DATA 00,00,00,14,4f,ef,00,04,4a,b9
162 Y7 DATA 00,00,04,9e,67,10,2f,39,00,00
163 9c DATA 04,9e,4e,b9,00,00,00,96,4f,ef
164 6n DATA 00,04,2c,7a,01,3c,4e,ae,ff,7c
165 aX DATA 22,3a,01,60,2c,7a,01,38,4e,ae
166 Hb DATA ff,64,42,81,2c,7a,01,2e,4e,ee
167 dM DATA ff,70,70,01,32,2c,00,12,6b,14
168 bv DATA b2,6c,00,0c,6e,0e,32,2c,00,10
169 eu DATA 6b,08,b2,6c,00,0e,6e,02,42,00
170 k8 DATA 4e,75,22,4d,26,6c,00,50,30,2b
171 U7 DATA 00,08,d0,7a,01,36,32,2b,00,0a
172 cF DATA d2,7a,01,30,2c,7a,00,fe,4e,ae
173 Uv DATA ff,10,22,4d,26,6c,00,50,30,2b
174 9q DATA 00,08,d0,45,32,2b,00,0a,d2,7a
175 xX DATA 01,14,2c,7a,00,e2,4e,ae,ff,0a
176 R8 DATA 22,4d,26,6c,00,50,30,2b,00,08
177 Or DATA d0,45,32,2b,00,0a,d2,46,2c,7a
178 pE DATA 00,c8,4e,ae,ff,0a,22,4d,26,6c
179 uj DATA 00,50,30,2b,00,08,d0,7a,00,e2
180 Ui DATA 32,2b,00,0a,d2,46,2c,7a,00,ac
181 xe DATA 4e,ae,ff,0a,22,4d,26,6c,00,50
182 cC DATA 30,2b,00,08,d0,7a,00,c6,32,2b
183 dO DATA 00,0a,d2,7a,00,c0,2c,7a,00,8e
184 AN DATA 4e,ee,ff,0a,10,28,00,04,0c,00
185 hI DATA 00,01,66,42,30,28,00,08,02,40
186 mT DATA 00,41,0c,40,00,41,66,52,30,28
187 68 DATA 00,06,0c,00,00,21,67,0c,0c,00
188 TU DATA 00,35,67,06,0c,00,00,10,66,38
189 31 DATA 13,c0,00,00,04,f4,60,2c,2f,08
190 ER DATA 22,7a,00,5e,20,3a,00,5e,2c,7a
191 RI DATA 00,32,4e,ae,fe,bc,20,5f,60,16
192 WO DATA 0c,00,00,02,66,14,4a,39,00,00
193 Wn DATA 04,f5,67,0c,0c,68,00,68,00,06
194 bu DATA 66,04,42,28,00,04,20,08,4e,75
195 2j DATA 4a,39,00,00,04,f4,66,c4,60,f2
196 AQ DATA 00,00,00,00,00,00,00,00,00,00
197 BR DATA 00,00,00,00,00,00,00,00,00,00
198 i6 DATA 00,00,00,00,00,00,00,08,00,00
199 DT DATA 00,00,00,00,00,00,00,00,00,00
200 2Q DATA 00,00,00,00,00,00,00,00,ff,ff
201 RR DATA ff,ff,ff,ff,ff,ff,00,00,00,00
202 GW DATA 00,00,00,00,00,00,00,00,00,00
203 HX DATA 00,00,00,00,00,00,00,00,00,00
204 IZ DATA 00,00,00,00,02,7f,00,00,00,00
205 A2 DATA 00,00,00,00,00,00,04,1e,00,00
206 FX DATA 73,61,76,65,69,66,66,2e,20,20
207 dt DATA 20,20,20,20,20,20,20,20,20,20
208 Tp DATA 20,20,20,20,20,20,30,30,2e,69
209 Uk DATA 6c,62,6d,00,69,6e,74,75,69,74
210 qg DATA 69,6f,6e,2e,6c,69,62,72,61,72
211 or DATA 79,00,64,6f,73,2e,6c,69,62,72
212 Ua DATA 61,72,79,00,67,72,61,70,68,69
213 o1 DATA 63,73,2e,6c,69,62,72,61,72,79
214 zJ DATA 00,69,6e,70,75,74,2e,64,65,76
215 3Q DATA 69,63,65,00,53,6e,69,70,49,66
216 eq DATA 66,00,00,00,00,01,00,00,03,ec
217 zF DATA 00,00,00,27,00,00,00,01,00,00
218 r8 DATA 00,1a,00,00,00,32,00,00,00,50
219 KG DATA 00,00,00,5a,00,00,00,76,00,00
220 fb DATA 00,84,00,00,00,9e,00,00,00,b0
221 Ab DATA 00,00,00,de,00,00,00,ec,00,00
222 6o DATA 00,f6,00,00,01,06,00,00,01,1c
223 Ve DATA 00,00,01,30,00,00,01,44,00,00
224 6P DATA 01,52,00,00,01,72,00,00,01,7a
225 QO DATA 00,00,01,a0,00,00,01,a8,00,00
226 r3 DATA 01,b6,00,00,01,bc,00,00,01,d0
227 He DATA 00,00,01,fe,00,00,02,0e,00,00
228 RL DATA 02,20,00,00,02,a4,00,00,02,be
229 9W DATA 00,00,02,f0,00,00,02,fc,00,00
230 EX DATA 03,12,00,00,03,26,00,00,03,2e
231 nj DATA 00,00,03,3e,00,00,03,46,00,00
232 dn DATA 04,4e,00,00,04,72,00,00,04,8a
233 JA DATA 00,00,04,f0,00,00,00,01,00,00
234 KT DATA 00,02,00,00,02,c8,00,00,00,02
235 nr DATA 00,00,00,07,00,00,03,4c,00,00
236 IY DATA 00,0a,00,00,00,02,00,00,00,08
237 fu DATA 00,00,03,34,00,00,00,22,00,00
238 AQ DATA 00,00,00,00,03,f2,00,00,03,e9
239 XI DATA 00,00,01,3f,48,e7,7f,fe,4c,ef
240 1K DATA 03,00,00,3c,4c,ef,00,1f,00,44
241 QI DATA 4e,b9,00,00,00,1c,4c,df,7f,fe
242 Oo DATA 4e,75,23,c8,00,00,04,3e,23,c9
243 mi DATA 00,00,04,7e,33,c0,00,00,04,86
244 Ud DATA 33,c1,00,00,04,88,33,c2,00,00
245 2i DATA 00,14,33,c2,00,00,00,24,33,c3
246 6U DATA 00,00,00,16,33,c3,00,00,00,26
247 mQ DATA 13,c4,00,00,00,1e,42,b9,00,00
248 iR DATA 04,82,20,7a,04,24,33,e8,00,20
249 e5 DATA 00,00,00,32,22,68,00,04,30,29
250 ZY DATA 00,02,48,c0,53,40,45,f9,00,00
251 dc DATA 00,3c,22,69,00,04,42,81,32,19
252 gm DATA 74,02,26,01,e8,4b,02,03,00,f0
253 jE DATA 14,c3,e9,49,51,ca,ff,f2,51,c8
254 xq DATA ff,ea,20,68,00,24,20,68,00,04
255 af DATA 42,80,42,81,32,28,00,00,23,c1
256 Hd DATA 00,00,04,66,30,39,00,00,00,14
257 8E DATA 4a,79,00,00,04,8e,66,06,5e,40
258 C7 DATA e6,48,60,08,06,40,00,0f,e8,48
259 dP DATA e3,48,23,c0,00,00,04,6a,30,3a
260 1F DATA 03,b8,80,fc,00,08,48,40,33,c0
261 Vz DATA 00,00,04,8c,42,82,34,3a,03,a8
262 PS DATA c4,c1,42,80,30,3a,03,9e,e6,48
263 PD DATA d0,42,42,82,14,28,00,05,13,c2
264 jc DATA 00,00,00,1c,53,02,43,fa,03,44
265 3N DATA 45,e8,00,08,22,1a,d2,80,22,c1
266 Om DATA 51,ca,ff,f8,42,a9,00,00,2c,79
267 gc DATA 00,00,04,92,4e,ae,ff,88,2c,79
268 Ob DATA 00,00,04,92,4e,ae,ff,7c,33,fc
269 uR DATA 40,00,00,df,f0,9a,42,b9,00,00
270 wA DATA 00,a0,42,79,00,00,04,8a,42,b9
271 am DATA 00,00,04,6e,45,fa,03,00,22,6a
272 UE DATA 00,00,42,81,42,82,47,f9,00,00
273 PB DATA 00,00,10,19,34,3a,03,36,67,0a
274 Pe DATA e1,48,10,29,00,00,e5,68,e0,48
275 nY DATA 17,80,10,00,52,41,b2,ba,02,fe
276 92 DATA 66,e2,22,4b,41,f9,00,00,00,00
277 OL DATA 4a,39,00,00,00,1e,66,14,20,3a
278 qL DATA 02,e8,22,00,11,b1,10,00,10,00
279 re DATA 51,c9,ff,f8,60,00,00,90,70,01
280 dE DATA 42,39,00,00,04,cc,42,81,42,85
281 kb DATA 42,84,14,31,10,00,4a,01,67,04
282 Nc DATA b6,02,67,1a,16,02,52,04,11,82
283 mE DATA 00,00,52,00,52,01,b2,ba,02,ae
284 x7 DATA 66,e2,53,04,11,84,50,00,60,5a
285 7f DATA 55,04,6a,04,10,05,60,06,11,84
286 bz DATA 50,00,53,00,42,04,14,31,10,00
287 zj DATA b6,02,66,12,52,04,52,01,b2,ba
288 sR DATA 02,84,66,ee,13,fc,00,01,00,00
289 yN DATA 04,cc,44,04,11,84,00,00,11,83
290 FJ DATA 00,01,42,04,54,00,1a,00,52,00
291 dy DATA b2,ba,02,64,67,02,60,a2,53,00
292 az DATA 4a,39,00,00,04,cc,66,0c,11,bc
293 9R DATA 00,00,50,00,11,82,00,01,54,00
294 CD DATA 24,3a,02,42,d5,aa,00,00,45,ea
295 2V DATA 00,04,4a,aa,00,00,66,0a,45,fa
296 Dc DATA 02,0c,52,79,00,00,04,8a,42,82
297 Ag DATA 14,00,5a,82,26,00,48,e7,7f,fe
```

```
298 y5  DATA 20,02,42,81,2c,79,00,00,04,92
299 X2  DATA 4e,ae,ff,3a,4c,df,7f,fe,4a,80
300 i1  DATA 67,00,01,14,4a,b9,00,00,04,6e
301 fp  DATA 66,08,23,c0,00,00,04,6e,60,08
302 Ae  DATA 20,7a,01,fe,21,40,00,00,23,c0
303 YP  DATA 00,00,04,72,20,40,42,a8,00,00
304 rm  DATA 11,43,00,04,d7,b9,00,00,00,a0
305 zs  DATA 41,e8,00,05,43,f9,00,00,00,00
306 C4  DATA 53,03,10,d9,51,cb,ff,fc,34,39
307 KA  DATA 00,00,04,8a,b4,79,00,00,00,16
308 xw  DATA 66,00,fe,94,33,fc,c0,00,00,df
309 iJ  DATA f0,9a,22,3a,01,82,24,3c,00,00
310 Qd  DATA 03,ee,2c,79,00,00,04,9a,4e,ae
311 fn  DATA ff,e2,23,c0,00,00,04,76,4a,80
312 gm  DATA 67,00,00,e0,22,39,00,00,00,a0
313 pt  DATA 06,81,00,00,00,9c,23,c1,00,00
314 LG  DATA 00,04,22,00,26,3c,00,00,00,a4
315 Xe  DATA 24,3c,00,00,00,00,2c,79,00,00
316 4k  DATA 04,9a,4e,ae,ff,d0,20,7a,01,68
317 Tk  DATA 23,e8,00,00,00,00,04,7a,42,83
318 Uh  DATA 16,28,00,04,43,e8,00,05,24,09
319 eG  DATA 22,3a,01,58,2f,03,2f,08,2c,79
320 Wf  DATA 00,00,04,9a,4e,ae,ff,d0,20,5f
321 lo  DATA 26,1f,b6,80,66,00,00,c6,61,00
322 bi  DATA 00,d4,4a,b9,00,00,04,7a,67,08
323 EU  DATA 20,79,00,00,04,7a,60,bc,22,3a
324 5L  DATA 01,28,2c,79,00,00,04,9a,4e,ae
325 Zp  DATA ff,dc,2c,79,00,00,04,92,4e,ae
326 b2  DATA ff,76,2c,79,00,00,04,92,4e,ae
327 kp  DATA ff,82,20,3a,01,12,4e,75,33,fc
328 sH  DATA c0,00,00,df,f0,9a,2c,79,00,00
329 rl  DATA 04,92,4e,ae,ff,76,2c,79,00,00
330 lT  DATA 04,92,4e,ae,ff,82,4a,b9,00,00
331 qj  DATA 04,6e,67,10,20,7a,00,d4,23,e8
332 ao  DATA 00,00,00,00,04,7a,61,00,00,7e
333 jH  DATA 43,fa,00,e6,23,fc,00,00,00,01
334 9v  DATA 00,00,04,82,60,20,20,7a,00,b4
335 A7  DATA 23,e8,00,00,00,00,04,7a,61,00
336 Py  DATA 00,5e,22,7c,00,00,04,a4,23,fc
337 Fp  DATA 00,00,00,02,00,00,04,82,91,c8
338 fD  DATA 45,fa,00,dc,26,4a,42,80,42,81
339 cw  DATA 24,3c,00,00,00,fa,76,3c,2c,79
340 wM  DATA 00,00,04,96,4e,ae,fe,a4,20,3a
341 Bo  DATA 00,8a,4e,75,61,00,00,26,23,fc
342 6v  DATA 00,00,00,03,00,00,04,82,60,00
343 9V  DATA ff,40,42,80,10,28,00,04,5a,00
344 eE  DATA 22,48,2c,79,00,00,04,92,4e,ae
345 IP  DATA ff,2e,4e,75,61,e8,4a,b9,00,00
346 08  DATA 04,7a,66,02,4e,75,20,7a,00,48
347 E7  DATA 23,e8,00,00,00,00,04,7a,60,e6
348 cs  DATA 00,00,00,00,00,00,00,00,00,00
349 dt  DATA 00,00,00,00,00,00,00,00,00,00
350 eu  DATA 00,00,00,00,00,00,00,00,00,00
351 fv  DATA 00,00,00,00,00,00,00,00,00,00
352 gw  DATA 00,00,00,00,00,00,00,00,00,00
353 hx  DATA 00,00,00,00,00,00,00,00,00,00
354 iy  DATA 00,00,00,00,00,00,00,00,00,00
355 jz  DATA 00,00,00,00,00,00,00,00,00,00
356 Rb  DATA 00,01,00,01,00,00,00,28,00,0a
357 yd  DATA 00,00,00,00,00,00,04,cd,00,00
358 EA  DATA 00,00,00,01,00,00,00,1e,00,0a
359 Bt  DATA 00,00,00,00,00,00,04,df,00,00
360 Po  DATA 00,00,00,01,00,00,00,05,00,03
361 1w  DATA 00,00,00,00,00,00,04,f3,00,00
362 uY  DATA 00,00,00,4e,6f,74,20,65,6e,6f
363 iZ  DATA 75,67,68,20,6d,65,6d,6f,72,79
364 uw  DATA 00,55,6e,61,62,6c,65,20,74,6f
365 Qg  DATA 20,6f,70,65,6e,20,66,69,6c,65
366 dC  DATA 00,43,41,4e,43,45,4c,00,4e,71
367 Fn  DATA 00,00,03,ec,00,00,00,0d,00,00
368 0I  DATA 00,01,00,00,03,ee,00,00,02,c6
369 jI  DATA 00,00,02,fc,00,00,03,26,00,00
370 0d  DATA 03,52,00,00,01,14,00,00,01,1e
371 Q6  DATA 00,00,02,50,00,00,03,5c,00,00
372 Vq  DATA 03,66,00,00,03,7e,00,00,03,88
373 JS  DATA 00,00,04,1a,00,00,00,24,00,00
374 hk  DATA 00,02,00,00,00,12,00,00,00,1e
375 de  DATA 00,00,00,24,00,00,00,2a,00,00
376 Je  DATA 00,30,00,00,00,54,00,00,00,a6
377 yz  DATA 00,00,00,b2,00,00,00,c8,00,00
378 sV  DATA 00,d8,00,00,01,36,00,00,01,3c
379 Or  DATA 00,00,01,98,00,00,01,ee,00,00
380 i5  DATA 02,10,00,00,02,3a,00,00,02,64
381 Kr  DATA 00,00,02,6c,00,00,02,7c,00,00
382 eD  DATA 02,a4,00,00,02,d0,00,00,03,0c
383 FB  DATA 00,00,03,3e,00,00,03,46,00,00
384 Zc  DATA 03,92,00,00,03,a0,00,00,03,b2
385 Gh  DATA 00,00,03,c0,00,00,03,ca,00,00
386 Tk  DATA 03,d4,00,00,04,06,00,00,04,28
387 A8  DATA 00,00,04,38,00,00,04,9c,00,00
388 io  DATA 04,b0,00,00,04,c4,00,00,00,10
389 uM  DATA 00,00,00,03,00,00,00,36,00,00
390 bu  DATA 00,3c,00,00,00,42,00,00,00,48
391 he  DATA 00,00,00,4e,00,00,00,60,00,00
392 6v  DATA 00,72,00,00,00,ac,00,00,00,f6
393 ID  DATA 00,00,01,30,00,00,01,7a,00,00
394 g7  DATA 02,8c,00,00,02,aa,00,00,02,dc
395 5v  DATA 00,00,02,e8,00,00,02,f6,00,00
396 ay  DATA 00,02,00,00,00,04,00,00,01,74
397 fU  DATA 00,00,02,96,00,00,00,01,00,00
398 gY  DATA 00,05,00,00,01,4e,00,00,00,00
399 wF  DATA 00,00,03,f2,00,00,03,ea,00,00
400 Kp  DATA 00,29,46,4f,52,4d,00,00,00,00
401 It  DATA 49,4c,42,4d,42,4d,48,44,00,00
402 xy  DATA 00,14,00,00,00,00,00,00,00,00
403 TN  DATA 00,00,00,00,00,00,0a,0b,00,00
404 Zq  DATA 00,00,43,41,4d,47,00,00,00,04
405 Lh  DATA 00,00,00,00,43,4d,41,50,00,00
406 2O  DATA 00,60,00,00,00,00,00,00,00,00
407 Zp  DATA 00,00,00,00,00,00,00,00,00,00
408 aq  DATA 00,00,00,00,00,00,00,00,00,00
409 br  DATA 00,00,00,00,00,00,00,00,00,00
410 cs  DATA 00,00,00,00,00,00,00,00,00,00
411 dt  DATA 00,00,00,00,00,00,00,00,00,00
412 eu  DATA 00,00,00,00,00,00,00,00,00,00
413 fv  DATA 00,00,00,00,00,00,00,00,00,00
414 gw  DATA 00,00,00,00,00,00,00,00,00,00
415 lj  DATA 00,00,00,00,00,00,00,00,42,4f
416 3t  DATA 44,59,00,00,00,00,00,00,03,f2
417 rc  DATA 00,00,03,eb,00,00,00,40,00,00
418 4t  DATA 03,f2,00,00,03,eb,00,00,00,40
419 gL  DATA 00,00,03,f2,00,00,03,e9,00,00
420 G1  DATA 00,05,20,6f,00,04,20,88,58,90
421 4v  DATA 42,a8,00,04,21,48,00,08,4e,75
422 xA  DATA 00,00,00,00,03,f0,00,00,00,02
423 1w  DATA 5f,4e,65,77,4c,69,73,74,00,00
424 an  DATA 00,00,00,00,00,00,00,00,03,f2
425 hj  DATA 00,00,03,e9,00,00,00,37,48,e7
426 ud  DATA 3c,20,26,2f,00,18,14,2f,00,1f
427 yV  DATA 7a,ff,2f,05,4e,b9,00,00,00,44
428 0I  DATA 12,00,70,00,10,01,28,00,72,ff
429 db  DATA b2,80,58,8f,66,04,70,00,60,66
430 7v  DATA 2f,3c,00,01,00,01,48,78,00,22
431 11  DATA 4e,b9,00,00,00,00,24,40,2a,0a
432 W7  DATA 50,8f,66,0e,2f,04,4e,b9,00,00
433 sF  DATA 00,58,70,00,58,8f,60,40,25,43
434 WK  DATA 00,0a,15,42,00,09,15,7c,00,04
435 ps  DATA 00,08,42,2a,00,0e,15,44,00,0f
436 Oz  DATA 42,a7,4e,b9,00,00,00,30,25,40
437 PF  DATA 00,10,4a,83,58,8f,67,0a,2f,0a
438 xv  DATA 4e,b9,00,00,00,6c,60,0a,48,6a
439 uZ  DATA 00,14,4e,b9,00,00,00,00,58,8f
440 3t  DATA 20,0a,4c,df,04,3c,4e,75,2f,0a
441 QG  DATA 24,6f,00,08,4a,aa,00,0a,67,0a
442 I4  DATA 2f,0a,4e,b9,00,00,00,80,58,8f
443 Y7  DATA 15,7c,00,ff,00,08,70,ff,25,40
444 tT  DATA 00,14,70,00,10,2a,00,0f,2f,00
445 Z2  DATA 4e,b9,00,00,00,58,48,78,00,22
446 p9  DATA 2f,0a,4e,b9,00,00,00,18,4f,ef
447 jS  DATA 00,0c,24,5f,4e,75,00,00,00,00
448 76  DATA 03,ec,00,00,00,01,00,00,00,06
449 nn  DATA 00,00,00,88,00,00,00,08,00,00
450 cu  DATA 00,0a,00,00,00,36,00,00,00,12
451 2Y  DATA 00,00,00,6a,00,00,00,c2,00,00
452 XI  DATA 00,46,00,00,00,7c,00,00,00,a6
453 xm  DATA 00,00,00,ce,00,00,00,00,00,00
454 Cp  DATA 03,f0,00,00,00,01,2e,4c,38,00
455 S0  DATA 00,00,00,2a,00,00,00,01,2e,4c
456 2Y  DATA 37,00,00,00,00,50,00,00,00,01
457 JX  DATA 2e,4c,36,00,00,00,00,82,00,00
458 tk  DATA 00,01,2e,4c,31,00,00,00,00,90
459 BK  DATA 00,00,00,01,2e,4c,31,34,00,00
460 oz  DATA 00,ac,00,00,00,01,2e,4c,31,32
461 r9  DATA 00,00,00,8c,00,00,00,03,5f,44
462 fK  DATA 65,6c,65,74,65,50,6f,72,74,00
463 Sw  DATA 00,00,00,96,00,00,00,03,5f,43
464 sP  DATA 72,65,61,74,65,50,6f,72,74,00
465 Vl  DATA 00,00,00,00,00,00,00,00,00,00
466 KZ  DATA 03,f2,00,00,03,e9,00,00,00,09
467 iq  DATA 20,2f,00,04,48,78,00,30,2f,00
468 Ne  DATA 4e,b9,00,00,00,00,50,8f,4e,75
469 5U  DATA 20,2f,00,04,2f,00,4e,b9,00,00
470 0b  DATA 00,42,58,8f,4e,75,00,00,03,ec
471 Fg  DATA 00,00,00,02,00,00,00,09,00,00
472 3r  DATA 00,0c,00,00,00,1c,00,00,00,00
473 3b  DATA 00,00,03,f0,00,00,00,03,5f,44
474 oz  DATA 65,6c,65,74,65,53,74,64,49,4f
475 om  DATA 00,00,00,14,00,00,00,03,5f,43
476 14  DATA 72,65,61,74,65,53,74,64,49,4f
477 hx  DATA 00,00,00,00,00,00,00,00,00,00
478 30  DATA 03,f2,00,00,03,e9,00,00,00,1d
479 up  DATA 48,e7,38,00,24,2f,00,10,26,2f
480 Ov  DATA 00,14,4a,82,66,04,70,00,60,28
481 KQ  DATA 2f,3c,00,01,00,01,2f,03,4e,b9
482 qW  DATA 00,00,00,00,20,40,28,08,50,8f
483 p4  DATA 66,02,60,e4,11,7c,00,05,00,08
484 Qa  DATA 31,43,00,12,21,42,00,0e,20,08
485 qH  DATA 4c,df,00,1c,4e,75,20,6f,00,04
486 iX  DATA 20,08,66,02,60,24,11,7c,00,ff
487 6p  DATA 00,08,70,ff,21,40,00,14,70,ff
488 C0  DATA 21,40,00,18,70,00,30,28,00,12
489 hu  DATA 2f,00,2f,08,4e,b9,00,00,00,18
490 1H  DATA 50,8f,4e,75,00,00,00,00,03,ec
491 9E  DATA 00,00,00,02,00,00,00,0a,00,00
492 iS  DATA 00,1e,00,00,00,6a,00,00,00,00
493 iA  DATA 00,00,03,f0,00,00,00,01,2e,4c
494 c6  DATA 39,00,00,00,00,10,00,00,00,01
495 em  DATA 2e,4c,35,00,00,00,00,14,00,00
496 P1  DATA 00,01,2e,4c,34,00,00,00,00,2c
497 OQ  DATA 00,00,00,01,2e,4c,31,00,00,00
498 i9  DATA 00,3c,00,00,00,01,2e,4c,31,32
499 Qs  DATA 00,00,00,4c,00,00,00,01,2e,4c
500 vS  DATA 31,30,00,00,00,70,00,00,00,03
501 oZ  DATA 5f,44,65,6c,65,74,65,45,78,74
502 gs  DATA 49,4f,00,00,00,42,00,00,00,03
503 nm  DATA 5f,43,72,65,61,74,65,45,78,74
504 FI  DATA 49,4f,00,00,00,00,00,00,00,00
505 4j  DATA 00,00,03,f2,00,00,03,e9,00,00
506 sx  DATA 00,25,2f,0e,2c,79,00,00,04,92
507 7K  DATA 4c,ef,00,03,00,08,4e,ae,ff,3a
508 dF  DATA 2c,5f,4e,75,00,00,2f,0e,2c,79
509 hP  DATA 00,00,04,92,22,6f,00,08,20,2f
510 TA  DATA 00,0c,4e,ae,ff,2e,2c,5f,4e,75
511 MR  DATA 2f,0e,2c,79,00,00,04,92,22,6f
512 gs  DATA 00,08,4e,ae,fe,da,2c,5f,4e,75
513 43  DATA 2f,0e,2c,79,00,00,04,92,20,2f
514 Z2  DATA 00,08,4e,ae,fe,b6,2c,5f,4e,75
515 65  DATA 2f,0e,2c,79,00,00,04,92,20,2f
516 Jg  DATA 00,08,4e,ae,fe,b0,2c,5f,4e,75
517 SX  DATA 2f,0e,2c,79,00,00,04,92,22,6f
518 a9  DATA 00,08,4e,ae,fe,9e,2c,5f,4e,75
519 UZ  DATA 2f,0e,2c,79,00,00,04,92,22,6f
520 RH  DATA 00,08,4e,ae,fe,98,2c,5f,4e,75
521 pe  DATA 00,00,03,ec,00,00,00,07,00,00
522 Qv  DATA 00,01,00,00,00,84,00,00,00,70
523 HE  DATA 00,00,00,5c,00,00,00,48,00,00
524 ur  DATA 00,34,00,00,00,1c,00,00,00,04
525 NJ  DATA 00,00,00,00,00,00,00,03,f0,00,00
526 9r  DATA 00,02,5f,52,65,6d,50,6f,72,74
527 yN  DATA 00,00,00,80,00,00,00,02,5f,41
528 sw  DATA 64,64,50,6f,72,74,00,00,00,6c
529 Po  DATA 00,00,00,03,5f,46,72,65,65,53
530 8z  DATA 69,67,6e,61,6c,00,00,00,00,58
531 TA  DATA 00,00,00,03,5f,41,6c,6c,6f,63
532 zj  DATA 53,69,67,6e,61,6c,00,00,00,44
533 Dv  DATA 00,00,00,03,5f,46,69,6e,64,54
534 Qf  DATA 61,73,6b,00,00,00,00,00,00,30
535 MW  DATA 00,00,00,02,5f,46,72,65,65,4d
536 Gd  DATA 65,6d,00,00,00,18,00,00,00,03
537 yz  DATA 5f,41,6c,6c,6f,63,4d,65,6d,00
538 gw  DATA 00,00,00,00,00,00,00,00,00,00
539 5E  DATA 00,00,03,f2
(C) 1988 M&T
```

Listing 1. (Schluß)

Kreuz & quer

Eine typische Situation: Sie wollen ein älteres oder fremdes Programm verbessern. Es geht aber nicht, da Sie nicht wissen, wo welche Variable verändert wird. Damit ist jetzt dank »XRef« Schluß.

Auch gut dokumentierte Programme sind schwer zu bearbeiten, wenn man zum Beispiel eine falsch gesetzte Variable sucht. XRef hilft Ihnen durch eine Referenzliste der Labels, Variablen, Felder und auch Konstanten. Dies funktioniert mit jedem beliebigen Basic-Programm, das im ASCII-Format vorliegt. Um dies zu erreichen, laden Sie das gewünschte Programm und speichern es folgendermaßen:

```
SAVE "Name",a
```

Für »Name« setzen Sie den Namen des Programms ein. Nun können Sie XRef anwenden. Starten Sie XRef wie gewohnt durch einen Doppelklick auf das Programmsymbol.

Zuerst erscheint die Abfrage nach dem Namen des zu testenden Programms. »Vollqualifizierter Pfadname« bedeutet einen Dateinamen mit allen erforderlichen Angaben. Ein Beispiel:

```
DF1:Basicprogs/Adressbuch
```

Die nächste Frage dreht sich um die numerischen Konstanten. Soll XRef auch diese berücksichtigen, drücken Sie <j>, ansonsten <n>. Es folgt die analoge Frage für Literale. Mit Literale sind die konstanten Zeichenketten gemeint. Also alle Strings, die direkt in Anführungszeichen im Basic-Listing stehen.

So, nun geht es richtig los. XRef testet jetzt alle Zeilen und gibt sie dabei auf dem Bildschirm aus. So sind Sie immer über den Fortgang der Arbeit im Bild. Sollte ein Label doppelt verwendet werden, erkennt XRef dies und gibt eine Meldung aus. Pro Basic-Zeile benötigt XRef eine halbe bis ganze Sekunde. Das bedeutet für ein Listing mit 600 Zeilen eine Zeit von bis zu 10 Minuten.

Der Ausdruck erfolgt mit 72 Zeilen pro Seite und 73 Spalten pro Zeile.

In der abgedruckten Form bearbeitet XRef Listings bis zu 2500 Zeilen. Bei längeren Programmen müssen die String-Felder »v$«, »l$« und »kon$« vergrößert werden. Die Variable »lixb« ändern Sie dann bitte auf 4/5 der Größe von l$.

Noch ein paar Bemerkungen sind nötig: Nach Anweisungen wie PRINT und ähnlichen sollten Sie immer ein Leerzeichen eintragen, da sonst der Befehl nicht erkannt wird. Gleiches gilt für den CALL-Befehl. Aber solche Leerzeichen erhöhen die Lesbarkeit des Programms und sollten sowieso eingegeben werden.

Nachdem Sie jetzt die ausgedruckte Liste vor sich haben, müssen Sie noch wissen, wie sie zu interpretieren ist.

An erster Stelle stehen die Labels (Sprungziele). Es werden sämtliche Sprungziele in der Reihenfolge ihres Auftretens ausgedruckt und erhalten dabei eine fortlaufende Nummer. In der dritten Spalte stehen nun die Nummern der Labels, von denen aus das Sprungziel aufgerufen wird. Um die Übersichtlichkeit zu erhalten, wird hier nicht eine genaue Zeile angegeben. Wenn also zum Beispiel ein Eintrag »2« lautet, erfolgt der Sprung hinter dem Label »2«. Die Suche wird also recht einfach.

Die zweite Tabelle beinhaltet die sortierten Variablen. Die numerischen Veränderlichen tauchen nur auf, wenn Sie am Anfang von XRef auf die zweite Frage <j> eingegeben haben.

Wenn Sie auf die Frage nach den Literalen mit <j> geantwortet haben, druckt XRef noch eine dritte Tabelle aus. Diese enthält die konstanten Zeichenketten. Der Aufbau der Tabelle ist genau wie bei den zwei ersten.

Durch die vielen Informationen, die Ihnen XRef liefert, ist es Ihnen in Zukunft ein leichtes, ältere oder fremde Programme zu verändern.

(Helmut Voelcker/rs)

Programmname:	XRef
Computer:	A500, A1000, A2000 mit Kickstart 1.2
Sprache:	Amiga-Basic 1.2

```
Programm : XRef
 1 y80 'XRef fuer Basic-Programme
 2 MP  'Dimensions und Startvariable
 3 qc2   DEFINT a-z
 4 PW    IF FRE(0)<=25000 THEN CLEAR ,50
         000&
 5 Tw    zs=72
 6 h6    sz=73
 7 JN    ref=30
 8 dU    DIM v$(1000)          'Variablen-T
         abelle
 9 Q7    DIM vbuch(129)
10 xL    DIM l$(1000)          'Label-Tabel
         le
11 YT    lixb=800
12 XY    lixi=lixb
13 eo    DIM kon$(500)
14 Gu    DIM basworte$(200)
15 Mx    DIM basbuch(129)
16 lE0 'Basicworte aus DATA holen
17 WX2   FOR i=0 TO 128:basbuch(i)=-1:NEXT
18 xm    RESTORE basi
19 86    bi=0
20 KH0 basi1:
21 ZZ2   READ basworte$(bi)
22 rX    a$=LEFT$(basworte$(bi),1)
23 8h    IF a$="/" THEN WelchesProgramm
24 A7    IF a$<>"x" AND a$<>"l" THEN
25 D04     PRINT basworte$(bi);" basi1: w
           eder x noch l"
26 Ug      STOP
27 NG2   END IF
28 Vn    a=ASC(MID$(basworte$(bi),2,1))
29 DZ    IF basbuch(a)=-1 THEN basbuch(a)
         =bi
30 a6    bi=bi+1
31 Lr    GOTO basi1
32 Yv0 'Basic-Datei abfragen
33 Tn  WelchesProgramm:
34 CJ2   PRINT " "
 35 JA     PRINT "X-Ref fuer AMIGA-Basic"
36 rK     PRINT "======================"
37 Fm     PRINT " "
38 uG0 WP1:
39 U52    INPUT "Vollqualifizierter Pfadna
          me des Basic-Programms: ",pname$
40 UK     ON ERROR GOTO FalscherProgName
41 zp     OPEN "I",#1,pname$
42 kn     ON ERROR GOTO 0
43 9p     l$(0)=" Noch kein Label":MID$(l$
          (0),1,1)=CHR$(LEN(l$(0))-1)
44 w5     li=1
45 DW     koni=0
46 6b     vi=0:FOR i=1 TO 128:vbuch(i)=-1:
          NEXT
47 Lf     LINE INPUT #1,ln$
48 YD     a=ASC(MID$(ln$,1,1))
49 52     IF a>127 THEN
50 Zw4      PRINT "Dies ist kein Basicprog
            ramm im ASCII-Format."
51 bx       PRINT "Laden Sie dieses Progra
            mm mit AMIGA-Basic"
52 AK       PRINT "und speichern Sie das P
            rogramm erneut mit"
53 qQ       PRINT "SAVE ";CHR$(34);"Progra
            mmname";CHR$(34);",A"
54 TP       FOR i=1 TO 4:BEEP:FOR J=1 TO 2
            000:NEXT:NEXT
55 sL       CLOSE
56 yA       STOP
57 rk2    END IF
58 If0 WP2:
59 h32    INPUT "Xref fuer numerische Kons
          tanten (J/N) ?",numkon$
60 WG     IF LEN(numkon$)<>1 THEN BEEP:G
          OTO WP2
61 7R     numkon$=UCASE$(numkon$)
62 PG     IF numkon$<>"J" AND numkon$<
          >"N" THEN BEEP:GOTO WP2
63 Rp0 WP3:
64 Lr2    INPUT "Xref fuer Literale  (J/N)
           ?",litkon$
65 EJ     IF LEN(litkon$)<>1 THEN BEEP:G
          OTO WP3
66 ou     litkon$=UCASE$(litkon$)
67 Eq     IF litkon$<>"J" AND litkon$<
          >"N" THEN BEEP:GOTO WP3
68 Y3     GOTO Lesen1
69 SD0 FalscherProgName:
70 i02    PRINT "Geraete- oder Dateifehler
          : ";ERR
71 DG     ON ERROR GOTO 0
72 k5     RESUME WP1
73 2f0 'Satz lesen und untersuchen
74 Gv  Lesen:
75 n72    LINE INPUT #1,ln$
76 GC0 Lesen1:
77 CU2    IF EOF(1) GOTO LesenEnde
78 ep     PRINT ln$
79 jo     liend=LEN(ln$)
80 8h     IF liend<3 THEN Lesen
81 d1     ongotoda=0
82 FH     lni=1
83 aD     wort$=""
84 n7     ersteswort=0
85 He0 'Pruefung ob Trenner
86 cp  Lesenzeichen:
87 Pk2    lz$=MID$(ln$,lni,1)
88 Dm     IF lz$="'" THEN Wortda
89 2Q     IF lz$=" " OR lz$="," OR lz$=";"
           THEN Wortda
90 ZA     IF lz$="(" OR lz$=")" THEN Wortda
91 Ps     IF lz$="-" OR lz$="*" OR lz$="+"
           THEN Wortda
92 4A     IF lz$="/" OR lz$="\" OR lz$="^"
           THEN Wortda
93 5x     IF lz$="=" OR lz$="<" OR lz$="
          >" THEN Wortda
```

Listing. Eine komplette Referenzliste für Basic-Programme erhalten Sie von »XRef«. Bitte mit dem Checksummer (Seite 159) eingeben.

```
 94 JW    IF lz$=CHR$(34) THEN GOSUB Konda
              'Gaensefuesschen
 95 LU    IF lz$=":" THEN Labda
 96 pd    wort$=wort$+lz$
 97 yX0 LesenNae:
 98 LS2   lni=lni+1
 99 iP    IF lni<=liend THEN Lesenzeichen
100 5S    IF wort$="" THEN Lesen
101 fy0 'Wort gefunden ?
102 FA  Wortda:
103 hE2   IF wort$="" THEN
104 Ub4     IF lz$="'" THEN Lesen ELSE Les
            enNae
105 dW2   END IF
106 CX    ersteswort=1
107 D5    GOSUB WortBas
108 am0 Wortda1:
109 6k2   IF wb$="J" THEN
110 4A4     IF wort$="DATA" THEN Lesen
111 WP      IF wort$="REM" THEN Lesen
112 mK      IF wort$="ON" THEN ongotoda=1
113 41      IF (wort$="GOTO" OR wort$="GOS
            UB") AND ongotoda=1 THEN ongot
            oda=2
114 gT      vorigwort$=wort$
115 6j      wort$=""
116 oI      vorigwbt$=wbt$
117 s0      GOTO LesenNae
118 VE2   ELSE
119 Id4     IF vorigwbt$="1" OR ongotoda=2
             THEN
120 NG6       IF lz$="," OR lz$=" " OR lz$
              =":" OR lni>liend THEN
121 Ej8         labneu$="X"
122 tM          IF LEN(vorigwort$)=3 THEN
123 OJA           IF vorigwort$="SUB" THEN
                   labneu$="V"
124 wp8         END IF
125 nD          GOSUB LabEintr
126 Ed          IF lz$=":" THEN ongotoda=0
127 R6          vorigwbt$=" "
128 Jw          wort$=""
129 4C          GOTO LesenNae
130 2v6       END IF
131 3w4     END IF
132 4x2   END IF
133 XC    vorigwbt$=" "
134 V0    GOSUB VariEintr
135 Q3    wort$=""
136 BJ    GOTO LesenNae
137 FR    BEEP:ON x GOSUB Labda, Konda, Le
          senEnde : GOTO LesenNae
138 fQ0 'Label gefunden ?
139 gp  Labda:
140 LF2   IF wort$="" THEN Wortda
141 Ws    IF ersteswort>0 THEN Wortda
142 me    GOSUB WortBas
143 ai    IF wb$="J" THEN Wortda1
144 P2    labneu$="V"
145 7X    GOSUB LabEintr
146 bE    wort$=""
147 MU    GOTO LesenNae
148 Iq0 'Literal Eintragen ?
149 Go  Konda:
150 Vn2   lit$=CHR$(34)
151 G50 Konda1:
152 DK2   lni=lni+1
153 fM    IF lni>liend THEN RETURN
154 Up    lz$=MID$(ln$,lni,1)
155 th    lit$=lit$+lz$
156 UW    IF lz$<>CHR$(34) THEN Konda1
157 EZ    IF litkon$="N" THEN RETURN
158 6w    IF LEN(lit$)>24 THEN lit$=MID$(
          lit$,1,23)+CHR$(34)
159 fI    kl=0:IF kl=koni THEN KonEintr
160 U10 KonSuch:
161 Ct2   lw=LEN(lit$):lv=ASC(MID$(kon$(kl
          ),1,1))
162 X7    IF lw<lv THEN
163 G54     IF lit$<=MID$(kon$(kl),2,lw)
            THEN KonEinf
164 nf      GOTO KonNae
165 bU2   END IF
166 dE    IF lw=lv THEN
167 pe4     IF lit$<MID$(kon$(kl),2,lw) T
            HEN KonEinf
168 4B      IF lit$=MID$(kon$(kl),2,lw) TH
            EN kon$(kl)=kon$(kl)+MKI$(li-1
            ):RETURN
169 sk      GOTO KonNae
170 gZ2   END IF
171 Q4    IF MID$(lit$,1,lv)<MID$(kon$(kl
          ),2,lv) THEN KonEinf
172 Vo0 KonNae:
173 LO2   kl=kl+1
174 ve    IF kl<koni THEN KonSuch
175 5Y0 KonEintr:
176 lw2   kon$(kl)=CHR$(LEN(lit$))+lit$+MK
          I$(li-1)
177 b8    koni=koni+1
178 Eq    RETURN
179 lg0 KonEinf:
180 a22   FOR i=koni TO (kl+1) STEP -1:kon
          $(i)=kon$(i-1):NEXT i
181 ln    GOTO KonEintr
182 bE0 LesenEnde:
183 wP2   CLOSE
184 vn    GOTO Drucken
185 SE0 'Pruefen ob Basic-Wort
186 kC  WortBas:
187 O22   wb$="N"
188 ht    a$=MID$(wort$,1,1)
189 6j    IF a$>="0" AND a$<="9" THEN RE
          TURN
190 yz    bl=basbuch(ASC(a$)):IF bl=-1 THE
          N RETURN
191 L40 WortBas1:
192 Uu2   lbas=LEN(basworte$(bl))
193 PY    IF LEN(wort$)<>(lbas-1) THEN W
          ortBasN
194 aP    IF wort$<MID$(basworte$(bl),2,l
          bas-1) THEN RETURN
195 iM    IF wort$>MID$(basworte$(bl),2,l
          bas-1) THEN WortBasN
196 YY    wb$="J":wbt$=MID$(basworte$(bl),
          1,1)
197 X9    RETURN
198 KW0 WortBasN:
199 k72   bl=bl+1
200 ks    IF bl<bi THEN WortBas1
201 bD    RETURN
202 Vm0 'Variable eintragen
203 b4  VariEintr:
204 gU2   worw$=MID$(wort$,1,1)
205 oA    IF worw$=CHR$(34) THEN wort$="":
          RETURN
206 Aa    IF numkon$="N" AND worw$>="0" A
          ND worw$<="9" THEN wort$="":RET
          URN
207 r2    vari$=CHR$(LEN(wort$))+wort$
208 SY    IF vi=0 THEN v$(vi)=vari$+MKI$(1
          i-1):vi=1:RETURN
209 zK    vl=0
210 3U    w=ASC(MID$(wort$,1,1))
211 Va    IF vbuch(w)<>-1 THEN vl=vbuch(
          w)
212 iB0 VariEintr1:
213 Nc2   lw=LEN(wort$)
214 4t    lv=ASC(MID$(v$(vl),1,1))
215 2H    IF lw<=lv THEN
216 JP4     varw$=MID$(v$(vl),2,lw)
217 U6      IF wort$=varw$ AND lw=lv THEN
            VariEiErg
218 zr      IF wort$>varw$ THEN VariEiN
219 9m      GOTO VariEiZuf
220 UN2   END IF
221 ay    worw$=MID$(wort$,1,lv)
222 MR    varw$=MID$(v$(vl),2,lv)
223 58    IF worw$>=varw$ THEN VariEiN
224 420 VariEiZuf:
225 LD2   FOR i=vi TO vl+1 STEP -1:v$(i)=v
          $(i-1):NEXT
226 420 VariEiZuf1:
227 x92   v$(vl)=vari$+MKI$(li-1)
228 2C    vi=vi+1
229 D70 VariEiZuf2:
230 7u2   FOR vlx=0 TO 128:vbuch(vlx)=-1:N
          EXT
231 qb    FOR vlx=0 TO vi-1
232 40    lv=ASC(MID$(v$(vlx),2,1))
233 SF    IF vbuch(lv)=-1 THEN vbuch(lv)=v
          lx
234 z6    NEXT vlx
235 91    RETURN
236 8j0 VariEiErg:
237 4k2   vari$=v$(vl)+MKI$(li-1)
238 ae    v$(vl)=vari$
239 Dp    RETURN
240 m90 VariEiN:
241 Oa2   vl=vl+1:IF vl<vi THEN VariEintr1
242 Ah    GOTO VariEiZuf1
243 bF0 'Label eintragen
244 mq  LabEintr:
245 1e2   ll=0:lim=li
246 bq0 LabEintr1:
247 Ra2   lwl=LEN(wort$)
248 xZ    lll=ASC(MID$(l$(ll),1,1))
249 HF    IF lwl<>lll THEN LabEin
250 p5    IF wort$=MID$(l$(ll),2,lll) THEN
           LabEiErg
251 h50 LabEin:
252 Vd2   ll=ll+1:IF ll<lim THEN LabEintr1
253 D3    IF ll<lixb THEN GOTO LabEiLix
254 x60 LabEiN1:
255 I02   l$(li)=CHR$(LEN(wort$))+wort$
256 lc    IF labneu$<>"V" THEN l$(li)=l$
          (li)+MKI$(li-1)
257 O1    wort$=""
258 OE    li=li+1
259 5j    GOTO LabEiEnd
260 u60 LabEiErg:
261 mr2   IF labneu$="V" THEN
262 pS4     IF ll=(li-1) THEN PRINT :PRINT
             "---DOPPELTES LABEL---":PRINT
             :RETURN
263 3n      labnae$=l$(ll)
264 v0      FOR i=ll TO li-2:l$(i)=l$(i+1)
            :NEXT
265 Wg      l$(li-1)=labnae$
266 Cq      GOTO LabEiEnd
267 ud2   ELSE
268 sU4     PRINT :PRINT"DOPPELT":PRINT
269 HA2   END IF
270 5X    l$(ll)=l$(ll)+MKI$(li-1)
271 Hv    GOTO LabEiEnd
272 qj0 'Spaetere Labels hinten eintr.
273 0R  LabEiLix:
274 jB2   IF lixi=lixb THEN
275 re4     IF labneu$="V" THEN GOTO LabEi
            N1
276 TC      l$(lixi)=CHR$(LEN(wort$))+wort
            $+MKI$(li-1)
277 zg      lixi=lixi+1
278 O2      GOTO LabEiEnd
279 RK2   END IF
280 kf    ll=lixb
281 vX0 LabEiLix1:
282 092   lwl=LEN(wort$)
283 W8    lll=ASC(MID$(l$(ll),1,1))
284 XC    IF lwl<>lll THEN LabEiLixN
285 Rj    IF wort$=MID$(l$(ll),2,lll) THEN
           LabEiLixErg
286 LQ0 LabEiLixN:
287 L92   ll=ll+1:IF ll<lixi THEN LabEiLi
          x1
288 OC    IF labneu$="V" THEN LabEiN1
289 gP    l$(lixi)=CHR$(LEN(wort$))+wort$+
          MKI$(li-1)
290 vY    wort$=""
291 Du    lixi=lixi+1
292 cG    GOTO LabEiEnd
293 ha0 LabEiLixErg:
294 3B2   IF labneu$<>"V" THEN l$(ll)=l$
          (ll)+MKI$(li-1):GOTO LabEiEnd
295 WR    l$(li)=l$(ll)
296 0q    li=li+1
297 rv    IF ll=(lixi-1) THEN lixi=lixi-1:
          GOTO LabEiEnd
```

Listing. Eine komplette Referenzliste für Basic-Programme erhalten Sie von »XRef« (Fortsetzung)

```
298 XO    FOR lwl=ll TO lixi-1:l$(lwl)=l$(
          lwl+1):NEXT
299 X6    lixi=lixi-1
300 AFO LabEiEnd:
301 6j2   wort$=""
302 Eq    RETURN
303 SP    FOR tll=0 TO li-1    'Testausdru
          ck
304 lJ    lw=ASC(MID$(l$(tll),1,1))
305 CD    PRINT "    ";tll;" ";MID$(l$(tll
          ),2,lw)
306 JC    NEXT tll
307 H1    IF lixi=lixb THEN LabEiEnd1
308 af    FOR tll=lixb TO lixi-1
309 q0    lw=ASC(MID$(l$(tll),1,1))
310 HI    PRINT "    ";tll;" ";MID$(l$(tll
          ),2,lw)
311 OH    NEXT tll
312 9P0 LabEiEnd1:
313 ci2   INPUT "    weiter",w$
314 Q2    RETURN
315 T70 'Drucken
316 3M  Drucken:
317 222   OPEN "O", #2, "PRT:"
318 T7    seite=0:zeile=zs+1
319 vo    ueb2$="Nr.| Label"+SPACE$(sz-10)
320 WK    MID$(ueb2$,ref-1,15)="| aufgerufe
          n von"
321 ee    ueb3$=STRING$(sz,"-")
322 ao    MID$(ueb3$,4,1)="+"
323 Wo    MID$(ueb3$,ref-1,1)="+"
324 3x    GOSUB DruUeb
325 Xi    ll=0
326 dE0 'Labels drucken
327 7O  DruLab:
328 8F2   zeile$=SPACE$(sz)
329 Hb    lae=ASC(MID$(l$(ll),1,1))
330 B6    zpos=1:N=ll: GOSUB Num4
331 fE    MID$(zeile$,zpos,lae)=MID$(l$(ll
          ),2,lae)
332 Mg    lref=lae+2
333 Av    IF lref>LEN(l$(ll)) THEN DruNLab
334 T10 DruLab1:
335 kL2   MID$(zeile$,ref-1,1)="| "
336 IL    zpos=ref
337 Z80 DruLab2:
338 ST2   N=CVI(MID$(l$(ll),lref,2)):GOSUB
           Num4
339 xF    lref=lref+2
340 H2    IF lref>LEN(l$(ll)) THEN DruNLab
341 UZ    IF (sz+1-zpos)<4 THEN GOSUB Dru
          Zeile:GOTO DruLab1
342 em    GOTO DruLab2
343 5n0 DruNLab:
344 fh2   GOSUB DruZeile
345 4e    ll=ll+1:IF ll<li THEN DruLab
346 Vw    IF vi=0 THEN DruEnde
347 Ns0 'Variable drucken
348 Kz2   MID$(ueb2$,5,17)="Variable sorti
          ert"
349 hE    IF zeile>(zs-17) THEN
350 TN4     GOSUB DruUeb
351 Gz2   ELSE
352 dh4     PRINT #2,"  "
353 ei      PRINT #2,"  "
354 Tg      PRINT #2,ueb2$
355 Ym      PRINT #2,ueb3$
356 Xy      zeile=zeile+4
357 ha2   END IF
358 Oj    vl=0
359 xe0 DruVar:
360 el2   zeile$=SPACE$(sz)
361 5j    lae=ASC(MID$(v$(vl),1,1))
362 fw    zpos=1:N=vl: GOSUB Num4
363 92    MID$(zeile$,zpos,lae)=MID$(v$(vl
          ),2,lae)
364 sC    lref=lae+2
365 O5    IF lref>LEN(v$(vl)) THEN DruNVar
366 JH0 DruVar1:
367 Gr2   MID$(zeile$,ref-1,1)="| "
368 or    zpos=ref
369 P00 DruVar2:
370 Qx2   N=CVI(MID$(v$(vl),lref,2)):GOSUB
           Num4
371 T1    lref=lref+2
372 VC    IF lref>LEN(v$(vl)) THEN DruNVar
373 cX    IF (sz+1-zpos)<4 THEN:GOSUB Dru
          Zeile:GOTO DruVar1
374 U2    GOTO DruVar2
375 Lf0 DruNVar:
376 BD2   GOSUB DruZeile
377 WM    vl=vl+1:IF vl<vi THEN DruVar
378 BC0 'Literale drucken
379 a42   IF litkon$<>"J" OR koni=0 THEN
           DruEnde
380 hu    MID$(ueb2$,5,17)="Literale sorti
          ert"
381 Dk    IF zeile>(zs-17) THEN
382 zt4     GOSUB DruUeb
383 mV2   ELSE
384 9D4     PRINT #2,"  "
385 AE      PRINT #2,"  "
386 zC      PRINT #2,ueb2$
387 4I      PRINT #2,ueb3$
388 3U      zeile=zeile+4
389 D62   END IF
390 uF    vl=0
391 oM0 DruKon:
392 AH2   zeile$=SPACE$(sz)
393 lu    lae=ASC(MID$(kon$(vl),1,1))
394 BS    zpos=1:N=vl: GOSUB Num4
395 WK    MID$(zeile$,zpos,lae)=MID$(kon$(
          vl),2,lae)
396 0i    lref=lae+2
397 3E    IF lref>LEN(kon$(vl)) THEN DruN
          Kon
398 Az0 DruKon1:
399 mN2   MID$(zeile$,ref-1,1)="| "
400 KN    zpos=ref
401 G60 DruKon2:
402 f02   N=CVI(MID$(kon$(vl),lref,2)):GOS
          UB Num4
403 zH    lref=lref+2
404 AL    IF lref>LEN(kon$(vl)) THEN DruN
          Kon
405 C6    IF (sz+1-zpos)<4 THEN:GOSUB Dru
          Zeile:GOTO DruKon1
406 Lk    GOTO DruKon2
407 3F0 DruNKon:
408 hj2   GOSUB DruZeile
409 gC    vl=vl+1:IF vl<koni THEN DruKon
410 bN0 'Programmende
411 zS  DruEnde:
412 d62   CLOSE
413 PK    END
414 Kx0 'Zeile Drucken
415 Fw  DruZeile:
416 Eg2   IF zeile>(zs-7) THEN GOSUB DruU
          eb
417 gP    PRINT #2,zeile$
418 ah    zeile$=SPACE$(sz)
419 Mk    zeile=zeile+1
420 8k    RETURN
421 mN0 'Seitenueberschrift
422 lv  DruUeb:
423 Nb2   IF zeile<=zs THEN PRINT #2,"
          ":zeile=zeile+1:GOTO DruUeb
424 I4    seite=seite+1
425 pH    ueb$=SPACE$(sz)
426 gC    sei$=STR$(seite)
427 Mq    MID$(ueb$,sz-4,LEN(sei$))=sei$
428 zZ    d$=DATE$
429 gK    MID$(ueb$,sz-17,10)=MID$(d$,4,2)
          +"."+MID$(d$,1,2)+"."+MID$(d$,7,
          4)
430 1b    MID$(ueb$,1,10)="Xref fuer "
431 93    MID$(ueb$,11,LEN(pname$))=pname$
432 VQ    PRINT #2,ueb$
433 y2    PRINT #2,"  "
434 1y    PRINT #2,ueb2$
435 q4    PRINT #2,ueb3$
436 k4    zeile=5
437 P1    RETURN
438 qj0 '4-stellige Nummer in Zeile
439 mm  Num4:
440 Fs2   N$=STR$(N)+"| "
441 Om    IF LEN(N$)>4 THEN N$=MID$(N$,2,
          4)
442 q1    IF LEN(N$)<4 THEN N$=" "+N$
443 80    MID$(zeile$,zpos,4)=N$
444 oG    zpos=zpos+4
445 X9    RETURN
446 Aw0 'Reservierte Basic-Worte
447 FC  basi:
448 3k2   DATA "xABS", "xALL", "xAND", "xA
          PPEND", "xAREA", "xAREAFILL"
449 Iw    DATA "xAS", "xASC" , "xATN", "xB
          ASE", "xBEEP", "xBREAK"
450 ci    DATA "lCALL", "xCDBL", "xCHAIN",
          "xCHDIR", "xCHR$", "xCINT"
451 bc    DATA "xCIRCLE", "xCLEAR", "xCLNG
          ", "xCLOSE", "xCLS"452 xp    DATA
          "xCOLLISION", "xCOLOR", "xC
          OMMON", "xCONT", "xCOS"
453 OG    DATA "xCSNG", "xCSRLIN", "xCVD",
           "xCVI", "xCVL", "xCVS"
454 Rc    DATA "xDATA", "xDECLARE", "xDEF"
          , "xDEFBDL", "xDEFINT"
455 it    DATA "xDEFLNG", "xDEFSNG", "xDEF
          STR", "xDELETE", "xDIM", "lELSE"
456 eJ    DATA "xELSEIF", "xEND", "xEOF",
          "xEQV", "xERASE", "xERL"
457 FU    DATA "xERR", "xERROR", "xEXIT",
          "xEXP", "xFIELD", "xFILES"
458 Hw    DATA "xFIX", "xFN" , "xFOR", "xF
          RE", "xFUNCTION", "xGET"
459 Xy    DATA "lGOSUB", "lGOTO", "xHEX$",
          "xIF", "xIMP", "xINKEY$"
460 Gt    DATA "xINPUT", "xINPUT$", "xINST
          R", "xINT", "xKILL", "xLBOUND"
461 Lt    DATA "xLEFT$", "xLEN", "xLET", "
          xLIBRARY", "xLINE", "xLIST"
462 Yl    DATA "xLLIST", "xLOAD", "xLOC",
          "xLOCATE", "xLOF", "xLOG"
463 er    DATA "xLPOS", "xLPRINT", "xLSET"
          , "xMENU", "xMERGE", "xMID$"
464 S2    DATA "xMKD$", "xMKI$", "xMKL$",
          "xMKS$", "xMOD", "xMOUSE"
465 5p    DATA "xNAME", "xNEW", "xNEXT", "
          xNOT", "xOBJECT.AX", "xOBJECT.AY"
466 QQ    DATA "xOBJECT.CLIP", "xOBJECT.CL
          OSE", "xOBJECT.HIT", "xOBJECT.OF
          F"
467 1F    DATA "xOBJECT.ON", "xOBJRECT.PLA
          NES", "xOBJECT.PRIORITY"
468 wH    DATA "xOBJECT.SHAPE", "xOBJECT.S
          TART", "xOBJECT.STOP"
469 GF    DATA "xOBJECT.VX", "xOBJECT.VY",
          "xOBJECT.X", "xOBJECT.Y"
470 MA    DATA "xOCT$", "xOFF", "xON", "xO
          PEN", "xOPTION", "xOR"
471 40    DATA "xOUTPUT", "xPAINT", "xPALE
          TTE", "xPATTERN", "xPEEK"
472 rw    DATA "xPEEKL", "xPEEKW", "xPOINT
          ", "xPOKE", "xPOKEL", "xPOKEW"
473 5h    DATA "xPOS", "xPRESET", "xPRINT"
          , "xPSET", "xPTAB", "xPUT"
474 XC    DATA "xRANDOMIZE", "xREAD", "xRE
          M", "lRESTORE", "lRESUME"
475 3X    DATA "xRETURN", "xRIGHT$", "xRND
          ", "xRSET", "lRUN", "xSADD"
476 hL    DATA "xSAVE", "xSAY", "xSCREEN",
          "xSCROLL", "xSGN", "xSHARED"
477 vV    DATA "xSIN", "xSLEEP", "xSOUND",
          "xSPACE$", "xSPC", "xSQR"
478 FJ    DATA "xSTATIC", "xSTEP", "xSTICK
          ", "xSTOP", "xSTR$", "xSTRIG"
479 Z8    DATA "xSTRING$", "lSUB", "xSWAP"
          , "xSYSTEM", "xTAB", "xTAN"
480 Ms    DATA "lTHEN", "xTIMER", "xTO", "
          xTRANSLATE$", "xTROFF", "xTRON"
481 7Q    DATA "xUBOUND", "xUCASE$", "xUSI
          NG", "xVAL", "xVARPTR", "xWAIT"
482 c1    DATA "xWAVE", "xWEND", "xWHILE",
          "xWIDTH", "xWINDOW", "xWRITE"
483 Cb    DATA "xXOR"
484 4N    DATA "////"
(C) 1988 M&T
```

Listing. Eine komplette Referenzliste für Basic-Programme erhalten Sie von »XRef« (Schluß)

Die Schatzkiste für Basic-Freaks

Eine Sammlung der besten Basic-Tips aus dem AMIGA-Magazin haben wir für Sie zusammengestellt. Endlich hat das zeitraubende Suchen in verschiedenen Heften ein Ende. Die folgenden Seiten bieten eine unentbehrliche Hilfe für alle Basic-Programmierer.

Beschleunigter A/C-Compiler

Der A/C-Basic-Compiler übersetzt Amiga-Basic-Programme in schnellen Maschinencode (siehe auch Artikel »Compiler contra Interpreter« in diesem Sonderheft). Doch auch der Compiler selbst läßt sich beschleunigen.

```
MAKEDIR ram:
COPY source to ram:
CD ram:
df0:AC-Basic
```

Durch diese einfache Befehlsfolge erreichen Sie eine erhebliche Steigerung der Compiliergeschwindigkeit. *(Tilo Renkl/rs)*

Laufwerk: Eine brennende Sache

Nachdem in der AMIGA 2/88, Seite 93, das Ein- und Ausschalten der Power-LED gezeigt wurde, ist nun die Leuchtdiode des Laufwerks dran. Das Aufleuchten erreichen Assemblerprogrammierer mit diesen Befehlen:

```
LED: EQU $BFD100
     move.b #127,LED
     move.b #119, LED
     move.b #0, LED +512
```

Auch wer in Basic programmiert, kann die LED aufleuchten lassen:

```
LED = 12570880&
POKE LED, 127
POKE LED, 119
POKE LED + 512,0
```

Um die LED wieder auszuschalten, braucht man lediglich die letzte 0 mit dem Wert 255 auszutauschen. Das kleine Programm kann jeder gut brauchen, der zum Beispiel dem Anwender zeigen möchte, daß er die Finger vom Laufwerk lassen soll. So verhindern Sie, daß jemand eine Diskette zu früh aus dem Laufwerk nimmt. Entwarnung ist erst, wenn die Lampe erlischt. *(Armin Hegglin/rs)*

Tastaturabfrage in Basic

Folgendes Unterprogramm zeigt, wie Sie eine Tastaturabfrage in Basic realisieren:

```
Sub Taste (x$,z$)STATIC
 i=0:y$=""
WHILE y$="" OR i=0
 y$=UCASE$ (INKEY$)
i=INSTR(x$,y$)
WEND
END SUB
```

Im String »x$« stehen die ASCII-Zeichen der für die Abfrage vorgesehenen Tasten. In »y$« wird der Code der tatsächlich gedrückten Taste zurückgegeben. Mit der INSTR-Anweisung prüft das Programm, ob das Ergebnis der Tastaturabfrage in »x$« enthalten ist. Ist dies der Fall, wird die Routine verlassen. Das heißt die Schleife wird erst dann beendet, wenn eine Taste gedrückt wird, die auch noch in das vorgegebene Muster paßt.

Der Aufruf für eine Ja/Nein-Abfrage lautet:

```
CALL Taste("JN",a$)
```

Diese Schleife reagiert nur auf »j« und »n«. Das Ergebnis steht schließlich in a$.

Ähnlich können Sie auch die Cursortasten testen:

```
CALL Taste(CHR$(28)+CHR$(29))
```

Sie können also Tasten selektiv abfragen. *(Helmut Künne/rs)*

PRINT und LPRINT

Wer versucht, in Amiga-Basic nach der Verwendung des Befehls LPRINT den Drucker über den Parallel-Port mit druckerspezifischen Codes zu füttern, erhält die Fehlermeldung: »File all ready open«. Da hilft auch keine CLOSE-Anweisung hinter dem LPRINT-Statement.

Eine Lösung: Geben Sie die Befehle in der folgenden Reihenfolge ein:

```
LPRINT "text"
OPEN "LPT1:" FOR OUTPUT
AS 1
CLOSE 1
OPEN "PAR:" FOR OUTPUT AS 1
PRINT #1, Steuercodes
```

Erst nach der zusätzlichen OPEN-Anweisung akzeptiert der Amiga »CLOSE 1«. Danach ist der Weg über die parallele Schnittstelle im wahrsten Sinne des Wortes frei. *(Georg Roth/rs)*

Bilder laden

Das Programm »LoadILBM-SaveACBM« auf der Extras-Diskette dient zum Laden von IFF-Bildern in Basic-Programmen. Allerdings hat das Programm einen kleinen Schönheitsfehler: Es berücksichtigt die PAL-Auflösung mit 256 Zeilen nicht. Bilder von Deluxe Paint II mit 200 bis 256 Zeilen werden als Interlace-Bild geladen. Man kann LoadILBM-SaveACBM leicht an die höhere PAL-Auflösung anpassen. Laden Sie das Programm und aktivieren das List-Fenster. Bei gleichzeitig gehaltener Shift-Taste drücken Sie nun dreimal auf die Cursortaste mit dem Pfeil nach unten. Folgendes sollte nach Ausführung der Anweisung in der obersten Zeile stehen:

```
IF scrHeight% > 200
     THEN kk=kk+2
```

Die Zahl 200 bezieht sich auf die NTSC-Norm. Dieser Wert sollte durch 256 ersetzt werden. Denken Sie nach der Änderung daran, das neue Programm wieder auf der Diskette zu speichern. *(Dieter Sonnenberg/rs)*

Basic-Patch für andere Versionen

Das große Ausgabefenster für Amiga-Basic ist der Hit (siehe Patch im Sonderheft 1, Seite 143 und AMIGA 4/88, Seite 108). Für die ältere Basic-Version 1.00 mit der Länge von 93156 Byte sind folgende Änderungen im Patch-Programm notwendig:

— Die Zahl &H57EC muß durch &H55E8,
— die Zahl &H5C00 durch &H2800 ersetzt werden.

Mit diesen kleinen Veränderungen können Sie dann auch die älteren Basic-Versionen patchen, um in den Genuß des großen Ausgabefensters zu kommen. *(Alexander Wagner/rs)*

Spiel mit der Maus kostet Zeit

Vor allem bei zeitaufwendigen Programmen kommt es oft vor, daß man aus Langeweile die Maus bewegt. Geht es Ihnen auch oft so? Das führt jedoch zu einer Verlangsamung des Programms. Als Beispiel dient eine FOR-NEXT-Schleife in Basic:

```
FOR i= 1 TO 1000 : NEXT i
```

Im Normalfall benötigt die Schleife etwa 4 Sekunden. Wenn Sie nun ungeduldig mit der Maus spielen, kann sich die Ablaufzeit um bis zu 0,6 Sekunden erhöhen. *(Torsten Kerschat/rs)*

List-Fenster: Keine halben Sachen

Das Ausgabefenster des Amiga-Basic ist mit dem Trick aus der AMIGA 4/88, Seite 108, bereits vergrößert worden. Eine feine Sache, aber was ist mit dem List-Fenster? Können nicht beide Fenster gleich die volle PAL-Auflösung unterstützen? Mit dem richtigen Patch geht auch dies:

Als erstes kopiert man die ExtrasD-Diskette mit dem Amiga-Basic (Version vom 27.4.87). Zum Patchen auf der Kopie benötigen Sie einen Diskettenmonitor (z. B. »Smart Disk«). Laden Sie zunächst den Block 0835. In der Zeile 070 stehen die Daten für das List-Fenster.

```
 0.   1.   2.   3.   4.   5.   6.   7.
4E75 000C 0124 00B8 0264 48E7 C0D0 2E2A
```

Die Werte in der ersten und zweiten Spalte entsprechen der linken oberen Koordinate. In der dritten und vierten Reihe stehen die Angaben für die rechte untere Ecke des List-Fensters. In ASCII-Schreibweise sieht die Zeile wie folgt aus:

```
Nu...$...dHc...*
```

Ändern Sie die Angaben in den ersten vier Reihen nun wie folgt:

```
 0.   1.   2.   3.   4.   5.   6.   7.
4E75 0000 0000 00F0 0268 48E7 C0D0 2E2A
```

Nach dieser Änderung wird der Editor abgeschaltet, die neue Checksumme berechnet und übertragen. Schreiben Sie den geänderten Block zurück auf die Diskette.

Das neue List-Fenster im PAL-Format funktioniert natürlich nur, wenn auch das Ausgabefenster, ähnlich wie in der AMIGA 4/88 beschrieben, vergrößert wird. Die Daten finden sich im gleichen Block 835, in der Zeile 050:

```
000E 4E75 2F0A 303C 0280 323C 00C8 0C43
```

Ändern Sie auch diese Zeile wie folgt ab:

```
000E 4E75 2F0A 303C 0280 323C 0100 0C43
```

Mit beiden Änderungen hat nun jeder Programmierer ein vollständig an den PAL-Amiga angepaßtes Amiga-Basic. Da macht das Programmieren gleich noch mal soviel Spaß.

(Michael Radel/rs)

Es darf gepatcht werden

Mit einem gepatchten CLI können Sie die tollsten Sachen machen. Zum Beispiel können Sie die Größe des CLI-Fensters und dessen Titel festlegen. Interessiert? Das ist noch nicht alles. Sie können außerdem einen vier Buchstaben langen CLI-Befehl angeben, der direkt nach dem Start des CLI ausgeführt wird.

»Vier Buchstaben, viel zu wenig?« — warten Sie es ab. Für die meisten Anwendungen reichen vier Buchstaben vollkommen. Doch dazu später. Schauen wir uns erst das zum Patchen benötigte Basic-Programm an. Als kleine Vorbereitung geben Sie bitte im CLI folgendes ein:

```
COPY system/CLI.info  RAM:
PROTECT RAM:CLI.info  WER
ENDCLI
```

So wird das Experimentieren erleichtert, da das CLI-Icon vor einem Überschreiben geschützt ist. Laden Sie nun Amiga-Basic und geben folgendes ein:

```
OPEN "SYS:system/CLI" FOR INPUT AS #1
OPEN "RAM:CLI" FOR OUTPUT AS #2
PRINT #2, INPUT$(&H76C,#1);
Wind$ = "0/0/640/256/Amiga-CLI"+STRING$(6,0)
Befehl$ = "list"
X$ = Wind$ + CHR$(0) + Befehl$
a$ = INPUT$(LEN(x$),#1)
PRINT #2,x$;
PRINT #2,INPUT$(LOF(1)-&H76C-LEN(x$),#1);
CLOSE
```

Die Bedeutung der Variablen:

— Wind$: Der erste Teil dieses Strings entspricht auch der Fenstereingabe des Befehls NEWCLI. Damit dürfte die Syntax bekannt sein. Wichtig ist, daß die gesamte Zeichenkette 27 Byte lang sein muß. Falls Ihr String kürzer sein sollte, füllen Sie ihn entsprechend mit Null-Bytes auf. Im obigen Beispiel werden 6 Null-Bytes angehängt.

— Befehl$: Der auszuführende Befehl. Seine maximale Länge beträgt vier Zeichen.

Das Programm übernimmt den ersten Teil des Original-CLI und kopiert die Daten in ein neues CLI-Programm. Dann flickt der »Patcher« die neuen Parameter des CLI-Windows ein. Am Schluß wird der Rest des Programms CLI kopiert. Sie haben nun ein neues Programm CLI in der RAM-Disk. Probieren Sie es gleich aus: Öffnen Sie das Fenster der RAM-Disk und klicken das CLI-Icon an. Wenn alles stimmt, öffnet sich ein riesiges CLI-Fenster, in dem sofort die Anweisung LIST ausgeführt wird. Im Prinzip können Sie auch alle anderen CLI-Befehle von der Workbench zugänglich machen. Doch damit nicht genug. Wenn Sie mehr wollen als nur LIST oder einen anderen Befehl auszuführen, verwenden Sie die Anweisung EXECUTE.

»EXECUTE ist aber länger als vier Buchstaben«, meinen einige Zweifler. Richtig, aber benennen Sie EXECUTE einfach in »X« um. Eine Alternative wurde in der AMIGA 8-9/87 beschrieben. Geben Sie EXECUTE mittels ASSIGN einen anderen Namen.

```
ASSIGN X: C:EXECUTE
```

Experten sehen schon, was dabei herauskommt. Sie können beim Start des CLI von der Workbench den Befehl Execute ausführen lassen und damit eine beliebige Befehlsdatei abarbeiten lassen. In dieser Batch-Datei können beliebig viele und beliebig lange CLI-Befehle stehen. Allerdings darf der Name des Batch-Files nicht länger sein als zwei Zeichen, aber das läßt sich verkraften, oder?

Vielleicht noch ein paar Anregungen zu diesem mächtigen Tip: Wenn Sie zum Beispiel ED von der Workbench starten möchten. Wählen Sie ein kleines CLI-Fenster, zum Beispiel:

```
"0/0/300/40/CLI-ED"+STRING$(10,0)
```

Als Befehl geben Sie »ED ?« ein. Wenn Sie nun das CLI-Icon in der RAM-Disk anklicken, öffnet sich ein CLI-Fenster, und der Anwender braucht nur noch den Namen der zu bearbeitenden Datei eingeben. Natürlich sollte man dafür nicht das wertvolle CLI-Icon opfern. Weitaus besser ist es, wenn Sie ein neues CLI erstellen und beispielsweise CLI_ED nennen. Hierzu muß man nur das gepatchte RAM-CLI markieren und aus dem Menü der Workbench »Duplicate« wählen. Die Kopie wird mit »Rename« umbenannt und auf die Workbench kopiert. Und ein letzter Hinweis. Erstellen Sie für alle CLI-Befehle, die Sie von der Workbench aufrufen möchten, eigene Icons. *(Angela Schmidt/rs)*

Ganze Sachen

Amiga-Basic ist gegenüber anderen Hochsprachen wie C oder Modula deutlich langsamer. Klar, Basic ist eine Interpretersprache. Wenn er den Befehl RUN erhält, führt der Interpreter die Anweisungen sofort aus. Hierzu übersetzt er Zeile für Zeile eines Programms in die Maschinensprache des Amiga. Der Vorteil des Interpreters: Der Programmierer schreibt sein Listing, tippt RUN ein und schon führt der Computer das Programm aus. Klappt etwas nicht, stoppt man das Programm und beseitigt im List-Fenster sofort alle Ungereimtheiten.

Der Nachteil: Ein lauffähiges Programm ist immer auf den »Übersetzer« angewiesen, und das Programm muß immer wieder in Maschinenbefehle übersetzt werden.

Damit auch Basic-Programme schnell sind, sollte man einige Ratschläge beherzigen: Vor allem die Berechnung mit Fließkommazahlen kostet Zeit. Wenn der Programmierer weiß, daß eine Zahl nur ganzzahlige Werte annimmt, wie ein Zähler für eine Schleife, warum soll er die langsamen Fließkommazahlen verwenden? Warum soll der Interpreter die Stellen hinter dem Komma berücksichtigen?

Integerzahlen sind beim Amiga durch ein an den Namen angehängtes Prozentzeichen kenntlich gemacht (a%, b%, Integer%). Wenn Sie nun in einem Programm nachträglich Fließkommazahlen berücksichtigen möchten, ist es ziemlich aufwendig, allen Variablen ein Prozentzeichen anzuhängen.

Hier helfen zwei Tricks:
— Es gibt die Möglichkeit, alle Variablen mit dem Befehl DEFtyp als Integer zu definieren:

```
DEFINT a-z
```

Nach dieser Zeile sind alle Variablen, die mit einem in den Grenzen angegebenen Buchstaben beginnen, automatisch als Integer festgelegt.
— Wenn Sie allerdings nur einzelne Variablen als Ganzzahl kenntlich machen möchten, wählen Sie die zweite Methode: Der ED aus dem CLI kann die Prozentzeichen hinter jede Variable setzen. Laden Sie hierzu das als ASCII-Text gespeicherte Basic-Programm mit dem ED:

```
ED Programmname
```

Die Befehlsfolge

```
T; RP E "Var" " Var%"
```

hängt der im ersten Parameter angegebenen Variable ein % an. Verwenden Sie

```
T; RP EQ."Var" "Var%",
```

erfolgt vor jedem Austausch eine Sicherheitsabfrage. Zu beachten ist, daß Integervariablen nur ganzzahlige Werte zwischen - 32768 und 32767 annehmen dürfen. Den Geschwindigkeitsvorteil zeigt ein Beispiel, das Sie selbst ausprobieren sollten:

```
PRINT Time$
FOR i1% = 0 TO 1000
NEXT i1%
PRINT Time$
FOR i2 =  0 TO 1000
NEXT i2
PRINT TIME$
```

Mit einer Variablen einfacher Genauigkeit benötigt die Schleife ungefähr 4 Sekunden. Die schnelle Variante ist nach ... probieren Sie es einmal aus. *(Torsten Kerschat/rs)*

So liegen Sie richtig

In Basic ist immer das zuletzt definierte Fenster aktiviert. Möchte man nun in einem anderen Fenster Daten von der Tastatur einlesen, leitet WINDOW n die Eingabe entsprechend um. Bevor jedoch eine Eingabe erfolgen kann, muß das Fenster mit der Maus aktiviert werden. Der Aufruf der Intuition-Anweisung »ActivateWindow(&WindowHdl)« vor dem INPUT-Befehl macht diesen Handgriff überflüssig:

```
LIBRARY "intuition.library"
WINDOW 2,"Ein Test-Fenster",(10,10)-(100,100),15
CALL ActivateWindow(WINDOW(7))
INPUT a$
```

Das kurze Beispiel zeigt, wie Sie ein Fenster auch ohne Maus aktivieren. In der ersten Zeile steht ein Statement zum Öffnen der Intuition-Library. Eingefleischte Basic-Programmierer wissen selbstverständlich, daß dieser Befehl nur funktioniert, wenn sich die Datei »Intuition.bmap« im selben Verzeichnis wie das Basic-Programm oder im Ordner »s« der Startdiskette befindet.
(Bruno Goldmann/rs)

Umlaute auch für Basic

Für den Basic-Programmierer ist die Behandlung der deutschen Sonderzeichen durch Amiga-Basic ein Ärgernis. Wer sich dazu verleiten läßt, sie in Sprung- oder Variablennamen zu verwenden, erlebt schon bei der Eingabe die Aversion des Systems gegen nationale Sonderwünsche. Am wenigsten mag Amiga-Basic das »ü«. Es wird nicht einmal in Strings akzeptiert. Der Interpreter löscht es zunächst im Listing und bei erneutem Abschicken der Programmzeile sogar im Programmtext. Wegen dieser Eigenheit ist das »ü« praktisch tabu. Es kann nur mühsam durch ersatzweise Eingabe von CHR$(252) ersetzt werden. Es gibt aber eine andere, sehr viel einfachere Lösung für den Umgang mit dem »ü«. Man kann sich nämlich zunutze machen, daß der Amiga das große »Ü« fast genauso darstellt wie das kleine. Soll also ein Text mit »ü« auf dem Bildschirm ausgegeben werden, ersetzt man den kleinen durch den großen Buchstaben. Auf dem Bildschirm ist der Unterschied nicht zu bemerken. Der Basic-Interpreter läßt das große »Ü« unangetastet. *(Jürgen Curdt/rs)*

Schönschrift

Wie kann man die vielen Schriften, die der Amiga zur Verfügung stellt, auch von Basic aus nutzen? Das untenstehende Programm zeigt, wie es gemacht wird. Das Beispiel verwendet die Schriftart Saphir aus dem Verzeichnis »fonts« der Workbench. Beachten Sie, daß das Basic-Programm nur funktioniert, wenn sich die Dateien »diskfont.bmap« und »graphics.bmap« im aktuellen Verzeichnis befinden.

```
1 Window 1, "Saphir-Schrift"
2 DEFLNG a-z
3 DECLARE FUNCTION OpenDiskFont() LIBRARY
4 LIBRARY "diskfont.library"
5 LIBRARY "graphics.library"
6 AltFont = PEEKL(Window(8)+52)
7 NeuFont$ ="sapphire.font"+CHR$(0)
8 Hoehe = 19
9 Pref = 98
10 text(0) = SADD(NeuFont$)
11 text(1) = (2^16) * Hoehe + Pref
12 font = OpenDiskFont (VARPTR(text(0)))
13 IF font <>0 THEN CALL SetFont(WINDOW(8),font)
14 PRINT
15 PRINT "Dies ist Saphir-Font in Amiga-Basic"
16 Ursprungszustand:
17 PRINT
18 CALL CloseFont(font)
19 CALL SetFont(WINDOW(8),AltFont)
20 PRINT
21 PRINT "Wieder ganz der Alte"
22 LIBRARY CLOSE
```

So nutzen Sie die Schriften aus dem Verzeichnis »fonts«

1	Das Programm öffnet zunächst ein Fenster
2	Alle Variablen werden als lange Ganzzahlen definiert
3	Die Funktion OpenDiskFont wird deklariert
4+5	Öffnen der Bibliotheken
6	Eingestellte Schrift merken
7	Name des neuen Zeichensatzes
8	Höhe der Schrift in Punkten
9	Festlegen der Preferences der Schrift
10	Adresse der neuen Schrift
11	Höhe und Preferences werden verbunden
12	Suche nach der neuen Schrift auf Diskette
13	Gefundene Schrift wird in Speicher geladen
15	Schriftprobe
18	Reaktivieren der alten Schrift
22	Libraries wieder ordnungsgemäß schließen

In der Tabelle rechts finden Sie die verschiedenen Schriften mit den verfügbaren Höhen und Preferences. In der Tabelle oben finden Sie einige Erläuterungen zum Programm. Wenn Sie das Beispiel intensiv studieren, können Sie es gezielt verändern. Rufen Sie doch einmal andere Schriften auf. Wie Sie in der Tabelle sehen, steht Ihnen schon auf der Workbench-Diskette eine große Auswahl zur Verfügung. Mit speziellen Font-Generatoren können Sie weitere Schriften kreieren. *(Stefan Grunwald/rs)*

Schriftname	Höhe	Preferences
topaz.font	8	65
	11	67
pcfont.font	8	67
ruby.font	8	98
	12	98
	15	98
sapphire.font	14	98
	19	98
diamond.font	12	98
	20	98
emerald.font	17	98
	20	98
garnet.font	9	98
	16	98
opal.font	9	98
	12	98

»Alle« Tasten im Griff

Mit den Basic-Funktionen gelingt es nicht, die Tasten <Shift>, <ALT> oder <CTRL> abzufragen. Wenn Sie allerdings PEEK(12577793) verwenden, können Sie die Kontrolle dieser Tasten realisieren. Liefert der PEEK-Befehl ein ungerades Ergebnis, wird eine der folgenden Tasten gerade betätigt:

```
49 <rechte Amiga-Taste>
51 <linke Amiga-Taste>
53 <Alt rechts>
55 <Alt links>
57 <CTRL>
59 <CapsLock> wurde aktiviert
61 <Shift rechts>
63 <Shift links>
```

Bei geraden Werten addieren Sie 1 hinzu. Die Summe entspricht der zuletzt gedrückten Taste, die zum Zeitpunkt der Abfrage bereits wieder losgelassen ist.

```
WHILE 1
  LOCATE 1
  PRINT PEEK(12577793)
WEND
```

Mit diesem Programm ermitteln Sie die Werte anderer Tasten. Eine nützliche Anwendung der Registerabfrage ist zum Beispiel das Warten auf einen Tastendruck:

```
WHILE PEEK(12577793) MOD 2 <> 1
WEND
FOR I = 1 TO 350: NEXT ' Verzögerung
a$ = INKEY$
```

Wurde eine Taste mit ASCII-Code gedrückt, steht das Zeichen in A$. Wurde eine Taste wie <Shift> gedrückt, läßt sich diese nach der beschriebenen Methode ermitteln. Die WHILE-Schleife wird in beiden Fällen verlassen. Dies ist mit dem einfachen Befehl INKEY$ nicht zu schaffen. Er reagiert nur auf Zeicheneingaben.

(Angela Schmidt/rs)

Blockweise

Oft ist der Bildschirm zu klein, um Fehler in einem Programm zu finden. Um ein längeres Basic-Programm begutachten zu können, empfiehlt es sich, es auf einem Drucker auszugeben. Hier behält man den Überblick. Oft braucht man allerdings nur eine Subroutine oder einen begrenzten Programmteil, nämlich den, wo sich der Fehler versteckt. Hat man Zeilennummern in seinem Listing verwendet, ist das einfach:

```
LLIST Anfang Ende
```

LLIST kann verwendet werden, um einen bestimmten Bereich auszudrucken. Was soll man aber tun, wenn man keine Zeilennummern verwendet hat? Als Abhilfe erstellt man zunächst mit dem ED des CLI eine Befehls-Datei mit Namen Druck:

```
ED s/Druck
```

In der Datei stehen folgende Anweisungen:

```
ED ram:BasicClip
TYPE >prt: ram:
BasicClip
```

Die Datei wird durch Drücken von <ESC> <X> und <Return> auf Diskette gespeichert. Später braucht man nur noch folgende Schritte:
— Man läßt parallel zum Amiga-Basic ein CLI-Window offen.
— Der gewünschte Programmteil wird mit der Maus markiert.
— Wählt man aus dem Edit-Menü den Punkt »COPY«, kopiert der Interpreter den Block in die RAM-Disk. Die Datei erhält den Namen »Basic-Clip«.
— Jetzt klickt man das CLI-Fenster an und gibt dort ein:

```
EXECUTE Druck
```

Der Editor ED erscheint mit dem Programmteil im Fenster, den man ausdrucken möchte. Jetzt kann man den Text sogar noch editieren. Ansonsten verlassen Sie den Editor direkt mit <ESC>, <X> und <Return>. Der TYPE-Befehl in der Datei Druck sorgt dann für die Ausgabe der Datei auf dem Drucker. Der Umweg über den ED hat einen bestimmten Grund: Diese Aktion braucht man, um in der Datei Basic-Clip die Zeichen für ein <Carriage Return>, »$13« in »$0A« (<Line Feed>)auszutauschen. Die meisten Drucker arbeiten nur mit dem zweiten Wert korrekt.

(Dr. Peter Kittel/rs)

Zusatzlaufwerke »en masse«

Besitzer einer Festplatte können diese partitionieren, also ein physikalisches Laufwerk in mehrere logische Einheiten aufteilen. Doch wußten Sie, daß man auch Disketten relativ einfach »zerlegen« kann? Dazu ist ein externes Laufwerk erforderlich. In der »mountlist« (zu finden im Ordner »devs«) trägt man ein:

```
DF2:    Device = trackdisk.device
        Unit = 1
        Flags = 1
        Surfaces = 2
        BlocksPerTrack = 11
        Reserved = 2
    PreAlloc = 11
        Interleave = 0
        LowCyl = 0 ; HighCyl = 39
        Buffers = 20
        BufMemType = 3

#

DF3:    Device = trackdisk.device
        Unit = 1
        Flags = 1
        Surfaces = 2
        BlocksPerTrack = 11
        Reserved = 2
PreAlloc = 11
        Interleave = 0
        LowCyl = 40 ; HighCyl = 79
        Buffers = 20
        BufMemType = 3

#
```

Anschließend müssen die beiden Treiber noch angemeldet werden:

```
MOUNT df2:
MOUNT df3:
```

Jetzt können beide Partitionen einer Diskette im externen Laufwerk mit:

```
SYS:system/FORMAT drive df2: name Part1
SYS:system/FORMAT drive df3: name Part2
```

formatiert und einzeln beschrieben werden. Beide Hälften verhalten sich wie separate Laufwerke. Man kann Daten zwischen den Parts kopieren, und beiden Partitionen kann ein anderer Name gegeben werden. Falls man Laufwerke unterschiedlicher Größe braucht, paßt man in der »mountlist« die Werte »LowCyl« und »HighCyl« von df2: und df3: entsprechend an. Klar, daß die beiden Bereiche sich auf der Diskette nicht überschneiden dürfen. Bleibt noch anzumerken, daß man das externe Laufwerk jetzt nicht mehr als »df1:« ansprechen kann. Dafür hat man ja zwei neue Laufwerke.

(Jochen Hauck/rs)

Deutsches Datum

Für viele Basic-Programmierer ist das amerikanische Format des Befehls DATE$ sicher ungewohnt. Mit einer Subroutine läßt sich die Angabe transformieren:

```
Datum:
i$=INKEY$
WHILE i$<>"e"
 da$=DATE$
 db$=LEFT$(da$,1)
```

```
 dc$=MID$(da$,2,1)
 Monat$=""
 IF db$=CHR$(48) AND dc$=CHR$(49) THEN Monat$="
Januar"
 IF db$=CHR$(48) AND dc$=CHR$(50) THEN Monat$="
Februar"
 IF dc$=CHR$(51) THEN Monat$="März"
 IF dc$=CHR$(52) THEN Monat$="April"
 IF dc$=CHR$(53) THEN Monat$="Mai"
 IF dc$=CHR$(54) THEN Monat$="Juni"
 IF dc$=CHR$(55) THEN Monat$="Juli"
 IF dc$=CHR$(56) THEN Monat$="August"
 IF dc$=CHR$(57) THEN Monat$="September"
 IF db$=CHR$(49) AND dc$=CHR$(48) THEN Monat$="
Oktober"
 IF db$=CHR$(49) AND dc$=CHR$(49) THEN Monat$="
November"
 IF db$=CHR$(49) AND dc$=CHR$(50) THEN Monat$="
Dezember"
Test2:
 LOCATE 16,27 : COLOR 2,1
 PRINT "Uhrzeit : " TIME$"    "
 LOCATE 18,27 : COLOR 3,1
 PRINT "Datum1 : "MID$(da$,4,2)". "Monat$" "RIGHT$
(da$,4)" "
 LOCATE 20,27 : COLOR 1,3
 PRINT "Datum2 : "MID$(da$,4,2)"."LEFT$(da$,2)"."
RIGHT$(da$,4)" "
 GOTO Datum
WEND
END
```

Der letzte Teil des Programms dient allein als Demonstration. Die entscheidende Routine »Datum« sollten Sie in Ihren eigenen Programmen verwenden. *(Michael Speicher/rs)*

Riesiger Screen mit Automatik

Der Amiga ist in der Lage, mehr als 640 x 512 Punkte auf dem Monitor darzustellen. Wie? Im sogenannten Overscan (bis 704 x 564). Um in Basic ein größeres Bild verwenden zu können, muß der Bildschirm nach links oben verschoben werden, damit alle Punkte sichtbar sind. Diese Einstellung erfolgt mit den Preferences. Sie kann aber auch per Programm vorgenommen werden:

```
LIBRARY "intuition.library"
LIBRARY "graphics.library"
DECLARE FUNCTION ViewAddress& LIBRARY
ViewAdd&=ViewAddress&
POKEW ViewAdd&+14,112
POKEW ViewAdd&+12,30
CALL RemakeDisplay
SCREEN 1,704,560,tiefe,4
WINDOW 2,,,16,1
LINE (0,0)-(695,555),3,B
FOR i=1 TO 10000
NEXT i
POKEW ViewAdd&+14,129
POKEW ViewAdd&+12,44
RemakeDisplay
END
```

Mit »MoveScreen« läßt sich ein Screen lediglich in der Vertikalen verschieben. Eine Möglichkeit, ein Bild auch in der Horizontalen zu bewegen und zu zentrieren, besteht im Ändern der View-Daten. Die letzten vier Zeilen des Beispiels stellen vor Verlassen des Programms die alte Bildposition wieder her. Es kann unter Umständen nötig sein, im Overscan das Bild mit den Reglern am Monitor zu verkleinern, oder je nach Gerät andere Startkoordinaten in der View-Struktur einzusetzen. Hier sollten Sie ein wenig experimentieren. *(Bruno Goldmann/rs)*

Funktionstastenabfrage in Basic

Auch mit Amiga-Basic ist es möglich, die Funktionstasten abzufragen. Das untenstehende Listing zeigt ein Beispiel dafür. Je nach gedrückter Taste verzweigt das Programm zu einem bestimmten Unterprogramm. Die Unterprogramme können Sie nach eigenen Wünschen gestalten. Hier sind Ihrer Fantasie keine Grenzen gesetzt. *(Thomas Palmer/rs)*

```
start: i$=INKEY$
IF i$="" THEN GOTO start
i = ASC(i$)
IF i>128 THEN ON i-128 GOSUB F1,F2,F3,F4,F5,
F6,F7,F8,F9,F10
GOTO start
F1:
 PRINT "F1-Taste wurde gedrückt"
 RETURN
F2:
 PRINT "Diesmal war es F2"
 RETURN
...
F10:
 PRINT "F10 funktioniert auch"
 RETURN
```

Ein Beispiel in Basic zur Abfrage der Funktionstasten

Umweg für Umlaute

Wenn man Umlaute und Sonderzeichen auf dem Drucker ausgeben möchte, kann es mit Amiga-Basic Schwierigkeiten geben. Hier eine Routine, die Abhilfe schafft:

```
DATA 196,91,228,123,214,92,246,124,220,93,252,125,
     223,126
  FOR i=1 TO 7
    READ u(i),e(i)
  NEXT i
  COLOR 2
  LOCATE 2,3:PRINT "Drucker einschalten nicht
  vergessen!"
  COLOR 1
  LOCATE 5,3:PRINT "Ein Wort oder einen Satz mit
   vielen Umlauten eingeben. Dann <RETURN>"
Hauptprogramm:
  LOCATE 10,1: PRINT SPACE$(80)
  LOCATE 10,3: LINE INPUT n$
  OPEN "par:" FOR OUTPUT AS #1
  CALL Chartest(n$,e(),u())
  PRINT #1,n$
  CLOSE #1
  GOTO Hauptprogramm
  END
SUB Chartest(a$,e(),u()) STATIC
  FOR i=1 TO 7
    FOR w=1 TO LEN(a$)
      x=INSTR(a$,CHR$(u(i)))
      IF x>0 THEN MID$(a$,x,1)=CHR$(e(i))
    ELSE Sprung
    NEXT w
Sprung:
  NEXT i
END SUB
```

Die Subroutine »Chartest« sorgt dafür, daß die Umlaute korrekt auf dem Drucker ausgegeben werden. Sie läßt sich in andere Programme einbauen und wie im Beispiel gezeigt einsetzen. *(Jürgen Bartels/rs)*

Keine Namen bitte

Wenn man ein Basic-Programm nicht auf der Diskette sichern möchte, von der Amiga-Basic geladen wurde, muß man umständlicherweise die Laufwerksnummer beziehungsweise den Namen der betreffenden Diskette angeben:

```
SAVE "df0:amiga"
SAVE "Programme:amiga"
```

Wer häufig auf die zweite Diskette zugreifen möchte, sollte diese im internen Laufwerk einlegen und den Befehl CHDIR anwenden:

```
CHDIR "df0:"
CHDIR "Programme:"
```

Danach greift der Amiga bei allen Diskettenoperationen automatisch auf die richtige Diskette zu. Es genügt also der Aufruf:

```
SAVE amiga
```

Natürlich kann man auch Unterpfade mit CHDIR kennzeichnen. Gerade in diesem Fall macht sich der Befehl bezahlt, da der Programmierer nicht immer den gesamten Pfadnamen einzutippen braucht. *(Marcus Spahn/rs)*

Patch: Basic ohne Bilder

Viele Basic-Programmierer haben sich ihre »persönliche« Basic-Diskette angelegt — mit Autostart, Amiga-Basic und allen wichtigen »bmap«-Dateien. Auf der Diskette läßt sich Platz sparen, wenn man auf die Workbench verzichtet; Basic-Programme lassen sich auch vom CLI starten. Dann sind die Icons überflüssig. Man kann die »info«-Dateien löschen. Allerdings erzeugt Amiga-Basic immer neue Piktogramme, wenn man ein Programm sichert. Die Lösung hierfür ist ein Patch:

```
CLEAR, 50000&
NeueStelle$ =""
For I = 1 TO 12
NeueStelle$ = NeueStelle$ + CHR$(0) :NEXT I
OPEN ":AmigaBasic" FOR INPUT AS #1 LEN=4096
OPEN ":NeuesBasic" FOR OUTPUT AS #2 LEN =4096
PRINT #2, INPUT$(H57a4,#1);
PRINT #2, INPUT$(H57a4,#1);
PRINT #2, INPUT$(H57a4,#1);
PRINT #2, INPUT$(H57a6,#1);
PRINT #2, NeueStelle$;
A$=INPUT$(12,#1)
PRINT a$
WHILE NOT EOF(1)
 PRINT #2,INPUT$(1,#1);
WEND
CLOSE #1,#2
```

Dieser Patch ist für die Basic-Version mit einer Länge von 103500 Byte geeignet. Ein kleiner Nebeneffekt: Programme werden vom Basic-Interpreter schneller gespeichert.
(Carsten Mallek/rs)

Kleine Felerr

Kennen Sie das? Man hat ein Programm geschrieben und testet es. Beim Laufenlassen stellt man fest, daß sich immer noch ein Bug (engl.: Wanze; Fehler) im Programm versteckt. Um das Listing auf dem Bildschirm erscheinen zu lassen, existieren zwei Wege:
— Die alte Methode: Man drückt gleichzeitig die Tasten <CTRL> und <C> und bricht das Programm ab. Nun gibt man LIST ein und wartet, und wartet...
— Die Alternative: Man bricht das Programm mit <CTRL C> ab und gibt »ü« ein. Sofort erscheint das List-Fenster. In der oberen Zeile zeigt der Amiga einen »Syntax Error« an. Klicken Sie einfach auf OK und editieren Ihr Programm. Vor allem bei langen Listings ist der Trick mit dem »ü« wesentlich schneller.
(Karsten Könning/rs)

Immer feste drauf

Der Programmeditor des Amiga-Basic befindet sich immer im »Insert«-Modus. Das bedeutet, der Amiga fügt eingegebene Zeichen an der Position der Schreibmarke immer vor dem nächsten Wort ein. Dies ist jedoch lästig, wenn man Schreibfehler in einem Programm korrigieren möchte: Immer muß man den alten Text mit der Taste <DEL> löschen. Wenn man dagegen den zu überschreibenden Text mit dem Zeiger der Maus als Block markiert und dann ein Zeichen eingibt, ersetzen die neuen Zeichen automatisch die alten. So kann man eine »Overwrite«-Funktion simulieren. *(André Deparade/rs)*

Mit 60 fängt der Spaß erst an

Hier ist die Lösung auf die Anfrage von Dietrich Bartel im Leserforum. Herr Bartel suchte einen Weg, um in Basic von 60 auf 80 Zeichen umzuschalten. Hier eine Lösung:

```
DECLARE FUNCTION OpenFont& LIBRARY
LIBRARY "intuition.library"
LIBRARY "graphics.library"

SUB CPL (Chars%) STATIC
 IF Chars%=60 Then Height%=9 ELSE Height%=8
 IF FontInfo&<>0 THEN CALL CloseFont(FontInfo&)
 Font$="topaz.font"+CHR$(0)
 Font&(0)=SADD(Font$) : Font&(1)=Height%*2^16
 FontInfo&=OpenFont&(VARPTR(Font&(0)))
 IF FontInfo&<>0 THEN CALL
SetFont (WINDOW(8),FontInfo&)
END SUB
```

Im Hauptprogramm deklarieren (= bekanntmachen) Sie zunächst die Funktion »OpenFont&«. Außerdem wird die »intuition.library« und die »graphics.library« geöffnet. Basic-Programmierer wissen, daß sie die zugehörigen »bmap.Dateien« benötigen. Der Aufruf des Unterprogramms erfolgt mit dem Statement »CPL 60« beziehungsweise »CPL 80«. Vergessen Sie nicht, vor Verlassen des Programms die Bibliotheken mit LIBRARY CLOSE wieder zu schließen. Alle Ausgaben auf dem Bildschirm erfolgen nach Aufruf von CPL im gewünschten Format. Aber Sie können die Routine auch noch ändern. Experimentieren Sie ruhig mit den verschiedenen Routinen zur Veränderung von Zeichensätzen.
(Roberto Papalino/rs)

Die Alternative: großes List-Fenster

Der im AMIGA-Magazin 8/88, Seite 87, abgedruckte Tip verändert das Ausgabe- und List-Fenster des Amiga-Basic mit einem Disketten-Monitor. Die Änderung läßt sich auch per Programm realisieren:

```
OPEN "R",1,"AmigaBasic",1
FIELD #1,1 AS A$
LSET A$ = CHR$(1)
PUT #1,22509
LSET A$ =CHR$(0)
PUT #1,22510
PUT #1,22540
PUT #1,22541
PUT #1,22542
LSET A$ = CHR$(240)
PUT #1,22544
LSET A$ = CHR$(104)
PUT #1,22546
CLOSE #1
END
```

Wenn Sie das Programm eingegeben haben, testen Sie es unbedingt mit einer Kopie von Amiga-Basic — sicher ist sicher.
(Matthias Brehl/rs)

Alles eine Sache der Intuition

Mit den Routinen der »intuition.library« kann der Programmierer einiges anstellen: Verschieben Sie doch einmal mit der Maus einen Screen:

```
DECLARE FUNCTION ViewAddress& LIBRARY
LIBRARY "intuition.library"
vi& = ViewAddress&
WHILE MOUSE(0)=0
 dx% = 66+MOUSE(1)/5.8
 dy% = 14+MOUSE(2)/2.5
 IF dx%>128 THEN dx%=128
 POKEW vi&+12,dy%
 POKEW vi&+14,dx%
 CALL RemakeDisplay
WEND
LIBRARY CLOSE
END
```

Das Programm erlaubt es, das Bild auf dem Monitor mit der Maus zu zentrieren. Sicher können Sie das Beispiel als Grundlage für eigene Experimente mit der Funktion RemakeDisplay verwenden. *(Michael Friedberg/rs)*

»Ene mene meck«, die Maus ist weg

Ein kleiner Gag für Basic-Programmierer/innen:

```
LIBRARY "graphics.library"
CALL FreeSprite(0)
```

Dieses Programm zaubert den Mauspfeil weg. Aber — oh Wunder — sowie die Maus wieder betätigt wird, erscheint der Zeiger wieder auf dem Bildschirm. *(Angela Schmidt/rs)*

Schnelles Blättern mit Basic

Die Tasten <Cursor hoch> und <Cursor unten> in Verbindung mit <Shift> dienen zum seitenweisen Durchblättern in einem Basic-Programm. Wenn Sie ein langes Basic-Programm mit etwa 60 Seiten Umfang oder sogar noch mehr durchblättern, dauert dies recht lange. Sie erreichen eine höhere Geschwindigkeit, wenn Sie das Listing vor dem Umblättern mit <Shift> und zweimal <Cursor rechts> aus dem sichtbaren Bereich verbannen. <Shift> und zweimal <Cursor links> holen den Text zurück.
(Jens Spitczok/rs)

Markieren ohne Druck

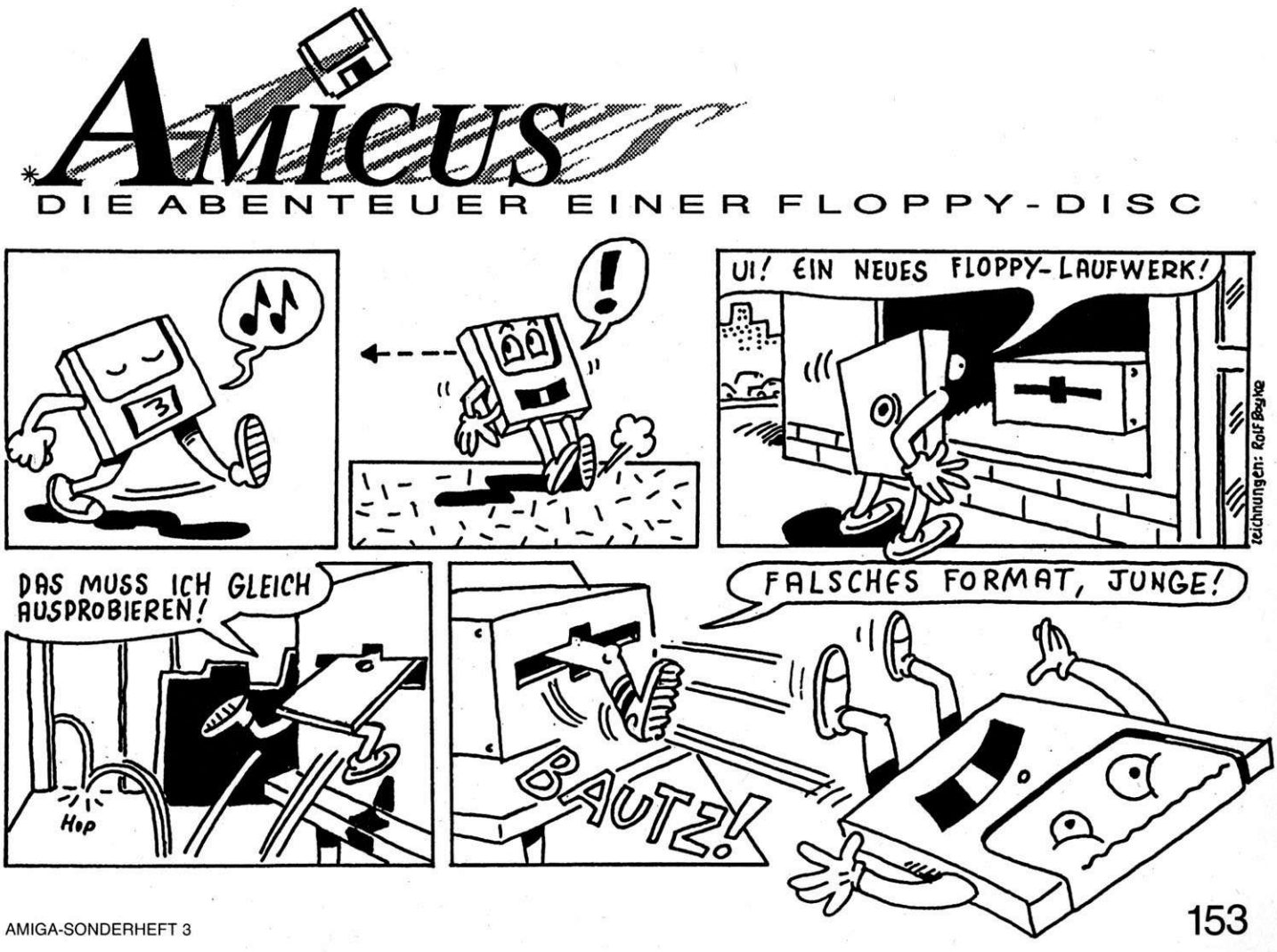
Dieser Trick befaßt sich mit dem Löschen und Kopieren größerer Basic-Programmteile. Er erlaubt es Ihnen, einen Block mit der Maus zu markieren, ohne ständig die linke Maustaste gedrückt zu halten.

Fahren Sie zunächst mit der Maus wie gewohnt an den Anfang des zu markierenden Bereichs. Drücken Sie nun ebenfalls wie gewohnt die linke — nicht loslassen — und dann die rechte Maustaste. Haben Sie beide Tasten gedrückt?

Lassen Sie jetzt die linke vor der rechten los. Schon können Sie mit der Maus ans Ende des Blocks fahren. Beide Tasten sind frei. Am Ende des Blocks wiederholt sich das Spiel in umgekehrter Reihenfolge:

— rechte Taste gedrückt halten;
— linke Taste ebenso;
— rechts zuerst loslassen.

Nun ist der Block markiert. Aber aufpassen, Sie dürfen während der gesamten Prozedur keine falsche Taste drücken.
(Tobias Helge Kosuch/rs)



Battleships

Bei dem Strategie- und Glücksspiel Battleships sollte man alle Schiffe am Rand des Spielfeldes plazieren. Die Möglichkeit eines Zufallstreffers wird dadurch verringert. Treffen Sie ein Schiff des Gegners, versenken Sie es nach Möglichkeit beim nächsten Spielzug. Der Gegner verliert in diesem Fall jeweils vier wertvolle Schüsse. *(Thomas Probst/rs)*

Phantasie III (1)

Jürgen Pfau hat die Lösung für »Phantasie III« gefunden. Folgende Ratschläge gibt er allen Abenteurern auf den Weg: Zuerst muß die zusammengestellte Truppe trainiert werden. Ist die Truppe stark genug, geht man in die »Hall of Giants« bei Tirith im Süden von Scandor. Dort findet man den toten Kilmor. Von dort aus wandert man weiter zu den »Pendragon town archives«. Dort findet man Filmon. Man wird von ihm zur Beerdigung von Kilmor geschickt. Diese findet in den »Dwarven burial grounds« bei Pendleton statt. Dort begrüßen Sie die Gäste und nehmen Platz. Nach der Zeremonie geht man zurück zu Filmon. Von dort aus geht's weiter zur »Chamber of Chronos«. In einem der vier Nebenräume findet man eine Geheimtür, die zum »Key of light« führt. In den unteren Räumen sind die Koordinaten für das »Plane of light«. Geben Sie anschließend die Koordinaten in einem der beiden oberen Räume ein. Im anderen betätigen Sie den »Activision lever«.

Verlassen Sie nun die »Chamber of Chronos«, der Weg führt zum »Plane of light«. Hier muß man mit der Light Feary sprechen. Mit dem »Transportation spell« kommt man zurück nach Pendragon. Gehen Sie erneut zu Filmon. Danach besucht man Lord Wood in einem Zelt in der Nähe von Lunsing. Man muß hier unbedingt den Zauberspruch »Summon elemental« besitzen. Hat man diesen nicht, geht man zu den »Gnome Catacombs«. Dort trifft man einen bestechlichen Zauberer (Wizard), der den genannten Spruch verkauft. Jetzt verkaufen Sie den »Key of light«, holen aber vorher den »Key of dark« aus der Hall of Giants. Gehen Sie anschließend in eine Strohhütte in der Nähe von Flugler. Von hier aus geht's erneut zur Chamber of Chronos und wieder zum Plane of Light. Dort lädt man ein neues Dungeon. Mit Hilfe des »Dark Key« wechseln Sie zur »Plane of Dark«. Gehen Sie in das »Castle of dark«, nehmen das »Wand of Nikademus« und wandern zum »Room of Nikademus«. Dort versuchen Sie die Wand zu zerbrechen. Man wird dann zu einem Ausgang teleportiert, durch den »Netherworld« erreicht wird. Nun gehen Sie zur »Fortress of Nikademus«, dort finden Sie besagten Nikademus. Suchen Sie ihn hinter einem der vielen geheimen Eingänge. Ist Nikademus gefunden, benutzen Sie den »57 Spell« — Lord Wood erscheint. Greifen Sie nun Nikademus an und besiegen ihn, ist »Phantasie III« gelöst und Scandor gerettet. *(Jürgen Pfau/rs)*

Katakis

Bei dem aufwendigen Shoot'em'up-Spiel »Katakis« ist eine Trainerversion eingebaut. So erreichen Sie den Trainer:
1. Warten Sie, bis der Amiga die Data-Disk verlangt.
2. Drücken Sie <Y>, Voraussetzung ist der deutsche Tastaturtreiber (Setmap d).
3. Sind die Daten geladen, trifft man die Auswahl zwischen Ein- und Zwei-Spieler-Modus und drückt den Joystick-Knopf.
4. Schließen Sie sofort die Maus an den Port 2 und halten deren rechte Taste so lange gedrückt, bis vom Laufwerk nachgeladen wird (LED leuchtet).
5. Hat man alles richtig gemacht, startet das Spiel ganz normal.
6. Verliert man das fünfte Spielerleben, wird der Highscore gelöscht und man verfügt über beliebig viele Raumschiffe. Die Score-Anzeige zählt von nun ab die Anzahl der zerstörten Raumschiffe.

Diesen Trick erfahren Sie auch, wenn Sie Katakis einmal ohne Trainer bis zum Schluß überstehen. *(Roland Schnarr/rs)*

Garrison

Die Medizinkästen helfen nur im Notfall. Erst wenn die Punktzahl unter 5000 sinkt, lohnt sich der Weg zum Verbandskasten. Das Punktekonto wird dann immer auf maximal 5000 aufgestockt. *(rs)*

Interceptor

Wer den »F/A-18 Interceptor« einmal gespielt hat, kommt so schnell nicht von dieser Flugsimulation los. Aber auch erfahrene Spieler haben Schwierigkeiten, die »Selectable Missions« zu bestehen. Oliver Stosiek hat Tips für die sechs Missionen zusammengestellt:

Visual Confirmation:

Nach dem Start das nicht identifizierte Flugzeug aufspüren und dabei auf die MIGs achten. Nur schießen, wenn man angegriffen wird und nach der Aufforderung schnell zum Flugzeugträger.

Emergency Defence Operation:

Zum Präsidentenflugzeug fliegen und dieses vor den Angriffen der MIG-29 schützen. Auch hier erst schießen, wenn man angegriffen wird. Ist der Präsident gelandet, schnurstracks zum Flugzeugträger zurück.

Intercept Stolen Aircraft:

Nach dem Start die geklaute F-16 suchen und zum Umkehren zwingen. Wenn diese nicht reagiert, abschießen. Anschließend zurück zum Flugzeugträger.

Search And Rescue Operation:

Zum Bruchpiloten fliegen, der in der Nähe einer Insel abgestürzt ist. Der Abwurf ist am genauesten durchzuführen, wenn Sie die Position des Bruchpiloten in zirka 5000 Fuß Höhe anfliegen und dann im Sturzflug bis auf zirka 1500 Fuß herabfallen. Kurz vor dem Abdrehen die Tastenkombination zum Abwurf der Rettungsinsel drücken.

Intercept Incoming Cruise Missile

Suchen Sie die Cruise Missile und zerstören diese mit einer Sidewinder-Rakete. Lassen Sie sich möglichst nicht vor dem Abschuß durch MIGs ablenken. Nach erfolgreichem Absch(l)uß mit vollem Schub zurück zum Flugzeugträger.

Carrier Sub Mission

Suchen Sie zuerst die vier begleitenden MIGs und schießen diese ab. Falls der Treibstoff zur Neige geht, einen der Flughäfen oder den Träger zum Auftanken ansteuern. Erfüllen Sie anschließend die Mission, viel Glück! *(Oliver Stosiek/rs)*

Ports of Call

Eine Reihe von guten Tips zu »Ports of Call« hat Michael Schmitt parat. Wir wollen Ihnen diese nicht vorenthalten:
1. Sie sollten anfangs ein Pre-Owned-Schiff erwerben.
2. Wählen Sie San Francisco als Heimathafen.
3. Schlagen Sie Waffengeschäfte nie aus.
4. Lehnen Sie den kleinen Nebenverdienst ab, der Sie dauernd nervt. Er bringt nicht viel Geld, aber große Schwierigkeiten.
5. Fahren Sie gleich zu Beginn die Strecke San Francisco — Cape Town. Dabei werden Sie Mitglied bei den »Kap Hornern«, ein Statuspunkt winkt.
6. Das Bergen von Schiffbrüchigen bringt ein gutes Gewissen und Statuspunkte. Sie müssen dazu mit der Spitze des Schiffes eine der blinkenden Ecken anfahren.

Schummeln Sie mit

Die Spiele auf dem Amiga bezaubern jeden. Leider erlebt selbst ein begnadeter Joystick-Jongleur selten das letzte Level, löst selbst der erfahrenste Abenteurer nur mit großer Mühe ein Adventure. Wir geben Ihnen Tips, wie Sie Ihr Lieblingsspiel in den Griff bekommen.

7. Meiden Sie Karachi. Diese Stadt ist Krisengebiet, Ihre Schiffe können darunter leiden.
8. Wenn Sie längere Zeit finanziell gut dastehen wollen, sollten Sie sofort nach Kauf eines Schiffes in Ihr Büro gehen und die Hypotheken tilgen.
9. Bleibt nur noch, Ihnen viel Spaß und Erfolg zu wünschen.

(Michael Schmitt/rs)

Karate Kid

Bei »Karate Kid« existiert ein Cheat-Modus. Mit <F10> aktivieren Sie den Trainer, mit <P> gelangen Sie in den nächsten Level. *(Carsten Flögel/rs)*

Marble Madness

Bei Marble Madness ist das letzte Level oft aus Zeitnot nicht zu schaffen. Wählen Sie den Zwei-Spieler-Modus und ziehen die blaue Kugel noch einige Level mit. Wenn immer die rote Kugel gewinnt, erhält diese Zeit gutgeschrieben. Dann ist der Weg durch das letzte Level nicht mehr so schwer. *(Jens Scheifgen/rs)*

Tron 5000

Alle, die »TRON 5000« besitzen, können sich freuen: Die einzelnen Level sind mit einem Malprogramm zu verändern. Die Level liegen auf der Diskette als IFF-Files vor, die beispielsweise von DPaint eingelesen werden können. Mit den ersten drei Farben erstellen Sie den Hintergrund, mit den weiteren setzen Sie Gebäude in die Landschaft. Auf der Diskette liegen noch weitere Bilder vor, die Sie verändern können. *(Daniel Hauke/rs)*

Bard's Tale II

Schaffen Sie bei »Bard's Tale II: The Destiny Knight« einen Dungeon nicht, weil die Ausrüstung Ihrer Recken zu schlecht ist? Vielleicht hilft folgendes kleine Basic-Listing:

```
REM SuperItem (w) by Mark Sapa
KILL "Destiny Knight Character Disk:items"
OPEN "DFO:items" FOR OUTPUT AS 1
PRINT # 1,CHR$(1);
FOR I=1 TO 339
PRINT#1,CHR$(255);
NEXT I
CLOSE 1
END
```

Nach dem Eintippen in Amiga-Basic und Speichern legen Sie die Character-Disk in das Laufwerk df0: und starten mit RUN das kurze Programm. Nachdem die Diskettenzugriffe abgeschlossen sind (LED erlöscht), können Sie alle Ausrüstungsgegenstände in »Garth's Equipment Shop« kaufen, die man sonst mühselig zusammensuchen muß. Allerdings benötigen Sie dafür eine ganze Menge Kleingeld. *(Mark Sapra/rs)*

Leisure Suit Larry...

Viele Larry-Anhänger ärgern sich, daß vor jedem Start des Programms eine Reihe von lustigen, auf die Dauer jedoch nervenraubenden Fragen zu beantworten ist. Die Programmierer haben jedoch einen Umweg vorgesehen:

Drücken Sie die Tastenkombination <ALT X>, und sofort erscheint »Thank you...« und der Spaß kann beginnen.

(Timm Püller/rs)

Sarcophaser

Beim »Sarcophaser« ist neben dem Spiel eine Trainerversion vorhanden. Drücken Sie gleichzeitig die Tasten F3, F5 und F6 (eventuell mehrmals), erhalten Sie unendlich viele Leben.

(Timm Püller/rs)

Interceptor (2)

Noch ein Trick, wie Sie ohne Schwierigkeiten die »Selectable Missions« erfüllen: Geben Sie das folgende kurze Listing mit Amiga-Basic ein. Speichern Sie es auf Diskette und starten das Programm mit RUN. Auf der im Laufwerk »df0:« eingelegten Diskette wird anschließend die Datei »config« erzeugt. Diese Diskette legen Sie nach dem Starten des Interceptors in das Laufwerk ein, wenn die DATA-Disk verlangt wird. Alle sechs Missionen sind dann erfüllt, Ihr Logbuch täuscht zahlreiche Erfolge vor. Hier das Listing, das den Interceptor täuscht:

```
OPEN "df0:config" FOR OUTPUT AS #1
FOR x=1 TO 78
PRINT#1, CHR$(1);
NEXT
CLOSE
```

(G.Sauer/rs)

Sex Vixens From Space

Michael Bartels hat das Adventure erfolgreich überstanden. Er schlägt folgenden Lösungsweg vor:

down, blastoff, up, south, south, south, open, south, east, up, get card (die Karte muß mit der Maus in den dunklen Schlitz eingefügt werden), north, north, pick up coin, get card (mit Maus auf Roboter klicken), west, get card (auf den Schlitz über Lift klicken), turn on computer, where is lila?, undress lila, make love, ask, east, get coin (in die »Vending Machine«), west, west, south, south, up, north, north, down, blastoff, up, south, south, south, open, south, open, west, west, eat meat, east, east, east, take a nap, west, south, get capsule (auf den Typen klicken), down, blastoff, up, south, south, south, open, south, north, north, undress, lie down, make love, ask, east, push green, push red, push green, west, west, wait, lie down, make love, get remote, run east, ride sky-bike, up, down, blastoff. *(Michael Bartels/rs)*

Impact

Das Arcade-Spiel »Impact« bietet durch sein Paßwort-System auch nach längerer Zeit noch Spielspaß. Dennoch ist es manchmal schwer, alle Level durchzuspielen. Meist scheitert ein Spieler gerade vor dem Erreichen des nächsten Schlüsselwortes. Hier nun die Liste für alle, die sich nicht plagen wollen, um die höchsten Stufen zu erreichen.

Level 11:	GOLD
Level 21:	FISH
Level 31:	WALL
Level 41:	PLUS
Level 51:	HEAD
Level 61:	FORK
Level 71:	ROAD
Level 81:	USER

(Matthias Brunner/rs)

Spiele von der Workbench laden

Viele Spiele, zum Beispiel »Defender of the Crown« oder »Shanghai«, verwenden eigene Zeichensätze. Wenn Sie diese Spiele von der Workbench starten wollen, muß der entsprechende Zeichensatz (Font) vorher in das Dateiverzeichnis »Fonts« auf der Startup-Diskette kopiert werden. Die Arbeit lohnt sich. Nun können Sie jederzeit die genannten Spiele von der Workbench starten, ohne Ihren Amiga mit den Originaldisketten booten zu müssen. *(Edwin C. Wirth/rs)*

Testdrive

Kennen Sie Testdrive? Wenn der Fahrer mit etwa 140 bis 160 Meilen/Stunde durch die Kurven rast, kann es schon mal passieren, daß der Wagen droht, aus der Kurve getragen zu werden. Dann sollte der Fahrer so lange den Feuerknopf drücken, bis der Wagen wieder auf gerader Strecke ist. *(K.E.B./rs)*

Flight Simulator II

Wer es leid ist, beim Einlesen gespeicherter Voreinstellungen in den RAM-Speicher die Diskette zu wechseln und dabei einen Programmabsturz durch Fehlbedienung zu riskieren, der möge schlicht die Datei »f7« auf die Programmdiskette (natürlich auf die Sicherheitskopie) kopieren und beim Aufruf des Menüpunktes »load RAM from disk« jeweils den Requester links oben anklicken (insgesamt zweimal) und die Sache ist erledigt.

Die Datei »f8« auf der Programmdiskette enthält übrigens die Landschaft, und diese kann (theoretisch) auch durch eine andere ersetzt werden. *(Detlef Rothe/rs)*

Ports of Call (2)

Die hohen Preise für Treibstoff und Reparaturen bei der Handelssimulation ärgerten Arpad Zölch. Er fand einen Weg, diese zu umgehen:

Die Preise sind Tagespreise, die sehr starken Schwankungen unterworfen sind. Wenn Sie in einen Hafen eingelaufen sind, verschaffen Sie sich zuerst einen Überblick über die Preislage. Wird Treibstoff für weniger als 100 Dollar angeboten, füllen Sie Ihr Schiff restlos voll. Sind die Reparaturkosten geringer als 30000 Dollar, lassen Sie Reparaturen durchführen. Der Schaden sollte allerdings 10 Prozent übersteigen. Sind die Bedingungen nicht günstig, bleiben Sie einen Tag im Hafen und beginnen erneut mit der Sondierung der Preise. *(Arpad Zölch/rs)*

Tass Times in Tone Town

»Wer löst Tass Times?« — sogar im Leserforum der AMIGA wurde schon nach Hilfe gefragt. Damit der Einstieg in dieses Adventure leichter fällt, verraten wir die ersten Schritte:

— Zuerst muß der Spieler nach Süden in die Küche gehen.
— In der linken Ecke steht ein »jar« (Glas).
— Wer das Glas untersucht, findet einen Schlüssel, den er an sich nehmen sollte (look into jar and take key).
— Dann führt der Weg einmal nach Norden und einmal nach Westen.
— Nun steht der Abenteurer vor einer verschlossenen Tür, die er mit dem Schlüssel öffnet (use key).
— In dem Raum gilt es, das Aquarium (FishBowl) zu untersuchen. Der Suchende findet dort die sogenannten »guitar picks«. In der musikalischen Stadt Tone Town dienen diese Utensilien als Zahlungsmittel.
— Weiter geht's nach Osten.
— Man gelangt in ein Laboratorium. Hier muß der Generator eingeschaltet werden (turn on switch).
— Der Generator ist das Tor nach draußen. Mit dem Befehl »enter hoop« oder »go hoop« verlassen Sie das Haus. Der Teleporter bringt Sie direkt in die Stadt.

Und wie geht es weiter? Das sollten Sie selbst herausbekommen. Ein paar Tips:

Wer in der Stadt nicht auffallen möchte, sollte sich die Haare färben lassen und die Kleider wechseln.

(Mit freundlichen Grüßen an »Softtechnics« und einen anonymen Amiga-Fan/*rs*)

Phantasie III (2)

Noch mehr Tips zu »Phantasie III«:

— Sie sollten auf jeden Fall einen Priester, einen Zauberer und mindestens einen Kämpfer auf Ihre Reise mitnehmen.
— Schicken Sie Ihre Gruppe zuerst in den Dungeon neben Pendragon. Vermeiden Sie aber zunächst den linken Gang und achten Sie auf geheime Stollen.
— Den Dungeon »Guard Tent«, der im Norden liegt, dürfen die Helden erst betreten, wenn sie den Level 10 oder 11 erreicht haben. Ansonsten nehmen die Leichtsinnigen ein schnelles Ende.
— Wenn die Suchenden anfangs ins Gras beißen, lösen Sie lieber Reset aus und beginnen von vorne. Es bringt nichts, in den ersten Stufen als »Untoter« herumzulaufen.
— Auf jeden Fall in jeder Stadt den Spielstand speichern.

Wenn Sie diese Ratschläge befolgen, sollte die Rettung Scandors leichter fallen. *(Thorsten Imsande/rs)*

Guild of thieves

»The Guild Of Thieves« ist ein Abenteuerspiel, welches direkt gebootet wird. Sie können es nicht von der Workbench aus laden. Leider läuft das Programm nur mit dem amerikanischen Tastaturtreiber. Die Vertauschung von <y> und <z> ist besonders störend. Doch dies läßt sich vermeiden: Erstellen Sie zunächst eine Sicherheitskopie der Originaldiskette. Fügen Sie in der »Startup-Sequence« der Kopie diese Zeile ein:

```
Setmap d
```

Den Befehl SETMAP und den Treiber für die deutsche Tastatur finden Sie auf jeder Workbench-Diskette. Kopieren Sie SETMAP aus dem Verzeichnis »systems« in den C-Ordner der Boot-Diskette. Legen Sie ebenfalls auf der Boot-Diskette mit MAKEDIR ein neues Dateiverzeichnis mit dem Namen »devs/keymaps« an:

```
MAKEDIR devs/keymaps
```

In dieses kopieren Sie die Datei »d« aus dem gleichnamigen Verzeichnis der Workbench-Diskette. Wenn Sie nun den Amiga mit der neuen Diskette starten, wird der deutsche Tastaturtreiber geladen. *(Sven Friedrich/rs)*

King of Chicago

Mit diesem Kniff werden Sie schnell zum Boß der Mafia:

1. Killen Sie den »Old man«, schon gehört der Norden Ihnen.
2. Erobern Sie den Westen. Nehmen Sie Maschinengewehre, aber verschonen Sie die Frauen.
3. Setzen Sie jetzt das Gehalt Ihrer Freundin Lola auf Null – seien Sie so richtig unfreundlich zu ihr.
4. Natürlich wird Lola Sie verlassen. Sie flirtet statt dessen mit dem Chef des Südens »Santucci«. Ein kleiner Junge erzählt Ihnen davon. Gehen Sie der Sache nach und folgen Ihrer alten Liebe.
5. Lola wird in ihrer Lieblingsbar von Santucci bedrängt. Gehen Sie in das Lokal. Dort müssen Sie nur noch Santucci und seinen Leibwächter Guido erledigen, dann sind Sie am Ziel.

(Manuel Semino/rs)

Karate King 2

Zwei Tips zu »Karate King 2«, die Sie auch in anderen Spielen umsetzen können:

1. Die Hintergrund- und Highscore-Grafiken bei »Karate King 2« liegen als IFF-Datei vor. Karate-Spieler können diese Bilder mit Deluxe-Paint 2 laden, verändern und austauschen. Wenn Sie Ihr Kunstwerk vollendet haben, müssen Sie es unter dem alten Namen auf der Diskette speichern. Arbeiten Sie mit einer Kopie. Wählen Sie die Farben Ihrer neuen Hintergrundbilder sorgfältig aus. Ansonsten erscheinen die Kämpfer in den unmöglichsten Farbtönen. Malen Sie Ihre ersten Szenarios am besten mit der Farbpalette der Originale. Anschließend können Sie beliebig mit den Farben experimentieren.
2. Das Programm verträgt sich nicht mit einer Speichererweiterung. Zum Glück hilft »NoFastMem«. Karate-Kämpfer sollten sowohl das Programm »NoFastMem« als auch den Befehl RUN ins C-Directory der Spieldiskette kopieren. Fügen Sie dann noch in der »Startup-Sequence« vor den ECHO-Befehlen den Befehl ein:

```
RUN NoFastMem
```

Wenn Sie nun die Diskette starten, sollte es keine Komplikationen mehr mit einer RAM-Erweiterung geben.

(Andreas Herzog/rs)

Wintergames

»So ein... schon wieder hat sich der Skispringer den Hals gebrochen.« Der Skisprung ist eine der schwersten Disziplinen der »Wintergames« von Epyx. Doch wie so oft ist das Motto auch hier »gewußt wie«:

Wenn man den Knopf des Joysticks genau am unteren Rand der Schanze drückt und nach dem Absprung den Joystick so schnell wie möglich im Uhrzeigersinn dreht, gelangt der Springer fast immer in die richtige Landeposition. Der Athlet erreicht dann meistens 225 Punkte. *(K.E.B./rs)*

Guru-Meditation

An dieser Stelle finden Sie einige Korrekturen zum AMIGA-Sonderheft 2.

AMIGA-Sonderheft 2, Seite 18, »Finanzen im Griff«
Im Kasten auf Seite 22 hat sich leider ein Druckfehler eingeschlichen. Die Programmzeile, vor der beim ersten Start des Haushaltsbuchs das Kommando »REM« eingefügt werden muß, ist die Zeile 19 des Listings.

AMIGA-Sonderheft 2, Seite 51, »Fonts mit allen Tricks«
Leider wurde bei der abgedruckten Version des »FontDesigners« die Datei »header.h« nicht veröffentlicht. Hier die kurze Header-Datei, die vom Listing »fontdesigner« eingebunden wird:

```
void Parent(),DirOut(),ErrOut(),SaveFont(),
Cllibs();
void Box(),Text_out(),Invert(),SetBit(),ClrBit(),
Line();
void AsciiOut(),Matout(),ChangeAscii(),
ChangeMatrix(),Mabox();
void SetMPoint(),DrawInto(),Font_to_Mem(),
Delete(),Scroll();
void Mem_to_Matrix(),Mirror(),Undo(),Rotate(),
Revers(),Expand_x();
void Expand_y(),Programm(),Show();
```

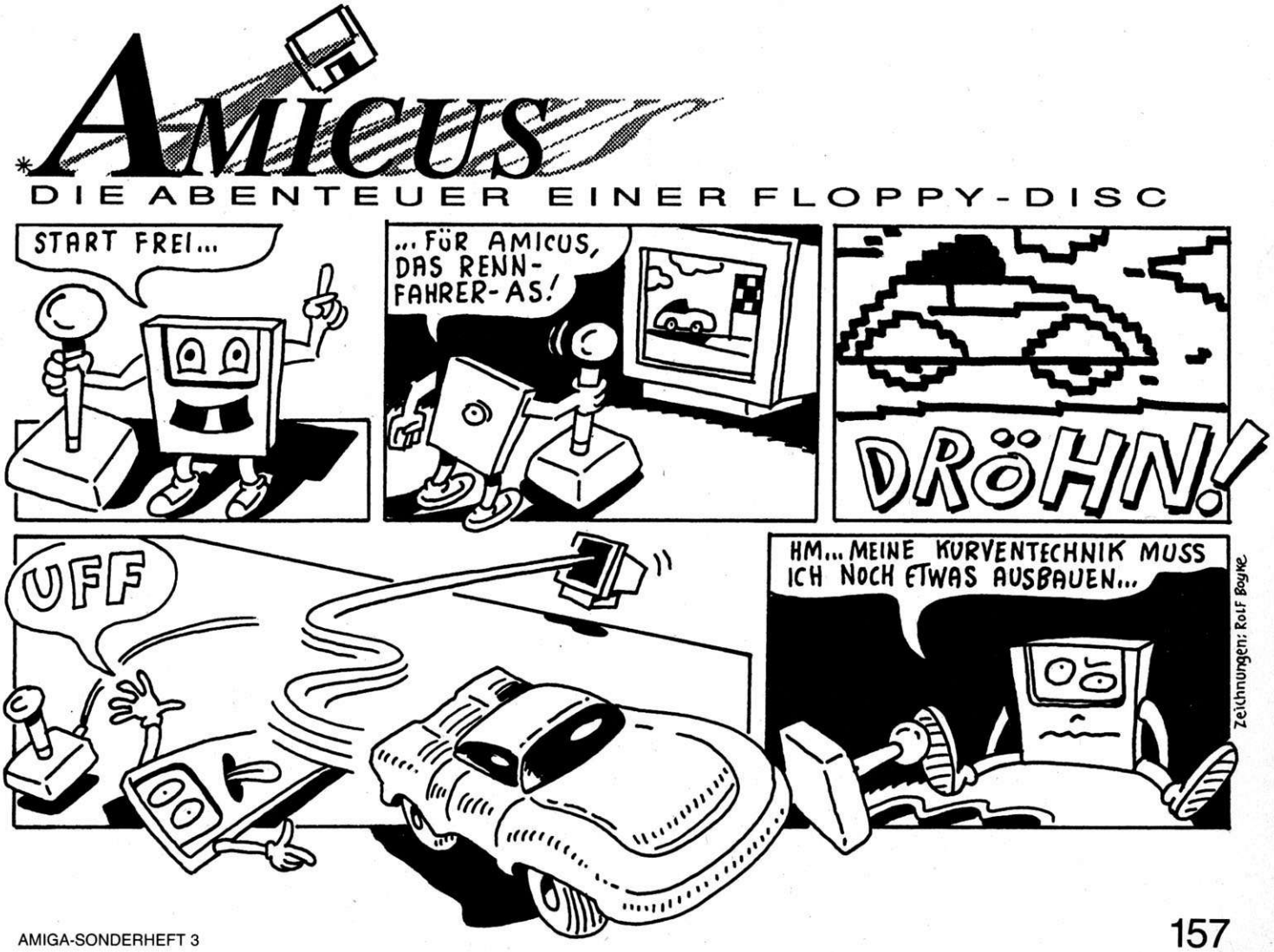
AMIGA-Sonderheft 2, Seite 90, »Kopieren mit vielen Extras«
In den Listingteil zu Supercopy haben sich folgenschwere Fehler eingeschlichen. Durch einen Fehler im Konvertierungsprogramm, das die Zeilennummern und die Checksummen generiert, fehlen alle Zeichen, die einzeln in einer Zeile stehen. Für die Leser, die nur »Supercopy« haben möchten, und die zahlreichen anderen Programme der Programmservice-Diskette nicht benötigen, bieten wir folgenden Service an: Schicken Sie uns eine formatierte Leerdiskette. Wir kopieren dann das lauffähige, compilierte Programm auf diese und schicken Ihnen die Diskette zurück. Schicken Sie die Diskette bitte an die Redaktion Sonderhefte, Stichwort Supercopy, Hans-Pinsel-Str. 2, 8013 Haar bei München. Vergessen Sie auf keinen Fall Ihren Absender (am besten diesen auf die Diskette schreiben).

AMIGA-Sonderheft 2, Seite 159, »Checksummer«
Beim Einlesen einer Datei, die größer als 32000 Zeichen ist, kommt es in der abgedruckten Version 1.1 des Checkie_42 zu Problemen. Die Zeile 122 des »Checkie 42« ist folgendermaßen zu modifizieren:

```
PRINT#2,e$;:flen=flen-32000
```

Nach dieser kleinen Änderung klappt das Einlesen von großen Dateien. Die Änderung ist übrigens auch bei der im AMIGA-Magazin 1/89 abgedruckten Version vorzunehmen.

AMIGA-Sonderheft 2, Seite 81, »Floppy-Speeder« mit Niveau
Leider hat sich inzwischen herausgestellt, daß »FastLoadCopy« mit dem A1000 in der PAL-Version nicht funktioniert. Es fehlen dem Programm zirka 3 KByte Speicherplatz... Eine Änderung können wir in diesem Fall leider noch nicht präsentieren. Vielleicht wissen Sie Rat? Bitte teilen Sie uns mit, wenn Ihnen eine Lösung einfällt. *(rs)*



BÜCHER

Amiga Basic

Basic ist »die« Programmiersprache für Heim-Computer. Zwar verdankt diese Sprache ihre hohe Verbreitung weniger einzigartiger Qualität, sondern mehr dem Umstand, daß Basic mit jedem neu verkauften Computer ausgeliefert wird. Dennoch ist Basic immer noch die Programmiersprache, in der jeder schon irgendwann einmal erste Erfahrungen gesammelt hat.

Wozu dann überhaupt noch Worte darüber verlieren? Amiga-Basic ist nicht irgendein Basic. Es bietet eine so große Vielfalt, daß auch erfahrene Basic-Programmierer anfangs von den sich bietenden Möglichkeiten wie erschlagen sind. Ganz zu schweigen von Einsteigern, die noch keine Erfahrungen mit Programmiersprachen haben.

Eine erstklassige Einführung in Amiga-Basic findet der Anfänger oder Umsteiger in »Amiga Basic« von Christian Spanik und Hannes Rügheimer. Die beiden Autoren beschreiben in flottem Ton den gesamten Sprachumfang. Die Einführung ist so geschrieben, daß kein Anfänger überfordert ist. Trotzdem werden sich auch »alte Hasen« nicht gleich gelangweilt abwenden. Denn dieses Buch ist so erfrischend locker geschrieben, daß das Lesen einfach Spaß macht.

Allerdings wird kaum jemand dieses Buch »just for fun« lesen. Was kann es in puncto Information bieten? Da gibt es einmal einen breit angelegten Basic-Kurs, der sich voll an den Bedürfnissen von Ein- oder Umsteigern orientiert. Das ganze Spektrum von Amiga-Basic wird ausführlich anhand von Beispielen beschrieben. Es ist sicher kein Nachteil, daß diese Beispiele gut aufeinander abgestimmt sind und nach dem Baukastenprinzip ein komplettes Programm ergeben. Beispielsweise wird eine Dateiverwaltung entwickelt, deren Daten im nächsten Kapitel grafisch ausgegeben werden.

Mit seinem neu erworbenen Wissen wird der frischgebackene Basic-Programmierer allerdings noch lange nicht alleine gelassen. Es fällt angenehm auf, daß die Autoren auch an Probleme gedacht haben, die bei den ersten eigenen Gehversuchen auftauchen. Damit diese nicht zu Stolperschritten geraten, finden sich im Anhang nochmals etwa 200 (!) sehr hilfreiche Seiten. Dazu gehören genaue Beschreibungen der Fehlermeldungen mit möglichen Abhilfen und ein Referenzteil mit Kurzbeschreibungen jedes Befehls.

»Amiga Basic« kann jedem Einsteiger, der an der Programmierung in Basic interessiert ist, nur wärmstens empfohlen werden. Er wird schwerlich ein Buch finden, das ihm das Programmieren in Basic auf angenehmere Art nahebringt und dabei noch einen soliden Referenzteil bietet, den er auch als weit Fortgeschrittener nicht gleich aus der Hand legt. *(so)*

AMIGA BASIC, Rügheimer, Spanik, Data Becker GmbH, 774 Seiten, ISBN 3-89011-209, Preis 59 Mark.

Das große Amiga-Spielebuch

Der Amiga ist zum Spielen geschaffen. Das beweisen zahlreiche Superspiele, die bisher schon veröffentlicht wurden. Doch wer kann schon die Spieleflut überblicken, die seit dem Erscheinen dieses Traumcomputers hereingebrochen ist? Hilfe verspricht hier »Das große Amiga-Spielebuch« der Berliner Firma »technicSupport«. Die verschiedenen Autoren vermitteln zu jedem vorgestellten Spiel eine ausführliche Beschreibung. Der Spieler wird quasi an der Hand genommen und durch das Spiel geführt.

Die Spielbeschreibungen beschränken sich keineswegs darauf, dem Leser den Mund wäßrig zu machen. Sie geben sehr präzise Anweisungen, bis zu genauen Tastenkombinationen, die das Spielen stark erleichtern. An einigen Stellen ist es den Autoren sogar gelungen, Tastenkombinationen zu finden, die im Handbuch nicht dokumentiert sind.

Im Vorwort verweisen die Herausgeber darauf, daß sie mit diesem Buch keineswegs den Raubkopierern das Leben erleichtern wollen. Statt dessen soll dem potentiellen Käufer mit einer ausführlichen Beschreibung die Entscheidung erleichtert werden. Ob dazu wirklich eine derart detaillierte Beschreibung geeignet ist, bleibt fraglich. Vielleicht wäre eine Marktübersicht für diese Zielgruppe besser gewesen.

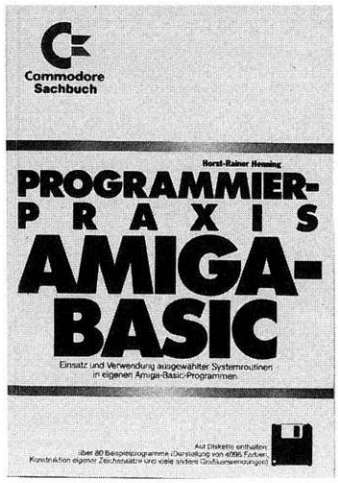
Zweifellos ist das Spielebuch aber eine recht gelungene Hilfestellung zu wichtigen Spielen für den Amiga. Der interessierte Spieler erhält ausführliche Anleitungen, die allerdings von unterschiedlicher Qualität sind. So wechselt brillanter Stil (beispielsweise bei Championship Golf) mit recht holprig zu lesendem Stoff (Chessmaster und Shanghai).

Das Buch ist eine größtenteils leicht zu lesende Anleitung für viele gute Amiga-Spiele. Fraglich bleibt am Ende nur die Zielgruppe. Nachdem die Besitzer von Raubkopien hiervon ausdrücklich ausgenommen sind, bleiben eigentlich nur die Amiga-Benutzer, welche die — häufig englische — Anleitung nicht verstehen, oder denen diese auf irgendeine Art abhanden kam. *(so)*

Das große AMIGA Spielebuch, Schmidt/Hertwig (Hrsg.), technic Support, ISBN 3-926847-02-6, Preis 49 Mark.

Programmierpraxis Amiga-Basic

Der Amiga unterstützt den Programmierer mit einer großen Anzahl Systembibliotheken. Das sind speziell angepaßte Routinen, die deutlich schneller sind, als die gewohnten Basic-Befehle. Der Nachteil ist lediglich, daß man von Basic aus nicht so einfach darauf zugreifen kann, wie von anderen Sprachen. »Programmierpraxis Amiga-Basic« hilft, dieses Problem zu umgehen.



Nach einer kurzen Einführung, welche die am dringendsten benötigten Basic-Befehle wie »CALL« und »LIBRARY« vorstellt, findet sich der Leser sofort bis über beide Ohren im System. Zu Beginn werden in Grundlagenkapiteln die Grafikfunktionen, die Speicherverwaltung und die Ein-/Ausgabe behandelt. Darauf folgen die Beschreibungen der wichtigsten Routinen und deren Aufruf von Basic aus.

Dem Basic-Programmierer stellen sich größere Probleme in den Weg, als Anhängern anderer Programmiersprachen. Allerdings hat sich inzwischen herumgesprochen, daß man auch von Basic aus Zugriff auf das Betriebssystem erhält, wenn man die entsprechenden Tricks und Schliche kennt.

Eben diese werden von dem vorliegenden Buch vermittelt. Der Stoff wird mit zahlreichen Beispielen unterlegt, welche die verwendeten Routinen demonstrieren. Alle Programme sind auch auf der beigelegten Diskette zu finden, so daß man sich mühsames Abtippen sparen kann. Eine weitere Erleichterung bietet das umfangreiche Stichwortverzeichnis, das ein schnelles Auffinden benötigter Routinen ermöglicht.

»Programmierpraxis Amiga-Basic« ist ein Buch für fortgeschrittene Basic-Anwender. Wer schon etwas Basic-Erfahrung mitbringt und aus dieser Sprache alles herausholen möchte, liegt mit diesem Buch genau richtig. *(so)*

Horst-Rainer Henning, Programmierpraxis Amiga-Basic, Commodore-Sachbuch (Markt & Technik-Verlag), 367 Seiten, ISBN 3-89090-434-3, Preis: 59 Mark

Checksummer

Dieses Programm ist unentbehrlich beim Abtippen unserer Listings. Es hilft, Tippfehler zu vermeiden und spart viel Zeit.

Ein längeres Listing ohne Fehler abzutippen ist (fast) unmöglich. Aus diesem Grund haben wir in Ausgabe 3/88 des AMIGA-Magazins eine Eingabehilfe — den Checksummer »Checkie 42« veröffentlicht. Die hier vorgestellte Version 1.1 enthält erweiterte Funktionen und bietet mehr Komfort. Damit möglichst viele unserer Leser dieses Programm auch tatsächlich anwenden, haben wir es möglichst kurz gehalten und in einer Sprache programmiert, die alle Abtipper besitzen: Amiga-Basic.

Die Form der Listings

Die Listingzeilen bestehen aus einer bis zu vierstelligen Zeilennummer, der zwei- beziehungsweise dreistelligen Prüfsumme und der eigentlichen Programmzeile. Beispiel:

```
10 TTO print "Hallo!"
   └─Prüfcode
└──────Zeilennummer
```

Nach einer Leerstelle im Anschluß an die Zeilennummer stehen bis zu drei Zeichen Prüfcode. Die einzelnen Zeichen können sein eine Ziffer (»0« bis »9«), ein kleiner Buchstabe (»a« bis »z«) oder ein Großbuchstabe (»A« bis »Z«). Die ersten beiden Zeichen der Prüfsumme sind der eigentliche Prüfcode. Im dritten Zeichen ist die Spaltenposition der ersten »Nicht-Leerstelle« verschlüsselt. Das ist für diejenigen Anwender interessant, welche die Struktur des Listings, also die Einrückungen durch Leerzeichen, übernehmen wollen. Ist dies nicht Ihre Absicht, können Sie die Eingabe der Checksumme schon nach den ersten beiden Zeichen mit <Return> abschließen. Bei Checkie 42 muß die Groß- und Kleinschreibung so wie im Listing abgedruckt übernommen werden.

Eingabehinweis:
Geben Sie »Checkie 42« (Version 1.1) bitte mit Checkie 42 aus der AMIGA-Ausgabe 3/88 oder 12/87 ein. Sollten Sie die alte Eingabehilfe nicht besitzen, so tippen Sie das Listing für die Version 1.1 im normalen Basic-Editor ohne Prüfsummen und Zeilennummern ab.

Der Umgang mit Checkie 42

Nach dem Start fragt das Programm nach einem Dateinamen. Unter dem angegebenen Namen speichert Checkie 42 die eingegebenen Listingzeilen ab. Existiert bereits eine Datei mit diesem Namen auf der Diskette, so haben Sie mit der Abfrage »Nur Checksummer ausgeben?« zwei Möglichkeiten:

<j> Ausgabe der Datei mit Checksumme auf den Bildschirm oder den Drucker.
<n> Einlesen der Programmzeilen aus der vorhandenen Datei und Eingabe der Checksumme mit der Tastatur.

Beide Alternativen sind gedacht für Anwender, die ein Listing nicht mit dem Zeileneditor des Checkie, sondern mit einem schnelleren und/oder komfortableren Editor ihrer Wahl — zum Beispiel dem Editor von Amiga-Basic (mit »..",a« speichern) erfaßt haben.

Checkie 42 errechnet nach der Eingabe <j> die Prüfsummen Ihres Textes und Sie können diese dann mit dem Listing im AMIGA-Magazin vergleichen. Bei der Ausgabe auf den Bildschirm schreibt das Programm die Programmzeilen inklusive Checksummen zusätzlich in eine Datei auf Diskette mit dem Zusatz ».chk«. Diese können Sie später zum Beipiel mit dem CLI-Befehl TYPE ohne erneute Berechnung der Prüfsumme noch einmal ausgeben.

Haben Sie »Nur Checksumme ausgegeben?« mit »n« beantwortet, dann können Sie dem Programm den Vergleich überlassen, in dem die Frage »Eingabe aus Datei« mit »j« beantwortet wird. Dann brauchen Sie nur noch die Checksummen eingeben. Der Checksummer holt sich die Zeile aus der angegebenen Datei statt von der Tastatur. Entspricht die eingegebene Prüfzahl nicht der errechneten, kann die Zeile gleich korrigiert werden.

Beantworten Sie obige Frage mit »n«, zählt Checkie die in der Datei vorhandenen Zeilen und wartet mit der Zeilennummer »Anzahl+1« auf die Eingabe einer neuen Zeile. Alle weiteren Eingaben hängt das Programm an die bestehende Datei an. Diese Funktion ist sinnvoll, wenn Sie ein Listing in mehreren Teilen abtippen wollen.

So tippen Sie Listings ab

Haben Sie anfangs einen Dateinamen eingegeben oder Sie hängen neue Zeilen an eine bestehende Datei an, dann arbeiten Sie im normalen Eingabemodus. Checkie 42 schlägt dabei eine Zeilennummer vor und wartet auf die Prüfsumme. Nach Eingabe derselben taucht der Cursor zwischen den zwei Trennstrichen auf. Dort muß nun die Zeile »ohne« Zeilennumer und Prüfsumme eingegeben werden. Nach Betätigen der Taste <Return> berechnet Checkie die Prüfsumme. Leerstellen vor und hinter der Programmanweisung werden ignoriert. Stimmen Programmzeile und Prüfsumme mit derjenigen im Listing überein, speichert der Checksummer die Eingabe ab und wartet auf die nächste Zeile.

Einfügemodus: Wahrscheinlich wird eine abgetippte Zeile mal einen Fehler enthalten. Checkie 42 positioniert den Cursor dann an den Anfang der Zeile und wartet auf die korrekte Eingabe. Korrekturen lassen sich mit der Backspace- oder Delete-Taste durchführen. Um Zeichenfolgen einzufügen, kann kurzfristig mit <F2> der Einfügemodus eingeschaltet werden. Dieser Modus sollte allerdings nach der Fehlerkorrektur wieder ausgeschaltet werden, da er die Eingabe verlangsamt.

Sonderfall-Prüfsumme ignorieren: Möchten Sie zum Beispiel eine Kommentarzeile nicht »original« übernehmen, läßt sich trotz einer falschen Prüfsumme eine Übernahme der Zeile mit der Funktionstaste <F6> erzwingen. Sie können damit aber auch falsche Programmzeilen übernehmen. Verwenden Sie deshalb die Taste <F6> nicht gewohnheitsmäßig. Der Checksummer teilt Ihnen nach Beenden des Programms mit, wieviel Zeilen er ungeprüft übernommen hat.

Prüfsumme und Zeilennummer ändern: Natürlich kann es auch vorkommen, daß die Programmzeile zwar richtig abgetippt wurde, sich bei der Prüfsumme aber ein Fehler eingeschlichen hat. Nach Betätigen von <F1> kann die Prüfsumme korrigiert werden. Während der Eingabe der Prüfsumme läßt sich mit <F7> die vom Programm vorgeschlagene Zeilennummer verändern. Damit können Sie gezielt nur bestimmte Teile eines Listings übernehmen.

Haben Sie eine mit einem anderen Editor geschriebene Programmdatei überprüft und nur in wenigen Zeilen Fehler festgestellt, lassen sich durch Vorgabe der Nummern diese Zeilen gezielt ändern. Bei Angabe der Zeilennummer in aufsteigender Reihenfolge benötigt das Programm übrigens erheblich weniger Zeit für die Suche der Zeilen in der jeweiligen Datei. Um die versehentliche Übernahme fehlerhafter Zeilen zu verhindern, sperrt das Programm bei fehlender Übereinstimmung der Prüfsummen die Taste <F7> (Änderung der Zeilennummer).

Fehlerfrei abtippen

Eingabe beenden: Die Kombination <Ctrl-e> beendet den Programmlauf nach vollständiger Eingabe des Listings oder für eine Unterbrechung.

Am Schluß noch ein Tip für alle Leser, denen unser Basic-Editor zu langsam ist. Die Berechnung der Prüfsummen erfolgt im Unterprogramm »CalcSumme«. Dieser Teil ist sehr einfach in schnellere Sprachen, wie beispielsweise C, umsetzbar.

Wer schon einmal Fehler in einem abgetippten Listing gesucht hat, der weiß, wie frustrierend diese Arbeit sein kann. Nutzen Sie deshalb den »Checkie 42«. Sie sparen viel Zeit und müssen sich nicht dauernd auf die Suche nach tückischen Fehlern begeben.

(Dieter Behlich/kn)

```
 1 CqO REM ************************************
 2 KF  REM *** Checksummer: Checkie 42       ***
 3 4c  REM ***          Version 1.1           ***
 4 Ft  REM ************************************
 5 Z7  Start:
 6 W42   GOSUB Init
 7 Pn    GOSUB OpenDatei
 8 mb    IF dn$="" THEN Ende
 9 b2    GOSUB Bild
10 OwO NeueZeile:
11 K42   GOSUB loeschen
12 zT    GOSUB EingabeSumme
13 pt    IF FEnde=wahr THEN Ende
14 MVO Wiederholung:
15 sQ2   GOSUB EingabeZeile
16 sw    IF FEnde=wahr THEN Ende
17 mO    GOSUB CalcSumme
18 fB    IF FZok = falsch THEN GOTO Wiederholung
19 oC    GOSUB Uebernahme
20 iv    GOTO NeueZeile
21 mAO Ende:
22 7S2   GOSUB fertig
23 72    END
24 UGO Init:
25 fD2   wahr=-1
26 fy    falsch=0
27 Kh    LZeile=240  : REM Anzahl Zeichen/Zeile
28 tz    LBZeile=60  : REM Anzahl Zeichen/Bildschirmzeile
29 OG    AnzBZeilen=LZeile/LBZeile
30 Nl    zx=6 : zy=14   : REM Position Zeile
31 PC    sx=20 : sy=5   : REM Position Checksumme
32 mF    AnzCsZ=3       : REM Anzahl Ziffern/Checksumme
33 QF    DIM z(LZeile)
34 05    DIM cs(AnzCsZ*2)
35 iz    cs(AnzCsZ)=0   : REM Zeilenstart
36 gv    a=0 : b=0 : c=0  : REM Hilfsvariablen
37 AI    i=0 : j=0 : k=0  : REM Zählvariablen
38 XL    FZok = wahr : FCz3=100
39 Wn    Checkfile=0  : Zeile=1
40 Bb    READ Faktor(i)
41 X1    WHILE Faktor(i)<>0
42 ga4     i=i+1
43 cm      READ Faktor(i)  : REM Faktorenreihe
44 9x2   WEND
45 Gm    AnzFak=i
46 RB    DATA 2,3,4,5,6,0
47 7j    RETURN
48 PWO OpenDatei:
49 BR2   ON ERROR GOTO Dateifehler
50 MS    CLS
51 Vu    LOCATE 3,5:dn$="":INPUT "Dateiname: ";dn$
52 Fp    OPEN dn$ FOR INPUT AS #1
53 KU    IF FFehler=53 THEN NeueDatei
54 4s    LOCATE 5,5:PRINT "Nur Checksummen ausgeben? (j/n) ":e$=INP
         UT$(1)
55 gM    IF e$="j" THEN
56 Kq4     LOCATE 7,5:e$="":PRINT "Bildschirm oder Drucker? (b/d) "
           :e$=INPUT$(1)
57 Ku      IF e$="d" THEN
58 KM7        OPEN "prt:" FOR OUTPUT AS #4
59 YH4     ELSE
60 7Y6       OPEN dn$+".chk" FOR OUTPUT AS #4
61 vo4     END IF
62 RR      WHILE NOT EOF(1)
63 yj6       LINE INPUT#1,e$
64 fg        apos=LEN(e$)
65 to        z(apos)=32
66 4L        FOR i=1 TO apos
67 sq8         z(i-1)=ASC(MID$(e$,i,1))
68 Xn6       NEXT i
69 cE        GOSUB CalcSumme
70 vp        i=0
71 Rh        WHILE z(i)=32 : i=i+1 : WEND
72 4v        IF FCz3=i THEN
73 f98         cs(6)=-16    : REM Leerstelle
74 nW6       ELSE
75 XA8         cs(6)=i : FCz3=i
76 A36       END IF
77 2o        PRINT USING "# # # #";Zeile; : PRINT " ";
78 Wf        PRINT#4,USING "# # # #";Zeile; : PRINT#4," ";
79 mC        FOR i=4 TO 6
80 7d8         IF cs(i)>35 THEN cs(i)=cs(i)+6
81 nS          IF cs(i)>9 THEN cs(i)=cs(i)+7
82 jy          PRINT CHR$(cs(i)+48);
83 8L          PRINT#4,CHR$(cs(i)+48);
84 n36       NEXT i
85 19        PRINT " ";e$
86 tl        PRINT#4," ";e$
87 Qm        Zeile=Zeile+1
88 rf4     WEND
89 IF      CLOSE 1 : CLOSE 4
```

```
90 H4      dn$=""
91 Fv      PRINT:PRINT "Taste drücken"
92 5r      e$=INPUT$(1)
93 rT      RETURN
94 SL2   END IF
95 7O    LOCATE 7,5:PRINT"Eingabe aus Datei? (j/n) ":e$=INPUT$(1)
96 L1    IF e$="j" THEN
97 u24     Checkfile=1
98 O8      CLOSE 1
99 eX      GOSUB backup
100 WZ     OPEN dn$+".bak" FOR INPUT AS #1
101 Sk     OPEN dn$ FOR OUTPUT AS #2
102 Oc     RETURN
103 bU2  END IF
104 77   WHILE NOT EOF(1)
105 eP4    LINE INPUT#1,e$
106 j5     Zeile=Zeile+1
107 Ay2  WEND
108 YI   CLOSE 1
109 8AO NeueDatei:
110 aM2  OPEN dn$ FOR APPEND AS #2
111 9l   RETURN
112 7QO Dateifehler:
113 Og2  FFehler=ERR
114 PB   IF FFehler=53 THEN RESUME NEXT
115 vy   ON ERROR GOTO 0
116 aDO backup:
117 yY2  OPEN dn$ FOR INPUT AS#1
118 Sj   OPEN dn$+".bak" FOR OUTPUT AS #2
119 8F   flen=LOF(1)
120 ih   WHILE flen>=32000
121 il4    e$=INPUT$(32000,1)
122 tL     PRINT#2,e$;:flen=flen-32000
123 QE2  WEND
124 e7   IF flen THEN
125 SB4    e$=INPUT$(flen,1)
126 xP     PRINT#2,e$;
127 zs2  END IF
128 zg   CLOSE 2  : CLOSE 1
129 R3   RETURN
130 d00 Bild:
131 f12  CLS
132 Gi   LOCATE 2,10
133 yY   PRINT ">>>>>>> Checkie 42 - Der ultimative Checksum
         mer! <<<<<<<"
134 xR   LOCATE 13,zx
135 Le   FOR i=1 TO 2
136 kx4    FOR j=1 TO LBZeile
137 HR6      PRINT "-";
138 hy4    NEXT j
139 H1     LOCATE 13+AnzBZeilen+1,zx
140 hx2  NEXT i
141 Hi   LOCATE 5,30:PRINT "<CTRL e> = Programm beenden"
142 vd   LOCATE 7,8:PRINT "<F2> = Einfügemodus ein"
143 uQ   LOCATE 7,40:PRINT "<F1> = Checksumme ändern"
144 1H   LOCATE 9,40:PRINT "<F6> = Zeile speichern"
145 ME   LOCATE 9,8:PRINT "<F3> = Zeile löschen"
146 v5   LOCATE 11,8:PRINT "<F7> = Zeilennummer ändern"
147 jL   RETURN
148 5f0 loeschen:
149 w82  LOCATE zy,1
150 8S   FOR i=1 TO AnzBZeilen
151 jf4    PRINT TAB(zx);SPACE$(LBZeile)
152 t92  NEXT i
153 OO   FOR i=0 TO LZeile
154 V84    z(i)=32
155 wC2  NEXT i
156 3F   LOCATE zy,1
157 om   PRINT USING "# # # #";Zeile
158 dy   apos=0
159 vX   RETURN
160 v50 EingabeSumme:
161 va2  Farbe=0 : i=1
162 AB   LOCATE sy,sx+1
163 Xm   PRINT "..."
164 O40 blinken:
165 aC2  Farbe=Farbe XOR 1
166 QH   COLOR Farbe,0
167 P6   LOCATE sy,sx-15
168 1n   PRINT "Checksumme:"
169 ni   LOCATE sy,sx+1
170 ya   e$=INKEY$
171 lT   IF e$="" THEN blinken
172 Xp   COLOR 1,0
173 dl   e=ASC(e$)
174 fM   IF e=5 THEN
175 KU4    FEnde=wahr   : REM <CTRL e>
176 hf     i=AnzCsZ+1   : REM fertig
177 Cc2  ELSEIF e=135 THEN
178 dc4    GOSUB NeuZeile
179 Bz     e=135
```

```
180 LB2   ELSEIF e=8 THEN
181 4h4     IF i>1 THEN
182 sZ6       i=i-1      : REM <BS>
183 XY        LOCATE sy,sx+i : PRINT "."
184 un4     END IF
185 Tn2   ELSEIF e=13 THEN
186 KZ4     IF i=AnzCsZ THEN i=AnzCsZ+1 : REM <CR>
187 cL2   ELSE
188 ra4     IF e>47 AND e<58 THEN
189 vi6       e=e-48                  : REM 0-9
190 AI4     ELSEIF e>64 AND e<91 THEN
191 Rv6       e=e-55                  : REM A-Z
192 jX4     ELSEIF e>96 AND e<123 THEN
193 Zx6       e=e-61                  : REM a-z
194 jS4     ELSE
195 yz6       GOTO blinken            : REM weder noch
196 6z4     END IF
197 JY      PRINT e$;
198 ve      cs(i)=e
199 D7      i=i+1
200 A32   END IF
201 wn    IF i<=AnzCsZ THEN blinken
202 f50 ESEnde:
203 2K2   COLOR 1,0
204 Oh    LOCATE sy,sx-15
205 cO    PRINT "Checksumme:"
206 gI    RETURN
207 aE0 NeuZeile:
208 dU2   IF FZok = wahr THEN
209 P44     NeuZeile=0
210 nn      WHILE e<>13 OR NeuZeile=0
211 Ux6       LOCATE zy,1:PRINT USING "####";NeuZeile;
212 oF        e=ASC(INPUT$(1))
213 gP        IF e>47 AND e<58 THEN NeuZeile=NeuZeile*10+e-48
214 Jn        IF NeuZeile > 9999 THEN e=8
215 in        IF e=8 THEN NeuZeile=INT(NeuZeile/10)
216 vj4     WEND
217 kh      IF Checkfile THEN
218 tQ6       IF NeuZeile < Zeile THEN
219 yy8         WHILE NOT EOF(1)
220 VGA           LINE INPUT#1,e$
221 cd            PRINT#2,e$
222 1p8         WEND
223 MH          CLOSE 1 : CLOSE 2
224 fY          GOSUB backup
225 Xa          OPEN dn$+".bak" FOR INPUT AS #1
226 Tl          OPEN dn$ FOR OUTPUT AS #2
227 9t          Zeile=1
228 cV6       END IF
229 Yp        WHILE (NeuZeile > Zeile) AND (NOT EOF(1))
230 fQ8         LINE INPUT#1,e$
231 mn          PRINT#2,e$
232 17          Zeile=Zeile+1
233 C06       WEND
234 cR        IF EOF(1) THEN
235 bI8         CLOSE 1
236 8i          NeuZeile=Zeile
237 uN          LOCATE zy,1:PRINT USING "####";NeuZeile;
238 9G          Checkfile=0
239 ng6       END IF
240 oh4     END IF
241 vV      Zeile=NeuZeile
242 qj2   END IF
243 Ht0 RETURN
244 1H  EingabeZeile:
245 n72   x=cs(AnzCsZ)
246 BW0 weiter:
247 ga2   cy=zy+INT(x/LBZeile):cx=zx+(x MOD LBZeile)
248 vy    LOCATE cy,cx
249 o6    COLOR 0,1
250 mI    PRINT CHR$(z(x));
251 y1    LOCATE cy,cx
252 68    IF x>apos THEN apos=x
253 xx    IF Checkfile AND FZok THEN
254 wl4     IF EOF(1) THEN
255 0R6       Checkfile=0 : CLOSE 1
256 jS4     ELSE
257 ix6       e$=INPUT$(1,1)
258 6z4     END IF
259 mV2   ELSE
260 Q24     e$=INKEY$
261 922   END IF
262 gp4     IF e$="" THEN weiter
263 0I      COLOR 1,0
264 0W      PRINT CHR$(z(x));
265 CF      LOCATE cy,cx
266 8G      e=ASC(e$)
267 UR2   IF ((e AND 127)<32) OR e=127 THEN Controlcode
268 op    IF imode THEN GOSUB insert
269 I6    PRINT e$
270 Ki    z(x)=e : e=30
```

```
271 3D0 Controlcode:
272 gF2   IF e=13 OR e=10 THEN
273 1N4     RETURN
274 092   ELSEIF e=30 THEN
275 2p4     a=1
276 Ea2   ELSEIF e=29 THEN
277 034     a=LBZeile
278 6G2   ELSEIF e=31 THEN
279 DD4     a=-1
280 Gb2   ELSEIF e=28 THEN
281 Rg4     a=-LBZeile
282 9s2   ELSE
283 UX4     GOTO noCrs
284 WP2   END IF
285 hz    x=x+a
286 5Z    IF x>=0 AND x<LZeile THEN weiter
287 tD    x=x-a
288 HB    GOTO weiter
289 kE0 noCrs:
290 dN2   IF e=8 THEN
291 1s4     IF x>0 THEN
292 Kk6       x=x-1
293 AM        LOCATE zy+INT(x/LBZeile),zx+(x MOD LBZeile)
294 Bb        FOR i=x TO apos
295 788         z(i)=z(i+1)
296 Ja          PRINT CHR$(z(i));
297 Z0          IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
298 FV6       NEXT i
299 pr        z(apos)=32 : PRINT " "
300 yR        apos=apos-1
301 ng4     END IF
302 Hi2   ELSEIF e=127 THEN
303 Kk4     FOR i=x TO apos
304 GH6       z(i)=z(i+1)
305 Sj        PRINT CHR$(z(i));
306 i9        IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
307 0e4     NEXT i
308 y0      z(apos)=32 : PRINT " "
309 7a      apos=apos-1
310 Vy2   ELSEIF e=129 THEN
311 oI4     GOSUB EingabeSumme
312 sC      x=cs(AnzCsZ)
313 9U2   ELSEIF e=130 THEN
314 513     imode=imode XOR 1
315 tU5       LOCATE 7,28
316 yZ3     IF imode THEN
317 iT5       PRINT "aus"
318 jS3     ELSE
319 YV5       PRINT "ein"
320 6z3     END IF
321 Kg2   ELSEIF e=131 THEN
322 L54     GOSUB loeschen
323 3N      x=cs(AnzCsZ)
324 7Y2   ELSEIF e=134  THEN
325 bD4     RETURN
326 PR2   ELSEIF e=5 THEN
327 ZT4     FEnde=wahr
328 eG      RETURN
329 F82   END IF
330 xr    GOTO weiter
331 If0 insert:
332 oJ2   IF apos>x THEN
333 3o4     FOR i=apos TO x STEP -1
334 Qt6       z(i+1)=z(i)
335 q64     NEXT i
336 PH      z(x)=32
337 Ny      apos=apos+1
338 Kp      IF apos=LZeile THEN apos=apos-1:z(LZeile)=32
339 uK      FOR i=x TO apos
340 1I6       PRINT CHR$(z(i));
341 H1        IF i MOD LBZeile=59 THEN PRINT:PRINT TAB(zx);
342 xD4     NEXT i
343 yA      LOCATE zy+INT(x/LBZeile),zx+(x MOD LBZeile)
344 UN2   END IF
345 vX    RETURN
346 cX0 CalcSumme:
347 3s2   a=0 : b=0 : c=0
348 z2    IF e=134 THEN
349 pv4     FZok=wahr
350 WX      FF6=FF6+1
351 Gz2   ELSE
352 hv4     WHILE z(apos)=32 AND apos>0
353 pI6       apos=apos-1
354 9x4     WEND
355 Ni      IF apos>0 THEN
356 036       WHILE z(c)=32
```

Listing. Der verbesserte »Checkie 42«. Bitte mit der ersten Version von »Checkie 42« oder ohne Prüfsummen und Zeilennummern eingeben.

```
357 Bt8         c=c+1
358 D16       WEND
359 jc4     END IF
360 EJ      FOR i=c TO apos
361 Y26       j=(i-c) MOD AnzFak
362 Qs        k=(i+1-c) MOD AnzFak
363 s1        a=a+((z(i) AND 127)-32)*Faktor(j)
364 2E        b=b+((z(i) AND 127)-32)*Faktor(k)
365 Ka4     NEXT i
366 pA      cs(4)=a+Zeile-INT((a+Zeile)/62)*62
367 4N      cs(5)=b+Zeile-INT((b+Zeile)/62)*62
368 4Q      FZok=(cs(1)=cs(4)) AND (cs(2)=cs(5))
369 tm2   END IF
370 Kw    RETURN
371 pM0 Uebernahme:
372 xD2   FOR i=0 TO apos
373 ih4     PRINT#2,CHR$(z(i));
374 Tj2   NEXT i
375 VZ4     PRINT#2,""
376 5R2   Zeile=Zeile+1
377 R3    RETURN
378 N90 fertig:
379 MJ2   IF Checkfile THEN
380 ZZ4     WHILE NOT EOF(1)
381 6r6       LINE INPUT#1,e$
382 DE        PRINT#2,e$
383 cQ4     WEND
384 0k      CLOSE 1
385 922   END IF
386 5q    CLOSE 2
387 nt    CLS
388 Rg    LOCATE 12,35
389 WL    PRINT "F E R T I G !!!"
390 I9    LOCATE 20,1
391 D1    IF FF6<>0 THEN
392 Ci4     PRINT "ACHTUNG!!!  ";
393 9H      PRINT FF6;" Zeile(n) wurde(n) ungeprüft gespeichert."
394 IB2   END IF
395 jL    RETURN
(C) 1988 M&T
```

Listing. Der verbesserte »Checkie 42« (Schluß)

Impressum

Herausgeber: *Carl-Franz von Quadt, Otmar Weber*

Chefredakteur: *Hans-Günther Beer*
stellv. Chefredakteur: *Gottfried Knechtel – verantwortlich für den redaktionellen Teil*
Chef vom Dienst: *Susanne Kirmaier*
Leitender Redakteur: *Klaus Schrödl*
Redaktion: *Ralf Sablowski, Klaus Sonnenleiter*
Redaktionsassistenz: *Andrea Kaltenhauser, Brigitte Bobenstetter, Sylvia Sailer, Helga Weber (202)*
Mitarbeiter der Redaktion: *Martin Jobst, Andreas Lietz, Christian Buchner, Nikolaus Huber, Reinhold Brings*

Alle Artikel sind mit dem Kennzeichen des Redakteurs (kn = Gottfried Knechtel, sk = Klaus Schrödl, rs = Ralf Sablowski, so = Klaus Sonnenleiter) und/oder mit dem Namen des Autors/Mitarbeiters gekennzeichnet

Art-director: *Friedemann Porscha*
Layout: *Erich Schulze (Cheflayouter), Rudolf Weikl*
Fotografie: *Sabine Tennstaedt, Ilona Wiewiorra*
Titelgestaltung: *Friedemann Porscha*
Spritzgrafik: *Norbert Raab*
Computergrafik: *Werner Nienstedt*

Auslandsrepräsentation:

Schweiz: *Markt&Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042-41 56 56, Telex: 862329 mut ch*

USA: *M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94063, Telefon: (4 15) 366-3600, Telex 752-351*

Österreich: *Markt&Technik Ges. mbH Hermann Raniger, Große Neugasse 28, A 1040-Wien, Tel. 0043-222-8579455, Telex: 047-132532*

Manuskripteinsendungen: *Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt&Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt&Technik Verlag AG verlegten Publikationen und dazu, daß Markt&Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.*

Produktionsleiter: *Klaus Buck (180)*

Anzeigenverkaufsleitung: *»Populäre Computerzeitschriften«: Alexander Narings (780)*
Anzeigenleitung: *Alicia Clees (313) – verantwortlich für Anzeigen*

Anzeigenformate: *1/1 Seite ist 266 Millimeter hoch und 185 Millimeter breit (2 Spalten à 86 Millimeter oder 4 Spalten à 43 Millimeter). Vollformat 297x210 Millimeter.*

Anzeigenverwaltung und Disposition: *Lisa Landthaler (233)*

Anzeigen-Auslandsvertretung: England: *F. A. Smyth&Associates Limited, 23a, Aylmer Parade, London, N2 OPQ. Telefon:0044/1/3405058, Telefax: 0044/1/3419602*
Taiwan: *Third Wave Publishing Corp., 1-4 Fl. 977 Min Shen E. Road, Taipei 10581, Taiwan, R.O.C., Tel. 00886/2/7630052, Telefax: 00886/2/7658767, Telex: 078529335*

Vertriebsleiter: *Helmut Grünfeldt (189)*

Verkaufsleiter Abonnement: *Benno Gaab (740)*

Verkaufsleiter Einzelhandel: *Robert Riesinger (364)*

Vertrieb Handelsauflage: *Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Straße 96, 7000 Stuttgart 1,*

Bezugsmöglichkeiten: *Leser-Service: Telefon (089) 46 13-366. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.*

Preis: *Das Einzelheft kostet DM 16,–*

Druck: *SOV Graphische Betriebe, Laubanger 23, 8600 Bamberg*

Redaktionsdirektor: *Michael M. Pauly*

Vorstand: *Otmar Weber (Vors.), Bernd Balzer, Werner Brodt*

Leiter Unternehmensbereich »Populäre Computerzeitschriften«: *Eduard Heilmayr, Werner Pest*

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen: *Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 46 13-0, Telex 5-22052*

Telefon-Durchwahl im Verlag: *Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089/46 13 und dann die Nummer, die in den Klammern hinter dem jeweiligen Namen angegeben ist.*

Mit **ComicSetter** können Sie Ihre eigenen Cartoons schreiben, zeichnen und editieren – auch wenn Sie kein Zeichenprofi sind. Figuren und Hintergrundszenen werden fertig mitgeliefert, Sie müssen sie nur nach Ihren Wünschen zusammenstellen. Beim Entwerfen von Szenen stehen Ihnen eine einfach zu bedienende Benutzeroberfläche und eine Vielzahl von Mal- und Zeichenwerkzeugen zur Verfügung. Erfinden Sie die Helden Ihrer Geschichte. Plazieren Sie sie nach Belieben in den verschiedenen Szenen.

Und das alles natürlich in einer fast unbegrenzten Farbvielfalt.

Sie werden erstaunt sein, in welch kurzer Zeit Sie Ihre Comics zu Papier bringen können.

Bestell-Nr.: 54119
Preis: DM 198,-* (sFr 178,-*/öS 1980,-*)

Deutsche Version in Vorbereitung.
Update DM 49,-* (sFr 49,-*/öS 490,-*)
(lieferbar 1. Quartal 1989)

Zusatzdisketten zu ComicSetter mit einer Vielzahl von Figuren und Szenen aus den Bereichen Superhelden, Science Fiction und Funny Figures:

ComicArt Super Heroes
Bestell-Nr.: 54123

ComicArt Science Fiction
Bestell-Nr.: 54124

ComicArt Funny Figures
Bestell-Nr.: 54125

Preis je Produkt: DM 69,-*
(sFr 62,-*/öS 690,-*)
*Unverbindliche Preisempfehlung

Markt&Technik-Produkte erhalten Sie in den Fachabteilungen der Warenhäuser, im Versandhandel, in Computer-Fachgeschäften oder bei Ihrem Buchhändler.

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Fragen Sie Ihren Fachhändler nach unserem kostenlosen Gesamtverzeichnis mit über 500 aktuellen Computerbüchern und Software. Oder fordern Sie es direkt beim Verlag an!

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

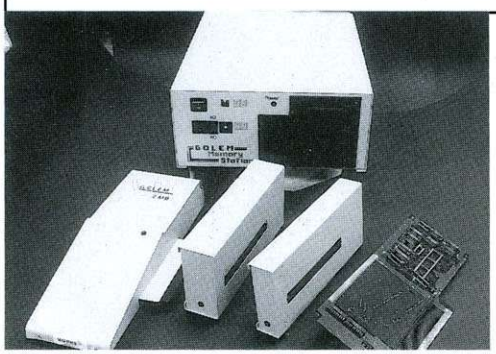
Bestellungen im Ausland bitte an: SCHWEIZ: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Telefon (042) 415656. ÖSTERREICH: Markt&Technik Verlag Gesellschaft m.b.H., Große Neugasse 28, A-1040 Wien, Telefon (0222) 587 1393-0; Rudolf Lechner&Sohn, Heizwerkstraße 10, A-1232 Wien, Telefon (0222) 67 75 26; Ueberreuter Media Verlagsges.mbH (Großhandel), Laudongasse 29, A-1082 Wien, Telefon (0222) 48 15 43-0.

225-1/811

Die Zeit war reif

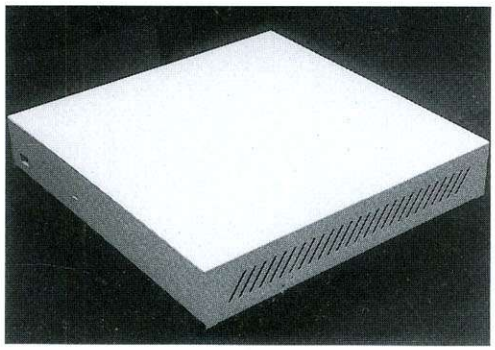
Golem

Amiga Festplatten



20 MB	1398,-
31 MB	1498,-
40 MB	1898,-
62 MB	2098,-

inkl. 2 MB RAM Controller
inkl. Elektronik für 2 Drives



20 MB	998,-
31 MB	1098,-
40 MB	1498,-
62 MB	1698,-

GOLEM MEMORY STATION

- 20 bis 61 Megabyte Festplatten
- Elektronik für 1 x 3,5 u. 1 x 5,25 Display Drive
- 2 MB dyn. RAM Controller als HD Interface
- anschlußfertig für alle Amiga
- preiswerte Nachrüstsätze für Floppies u. RAM
- Integrierung vorhandener Golem-Hardware
- Restelektronik wird vom Hersteller zurückgekauft
- Einbau einer 2. HD im 5,25 Slot problemlos
- stabiles Amiga-farbenes Metallgehäuse

Harddisk-Treiber der Spitzenklasse

- 100 % in Maschinensprache
- verwaltet 2 Festplatten bis 16 Köpfe, 2048 Zylinder
- prüft u. sperrt fehlerhafte Tracks
- unterstützt Fast-Filing-System
- Maus- u. Menü-gesteuerte Formatsoftware
- hardformatieren in ca. 2 Min., Softformat in 3 Sec.
- mit HD Interface oder HD-2 MB RAM Interface

GOLEM HD 3000

- 20 bis 62 Megabyte Festplatten
- eigenes Schaltnetzteil mit Lüfter
- anschlußfertig für alle Amiga
- ideal als Monitorunterbau od. A 1000 Überbau
- superflaches, formschönes Metallgehäuse
- Harddisk-Treiber wie Memory Station

Aufpreis für 2 MB RAM Interface 200,-

★★★★★ Wir verwenden ausschließlich NEC Qualitätsfestplatten ★★★★★

Nachrüstsätze zur Golem Memory Station

2 Megabyte RAM Bank	1099,-
3,5 Zoll NEC Drive	200,-
5,25 Zoll NEC Drive	220,-

Einführungsangebot
bis zum 28.02.1989

30 MB zum Preis von **20 MB**
60 MB zum Preis von **40 MB**

Bestellen Sie einfach per
0231/818325-27
Fax: 0231/817429

Kupke GmbH
Burgweg 52a
4600 Dortmund 1