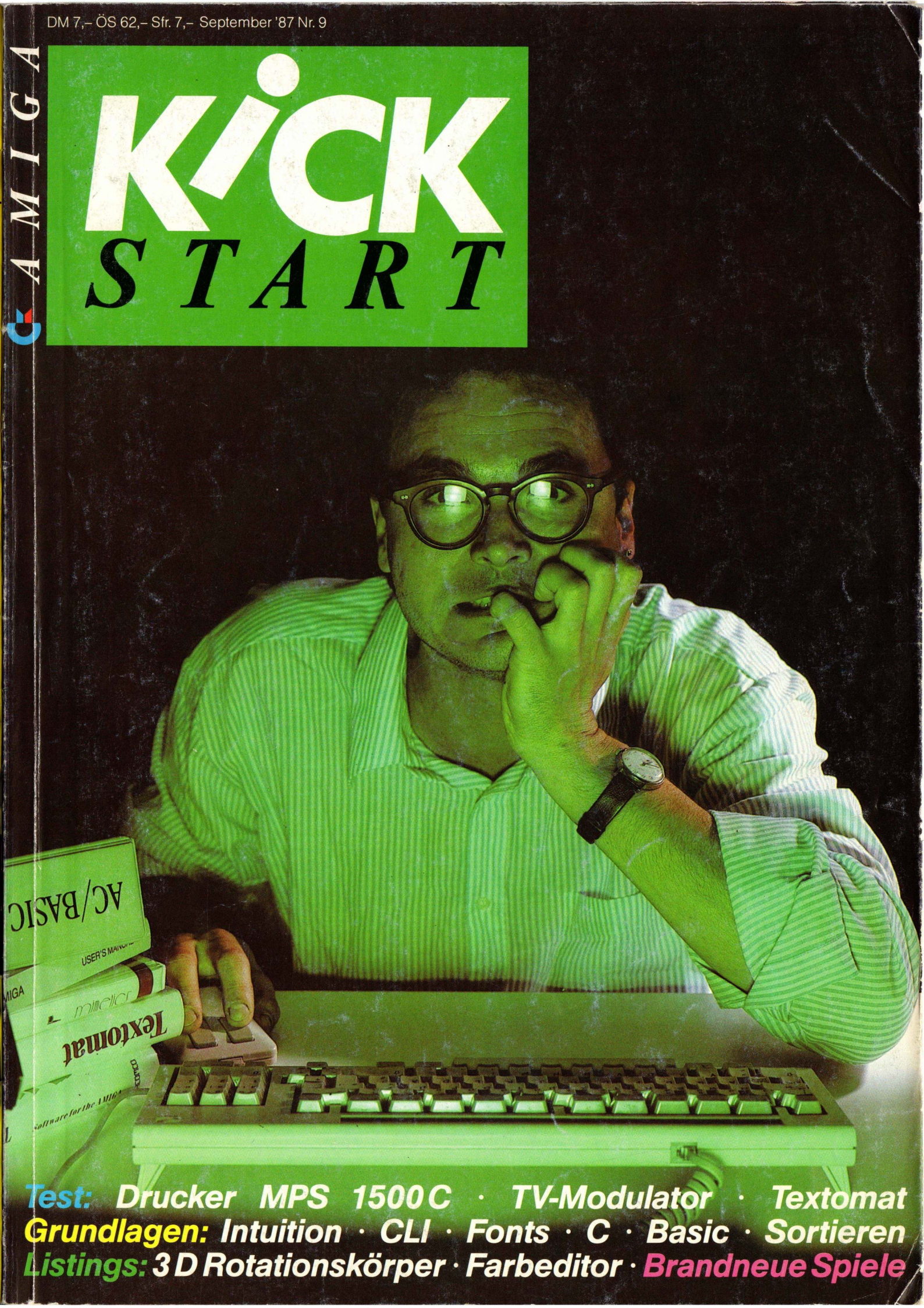
DM 7,– ÖS 62,– Sfr. 7,– September '87 Nr. 9

AMIGA

KICK START



Test: Drucker MPS 1500C · TV-Modulator · Textomat
Grundlagen: Intuition · CLI · Fonts · C · Basic · Sortieren
Listings: 3D Rotationskörper · Farbeditor · Brandneue Spiele

Amiga 2000
4 MB Steckkarte
2 MB bestückt
999,–

Kupke Computertechnik GmbH

6 Gründe dafür Golem

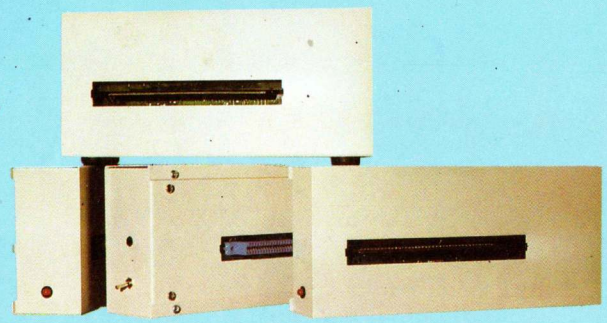
Drives & Ram Box

3,5 Zoll externe Amiga-Laufwerke
Amiga 500/1000/2000

auch als Einschubdrive für Amiga 2000 **331,–**

1. 100% kompatibeles NEC 1036 a
2. Amiga-farbenes Metallgehäuse
3. farblich passende Frontblende
4. durchgeführter Floppybus
5. Ein-/Aus-Schalter
6. Preis **369,–**

externe RAM-BOX Amiga 1000
auch für Amiga 500 (mit Adapter)



Golem 2 MB erweitert den „Amiga" auf 2,5 MB

1. Autokonfigurierend (ab Kick 1.2)
2. Amiga-farbenes Metallgehäuse
3. durchgeführter Systembus
4. Ein-/Aus-Schalter
5. Erweiterbar
6. Preis

2 MB 1198,–

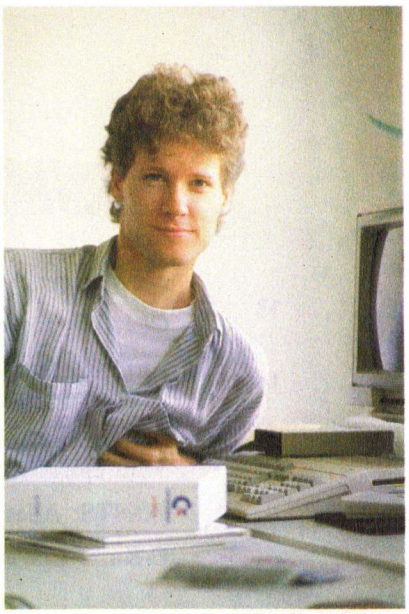
5,25 Zoll Amiga Laufwerk
40/80 Track Umschaltung, Ein-/Aus-Schalter
Busdurchführung vorbereitet, 880 KB Speicherkapazität **479,–**

500er Peripherieadapter
für 1000er Peripherie am 500er **59,90**

Kickstart V 1.2 Eprom Modul
ansteckbar am Systembus, abschaltbar, sodaß andere Kickstartversionen wieder gebootet werden können **199,90**

Wir liefern im 3-Tage-Rhythmus

Kupke Computertechnik GmbH
4600 Dortmund, Apelank 28
Tel.: 02 31/85 26 05



EDITORIAL

Während der heißen Tage, die es nach der langen Kälteperiode noch reichlich gab, lagen wir nicht untätig am Strand, sondern arbeiteten an dieser KICKSTART-Ausgabe. Natürlich durfte jeder Redakteur auch einige Tage ausspannen und dabei kreative und produktive Energie tanken, aber in erster Linie wurde die 'Sommerpause' dazu genutzt, neue Soft- und Hardware für den AMIGA zu testen.

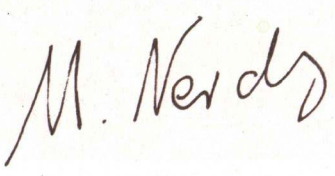
Die Leserumfrage ist inzwischen abgeschlossen. Wir möchten deshalb an dieser Stelle all jenen noch einmal recht herzlich danken, die teilgenommen haben, auch wenn die Preise nicht großartig waren. Aber schließlich sollen damit Ihre Vorstellungen von einem AMIGA-Fachmagazin verwirklicht werden. Ich hoffe deshalb, daß Sie uns auch weiterhin eifrig Ihre Kritik mitteilen und uns damit den Weg weisen.

Einige der Vorschläge wurden von uns bereits umgesetzt: So schreiben wir nun bei jedem Test, auf welcher AMIGA-Konfiguration das Programm lauffähig ist. Außerdem wird nun jeder Hard- und Softwaretest durch eine Übersicht ergänzt, in der stichpunktartig die wichtigsten Plus- und Minuspunkte aufgeführt werden.

Geändert hat sich auch der Public-Domain-Service. Fast alle Programme können nun ganz einfach von der Workbench aus gestartet werden, die Disketten sind direkt bootfähig. Also Diskette ins Laufwerk, Reset, und der Rest geht von selbst!

Wir hoffen, daß Sie mit diesen Verbesserungen und unserem jüngsten Heft zufrieden sind und freuen uns weiterhin auf Ihre Meinung.

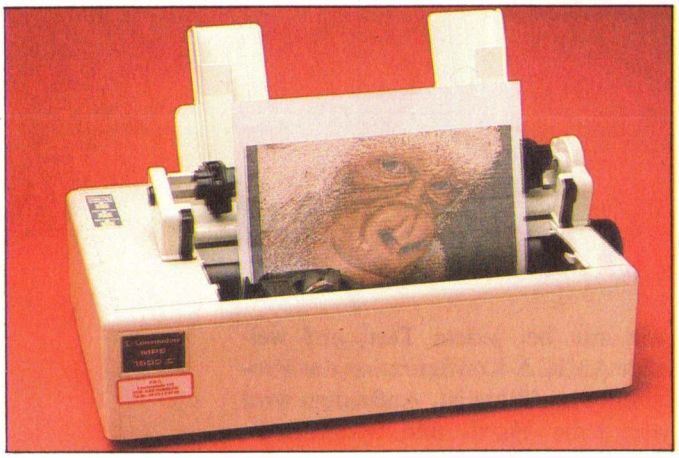
Markus Nerding



P.S.: Ab sofort gibt es eine „Monatsdiskette", auf der sich jeweils die Listings von zwei Ausgaben der KICKSTART befinden. Dann muß nicht jeder den entsprechenden Compiler haben, um Programme, die in C, Pascal oder Assembler geschrieben sind, ablaufen zu lassen.

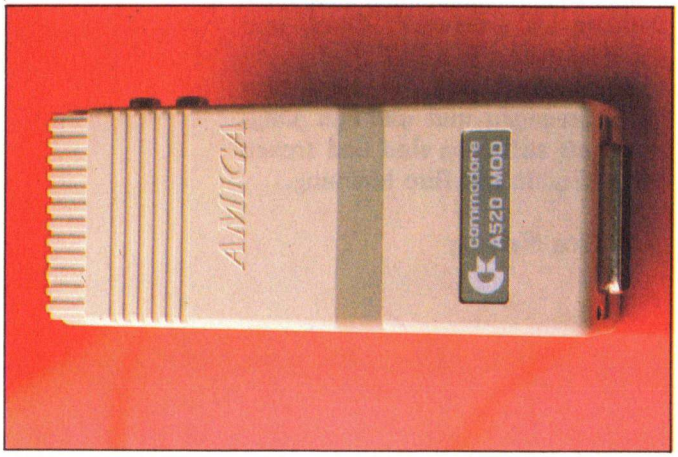
Inhalt:

HARDWARE



Neues von COMMODORE

COMMODORE hat sich entschlossen einen neuen Farbdrucker in das Sortiment zu übernehmen. MPS 1500 C ist sein Name und der Preis ist eine kleine Sensation!



Auch der TV-Modulator ist ganz neu, doch wie gut ist die Bildqualität und für wen lohnt er sich?

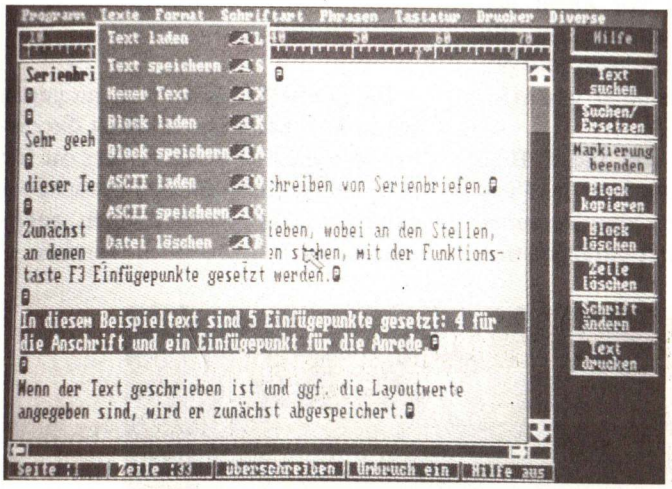
GRUNDLAGEN

SPIELE

Spiele

Gute Grafik und Animation versprechen die neuen Actionspiele GARRISON und BARBARIAN. Durch sehr gute und stimmungsvolle Grafiken kann auch das Adventure THE GUILD OF THIEVES überzeugen, das damit die Nachfolge von THE PAWN antritt. Viel Geschicklichkeit verlangt dagegen das Spiel BAD CAT von Rainbow Arts.

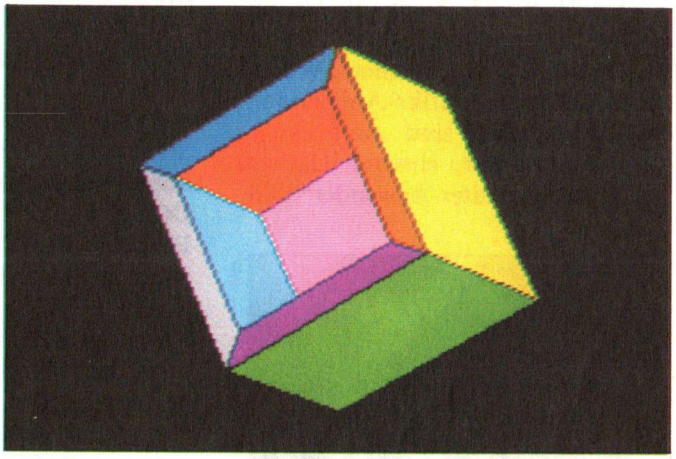
SOFTWARE



Textverarbeitung

Neben der neuesten Version von UBM-Text wird auch eine weitere deutsche Textverarbeitung vorgestellt, der neue TEXTOMAT. Bringen diese Programme endlich die erhoffte Qualität, die dem AMIGA auf diesem Sektor bislang fehlt?

LISTING



3 D Realtime Rotation

Ein sehr interessantes Programm zum Erstellen und Darstellen von dreidimensionalen Körpern. Diese können dann über die Tastatur gedreht und gezoomt werden. Das Programm ist in C geschrieben und wird im nächsten Heft noch um einige Funktionen ausgebaut.

RUBRIKEN:

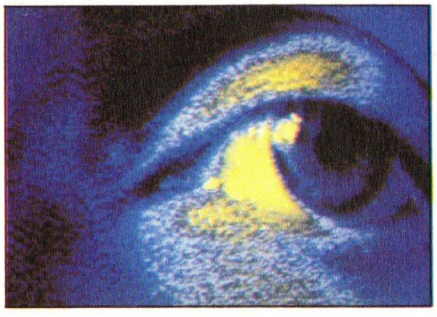
NEWS

WORKSHOP für Computergrafik und Animation

Vom 21. bis 23. August findet, im Anschluß an ein internationales Austauschseminar, ein offener „Workshop Computergrafik/-Animation“ statt. Eingeladen sind „verborgene Computerkünstler, einsame Hacker und Homecomputer-Anwender“ und

alle, die sich für dieses Anwendungsgebiet interessieren. Gearbeitet wird vorwiegend mit dem AMIGA, aber auch Atari ST, MacIntosh und IBM sind vertreten. Mit Geräten wie Digitizern, Genlok-Interfaces, Farbdruckern, Videorecordern und Kameras geht es vor allem um den Gedankenaustausch und die kreative Umsetzung der Ideen. Die dabei entstehenden Produktionen werden am Sonntag (23.8.) ab 18 Uhr bei einer öffentlichen Veranstaltung vorgestellt. Interessierte wenden sich bitte an:

Verein für visuelle Kommunikation e.V. Schwarzer Bär 6, 3000 Hannover 91, 05 11 / 44 14 40; Heiko Idensen 05 11 / 70 95 59.



TRUE BASIC für AMIGA, IBM...

TRUE BASIC ist weiter im Vormarsch. Nachdem es bereits Versionen für MacIntosh, IBM und AMIGA gibt, ist nun auch eine spezielle Atari-ST-Version fertig. Dieser von den BASIC-„Urvätern“ entwickelte Dialekt enthält viele Elemente späterer Sprachschöpfungen wie Pascal, C oder Modula. Neben Strukturen wie DO WHILE, DO UNTIL, IF/ELSE/ENDIF, SELECT CASE und Unterprogrammaufrufen, die auch Rekursionen erlauben, ist vor allem der Aufruf von Libraries interessant, die bereits als Module existieren und die auch in C oder Assembler geschrieben sein können.

Der große Vorteil von TRUE BASIC ist die Übertragbarkeit der Programme zwischen den verschiedenen Rechnern. Dies betrifft natürlich nur den normalen Sprachumfang, denn jede Version hat eine Library, die den Zugriff auf spezielle Systemroutinen erlaubt.

Zu TRUE BASIC gibt es außerdem ein RUN-TIME-PAKET, mit dem die Programme so zusammengebunden werden, daß sie ohne den Interpreter laufen. Jeder Besitzer dieses Pakets kann diese Programme dann frei weitergeben oder weiterverkaufen.

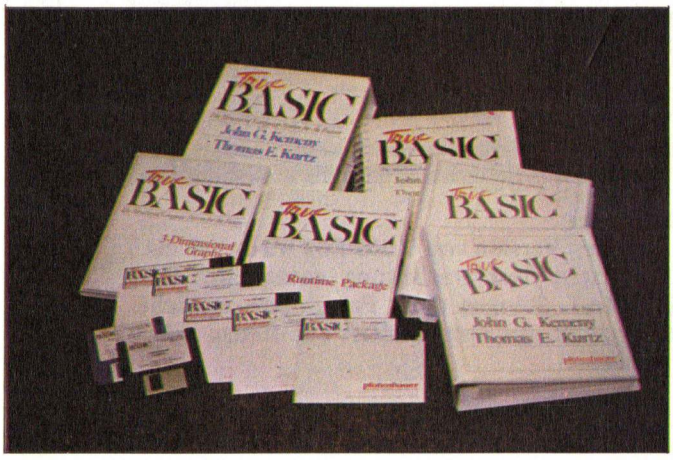
Mittlerweile ist auch eine Anzahl von Erweiterungen in Form von Libraries erhältlich: 3-D Grafik, Erweiterung der Zeichenketten-Library, Suchen und Sortieren, Developers Kit. Für den IBM sind sogar noch weitere erhältlich.

Der Distributor von allen TRUE BASIC-Produkten für Deutschland ist die Firma Pfotenhauer. Dort ist auch die deutsche Version von TRUE BASIC für den IBM, mit zwei deutschen Handbüchern, erhältlich.

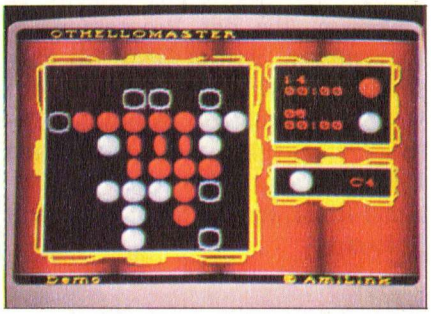
Pfotenhauer Microcomputer-Anwendungen
Neulandstraße 16
7590 Achern
Tel.: 0 78 41 / 50 56

Othello Master

Bei „Othello Master“ ist hingegen schon alles klar. Das Spiel ist die bisher spielstärkste Reversi-Version für den Amiga. Graphik und Sprachausgabe gibt es in einer deutschen, englischen und französischen Version – alles in einem Programm. Besonders schön animiert ist das Umdrehen der Steine, wenn sie von der Gegenseite „gekapert“ werden. Wer nicht gegen einen menschlichen Gegner antreten will, findet im Copmuter einen hartnäckigen Kontrahenten. Spielstufe vier ist kaum zu schlagen. Bei Reversi-Programmen bisher nicht vorhanden waren Optionen wie Anzeigen aller möglichen Züge, Position aufbauen etc. Othello Master ist erhältlich bei AmiLink, 7 Rue Cherbu-

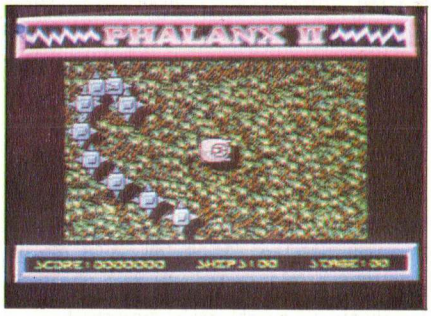


liez, 1207 Geneva, Schweiz, Tel. 00 41 - 22 - 86 54 42.

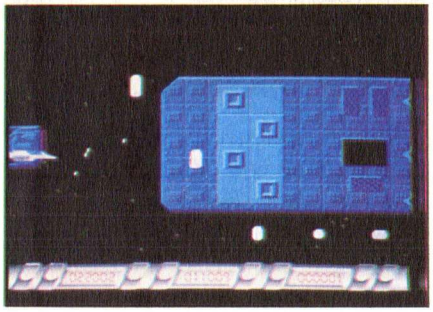


Uridium – oder was?

Digital Dreams wird noch im August die Fortsetzung von „Phallanx" präsentieren: „Phallanx II – The Return" erinnert ein wenig an „Uridium" für den C 64. Ein dreidimensional scrollendes Raumschiff verteidigt die Erde gegen ganze Wellen von Angreifern. Hintergrundgraphiken des horizontal durchscrollenden Spiels sind neben Raumschiffen auch Dschungel- und Kraterlandschaften. Besonders die Graphik wurde gegenüber dem ersten Spiel stark verbessert.



Dem gleichen Thema widmet sich auch das Spiel „Neutralizer". Es ist das erste Werk einer Gruppe junger Programmierer aus Köln. Die Raumschlacht tobt hier vor dem Hintergrund feindlicher Superraumschiffe. Obwohl auch dieses Spiel sehr stark an „Uridium" erinnert, bietet es gegenüber seinem „Vorbild" einige wesentliche Veränderungen. So fliegen im Weltraum diverse Maschinenteile



herum. Wer sie aufsammelt, verfügt über verschiedene, den jeweiligen Gegnern angepaßte Geschütze. Es gibt insgesammt acht Levels, der Sound ist gut, die Graphik sogar ausgezeichnet. Der Vertrieb von beiden Spielen war bei Redaktionsschluß noch nicht geklärt.

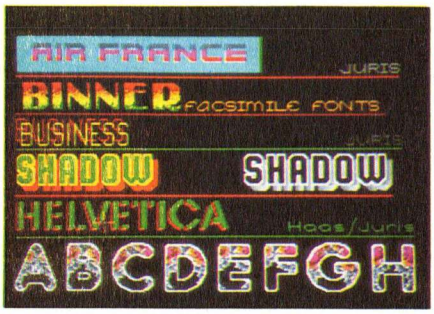
Der jüngste Werbegag von Commodore

Einen neuen Weg der Werbung haben sich die beiden Großkonzerne Ariola und Commodore ausgedacht: Es ist die Pop-Gruppe '16 Bit', deren Platten bei Ariola erscheinen und die ihre Musik mit reichlicher Computerunterstützung produziert. Die neueste Maxi-Single (Titel: „Changing Minds") der erfolgreichen Gruppe zeigt auf der inneren Plattenhülle ein Bild des AMIGA 500. Der Slogan „Mit diesen 16 Bit kann jeder komponieren" soll auf die besonderen musikalischen Fähigkeiten des Rechners aufmerksam machen. Nebenbei werden noch die 'Solisten' vom AMIGA vorgestellt, 'Fat Agnus', 'Paula' und 'Denise' dürfen natürlich auch nicht fehlen. Auf diese Maxi-Single soll ein umfangreiches Programm an Promotion-Aktivitäten aufbauen.

(AK)

50 neue Schrifttypen

Klaus Juris, Produzent der neuen Schrifttypen, hat eine Misere im Bereich der für den Amiga verfügbaren Schriftarten erkannt. Deswegen entwarf er mit viel Mühe 50 unterschiedliche Schriften, die seiner und unserer Meinung nach selbst Profi-Ansprüchen gerecht werden. Die Headline-Gestaltung per Brush im DeluxePaint II-Format ist sicher nicht der komfortabelste Weg, bietet aber durch die enorm große Auswahl eine Entschädigung. Eine kleine Kostprobe verschiedener Schrifttypen bietet das nebenstehende Bild. Beziehen kann man die Diskette mit den 50 Schriften zum recht hohen Preis von 89 DM (incl. Versand bei Vorkasse); die



Nachnahmelieferung kostet 4,50 DM mehr.

Klaus Juris
Grafik-Design
Bahnhofstr. 106
6392 Neu-Ansbach

(AK)

Neue Spiele von Rainbow Arts

Kurz nach Redaktionsschluß erreichten uns noch einige „heiße News" von Rainbow Arts. Wir mußten ein bißchen zaubern, deshalb bitten wir um Verständnis, daß die Information darüber recht knapp ausfällt.

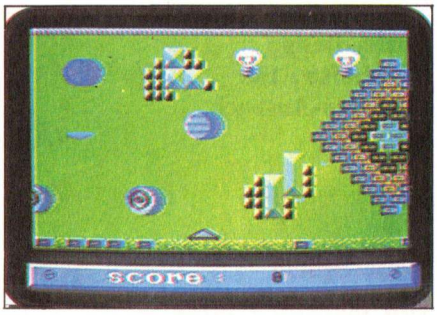
In achtzig Tagen um die Welt:

Die Faszination einer Weltreise, der Reiz fremder Kulturen und die Erforschung unheimlicher Plätze hatte schon Jules Verne zu dem gleichnamigen Weltbestseller inspiriert. In einem edlen englischen Club schließt sein Held Phileas Fogg im Jahre 1872 die Wette ab, die Welt in 80 Tagen zu umrunden. Diese Wette, von den Clubmitgliedern belächelt, entwickelt sich zu einem Abenteuer auf Leben und Tod. Der Spieler übernimmt die Rolle von Passpartout, dem treuen Diener. Um seinen Herrn vor Problemen zu bewahren, befreit er gefangene Jungfrauen aus heidnischen Tempeln, tanzt mit Indianern und anderes mehr. Die knappe Kasse wird durch Kartenspielen aufgebessert.

Jinks:

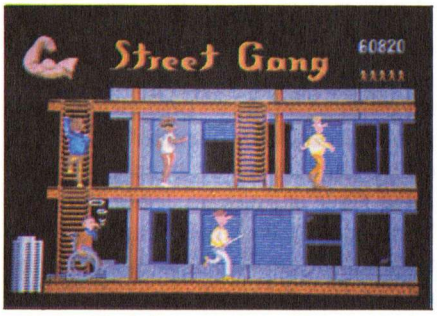
Von der intergalaktischen Raumkomission wurde der Planet Atavi

bisher als technisch sehr rückständig eingestuft. Diese Einstufung soll einer Überprüfung unterzogen werden. Der Spieler soll mit seinem Raumgleiter eine kleine Aufklärungssonde über den Planeten steuern, um technische Neuentwicklungen zu studieren. Dabei stellen sich die Bewohner als klug und gerissen heraus. Sie stellen der Sonde und auch dem Mutterschiff so manches Hindernis in den Weg.



Street Gang:

Mickys Leben war bisher ebenso normal wie langweilig. Mit seinen Eltern wohnte er in einer Kleinstadt auf dem Land. Doch dann kam der Tag, an dem sich alles ändern sollte: Seine Eltern zogen mit ihm nach New York. Dort werden die Straßen von den verschiedensten Banden regiert. Um Mitglied in der Bande seiner Straße zu



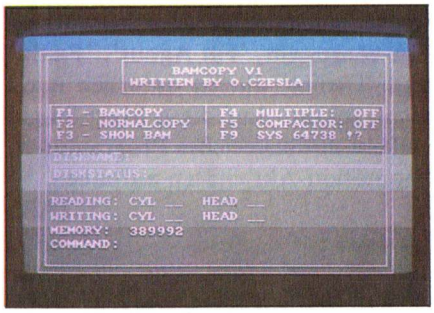
werden, macht sich Micky auf den Weg, dem berüchtigten Locke sein heißgeliebtes Haarbüschel zu rauben. Mickys Abenteuer ist ein Geschicklichkeitsspiel, die Anleitung dazu besteht aus einem kurzen Comic.

Ebenfalls eingetroffen sind „Garrison" und „Bad Cat" – die Besprechungen dazu finden Sie im Spieleteil. Folgende Titel sind von Rainbow Arts für die Monate August bis Oktober angekündigt: Fruity Freak (ähnlich MR. Do), Rocket Attack, Space fight, Shooting Star, Pack Boy und Mind Breaker (eine Fortsetzung von Break).

cpl

Bam-Copy „Kickstart"

Ein neues Kopierprogramm, das BAM-selected kopieren kann, hat Oliver Czesla (Digital Dreams) entwickelt. Die Daten werden beim Lesen kompaktiert bzw. beim Schreiben dekompaktiert. Ein möglichst schlichter Aufbau spart weiteren Speicherplatz. Mit diesem BAM-COPY will Digital Dreams vor allem den Amiga-Besitzern ohne Speichererweiterung und Zusatzfloppy das Leben erleichtern, da die meisten Disketten nicht ganz voll sind und dann gleich in einem Rutsch kopiert werden können. BAM-COPY besitzt u. a. eine Verify-Option. Es ist nicht für das Duplizieren kopiergeschützter Software gedacht. Das „Kickstart-Copy" wird in unserer nächsten Ausgabe abgedruckt und ist dann auch auf der Monatsdiskette erhältlich.



Marauder III

Im Gegesatz zu BAN-COPY dienen die Kopierprogramme der MARAUDER-Serie zum Anlegen von Sicherheitskopien geschützter Originale. Da dies im gesetzlich erlaubten Rahmen beim Amiga durchaus sinnvoll ist (Zerstören von Disketten beim Zurückschreiben der HI-Scores etc.), waren bereits MARAUDER I und II sehr weit verbreitet. MARAUDER III erscheint im September und hat u. a. die Option für 81 Spuren. Vertrieb: Discovery Software.

HP Calculator

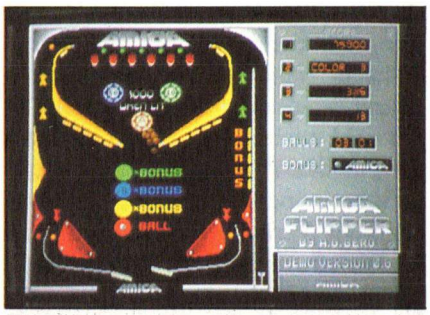
Den Hewlett – Packard Calculator DX 16 setzt zur Zeit Kirk Swenson für Discovery Software auf den Amiga um. Mit seinen zahlreichen Optionen, die das Programmieren unterstützen, ist das Gerät für Professionals und solche, die es werden wollen, ein wichtiges Hilfsmittel. Besonders wichtig sind die diversen Umrechnungsmöglichkeiten wie Dezimal-Binär etc. Vertrieb: Discovery Software.

Micro Illusions setzt nach

Nachdem „The Faery Tale Adventure" zum Hit wurde, schiebt Micro Illusions noch im Herbst vier weitere Spiele nach. Eines davon wird wieder ein Rollenspiel in der Art von „Faery Tale" sein.

Amiga Flipper

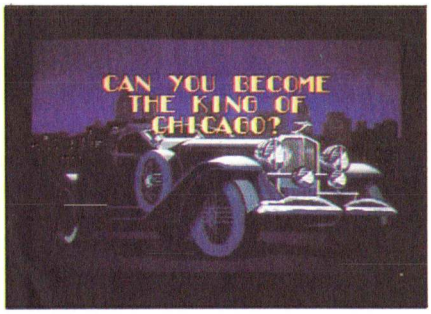
„Slamball" heißt das erste Flipperprogramm für den Amiga, geschrieben von H. G. Berg. Bis zu vier Spieler können im Competition-Mode gegeneinander antreten. Drei verschiedene Flipper sollen zur Auswahl stehen. Graphik und Sound waren schon in der Demo-Version sehr gut. Die Version 0.9 ist übrigens als Public-Domain-Diskette frei erhältlich. Bei ihr kann man bereits mit den ALT-Tasten spielen. Vertrieb und Preis des Programms waren bei Redaktionsschluß noch nicht bekannt.



King of Chicago

Master Designer hat Ende Juli eine Demoversion von „King of Chicago" herausgegeben. Sie beinhaltet einen Teil des Sounds und der animierten Graphiken des Spiels. Die Demover-

sion ist Public Domain.



Zeitschriften auf Diskette

Das erste von zwei Magazinen auf Diskette stammt von der guten alten Insel. 'JUMPDISK' erscheint monatlich. Mittlerweile ist die Magazin-Diskette auch direkt in Deutschland zu beziehen. Die Juni-Ausgabe, die uns vorlag, konnte über die Workbench durch das Anwählen eines Icons gestartet werden. Nach doppeltem Anklicken erschien ein Menü, in dem der Inhalt des Magazins aufgelistet war: Unter anderem gab es eine Reihe einfacher Spielchen, Bilder, Animations-Demos sowie einige nützliche Utilities. Die Herausgeber haben sich große Mühe gegeben, so ist etwa der Inhalt des Magazins von einem eigenständigen Programm aus zu erreichen.

'JUMPDISK' kann man unter folgender Adresse beziehen:

PDC Klein
Louisenstr. 115
6280 Bad Homburg

RAINBOW ARTS-WETTBEWERB

Zusammen mit der Kickstart verlost Rainbow Arts in der nächsten Ausgabe dieser Zeitschrift 20 Amiga-Originale des neuen Spiels „BAD CAT". Als Hauptpreis gibt es sogar eine Einladung zu einem einwöchigen Besuch bei Rainbow Arts. Der Gewinner kann sich Tips zum Programmieren holen, darf bei der Entwicklung der neuen Programme ein Auge riskieren und bekommt einen kompletten Satz der Rainbow-Arts-Titel mit nach Hause – „wenn er brav ist", so Marc Ullrich, der RBA-Geschäftsführer. Kickstart verlost Jahresabos und Spiele.

Freuen können sich alle Amiga- und ST-Besitzer, die im Bereich des Senders Mainradio (Würzburg und ca. 60 km Umgebung) wohnen. Bereits im Juni und Juli wurden beim Mainradio in Zusammenarbeit mit unserer Redaktion mehrere Jahresabos von Kickstart und ST-Computer verlost. Im August und September gibt es wieder einige Abos und dazu noch Programme von Psygnosis (Barbarian), Rainbow Arts (Bad Cat), Application Systems u.a.m. zu gewinnen. Mainradio findet man auf UKW, Frequenz 103. Die Verlosung findet in der Morgenshow (5.30 – 7.30 Uhr) statt.

Ein weiteres Disketten-Magazin kommt aus deutschen Landen: 'AmigaJUICE'. Anders als bei dem englischen Magazin kann man die Diskette direkt oder nach Laden der KICKSTART-Software einlegen, je nachdem, welchen Rechner man besitzt. Am Anfang sieht der Leser eine Grafik sowie einige redaktionelle Mitteilungen. Danach erscheint ein Icon auf dem Workbench-Bildschirm, das nach zweimaligem Anwählen ein Fenster mit dem Inhalt des Magazins öffnet. Die unterschiedlichsten Beiträge sind in acht verschiedenen Ordnern verstaut: Sie reichen von Aktuellem, Redaktion, Programmen, Software-Tests, Hardware über Unterhaltung bis hin zu Tips&Tricks. Einige Unter-Ordner waren noch nicht gefüllt. Der Herausgeber hofft jedoch, wenn sich sein Mitarbeiterstab erweitert hat (das Magazin soll von seinen Lesern selbst gestaltet werden) und mehr Artikel bei ihm eintreffen, diese Lücken zu füllen. Die Zeitschrift auf Datenträger machte einen durchaus soliden und akzeptablen Eindruck und kann unter Einsendung einer Leerdiskette mit beiligendem Rückporto bei nachstehender Adresse bezogen werden:

Amiga JUICE-Redaktion
Tilman Wittenhorst
Danquardstr. 12
3015 Wennigsen

Infocom-Adventure-Fans gesucht!

Die 'Infocom User Group' hat sich speziell den Aqventures von Infocom verschrieben. Dieses Softwarehaus ist bekannt für seine vorzüglichen und meist sehr komplexen Text-Adventures. Sie zeichnen sich besonders durch gute Handlungen und Texte aus. Die Grupe berücksichtigt jedoch auch andere Abenteuer-Spiele. User, die an einem bestimmten Punkt im Adventure nicht weiterkommen oder die einfach nur ein paar Tips zu diesem oder jenem Spiel benötigen, können sich mit der 'Infocom User Group' in Verbindung setzen. Obwohl sich die Vereinigung nicht direkt als Club bezeichnet, sondern vielmehr nur eine Interessengemeinschaft bildet, gibt es eine Zeitung namens 'Frobozz Time'.

Wer will, kann sich mit folgender Adresse in Verbindung setzen:
Infocom User Group
Weser Stadion
2800 Bremen

MESSE

The „Consumer Electronics Show“ Chicago 1987

Die CES ist nichts weniger als die größte Elektronik-Fachmesse der Welt: Treffender als mit diesem Superlativ kann man sie nicht beschreiben. Zweimal im Jahr tummelt sich ein ausgesuchtes Publikum (nur Elektronikfachhändler und Journalisten sind zugelassen) auf einer Ausstellungsfläche von mehr als 30 000 Quadratmetern bei mehr als 1400 vertretenen Firmen. Der Ausstellungsort wechselt im Halbjahresrhythmus: Die Sommer-CES findet stets in Chicago, die Winter-CES in Las Vegas statt.

Die großen Messen in Deutschland (CEBIT), England (PCW) und USA (CES) unterscheiden sich in ihrer Struktur wesentlich: Die PCW ist eine öffentliche Verkaufmesse. Auch die CEBIT ist öffentlich, doch keines der dort ausgestellten Produkte kann sofort gekauft und mitgenommen werden. Die CES geht noch einen Schritt weiter: Wie erwähnt, ist sie nicht für jedermann zugänglich, und auch direkt mitnehmen kann man nichts. Es bleibt also nur die Bestellung, und auch die nur bei größeren Stückzahlen. Dennoch zählt die CES regelmäßig über 100 000 Besucher. Dies liegt daran, daß sie nicht auf Computer beschränkt ist. So gab es in der Zeit zwischen 30. Mai und 2. Juni etwa vierzig Workshops zu unterschiedlichen Themen, von „New Dimensions in Retail Sales Training“ bis hin zu „The Management of Family-Owned Business“. Im Bereich Photographie wurde eine Kamera vorgestellt, die Bilder auf Diskette abspeichert. Sie schafft fünf Bilder pro Sekunde, wiegt rund ein Kilo und soll etwa tausend US-Dollar kosten. Erhältlich für Händler – ab Herbst. Für Musikfreaks gibt es jetzt den Plattenspieler mit visco-elastischem Material, das unerwünschte mechanische Vibrationen auffängt, Videofans können ab Herbst auf das CD-Video zurückgreifen, und wer von einem reinen Spielcomputer in Spielhallenqualität träumt, findet die entsprechende Konsole bei Nintendo.

Auch für die Amiga-Besitzer wurde natürlich einiges geboten. Allerdings war Commodore selbst diesmal leider nicht vertreten. Als Grund dafür vermuteten die Kollegen von AmigaWorld USA den abrupten Wechsel im Management der Firma, der, entgegen offiziellen Verlautbarungen, „wohl nicht ganz freiwillig vonstatten gegangen sei.“ Deshalb wurde auch die für September angekündigte AmigaWorld-Messe in San Francisco wieder abgesagt.

An erster Stelle standen bei den Amiga-Neuheiten erwartungsgemäß Spiele. Zwar soll es in absehbarer Zukunft auch gute Anwender-Software geben, doch auf der Messe selbst war sie kaum zu finden. Ebenso dünn waren neue Drucker oder andere Hardware gesät. Einzig Epson präsentierte mit dem LX 800 einen 9-Nadel-Drukker von guter Qualtät und zu einem günstigen Endpreis von 269 US-Dollar, der ab August erhältlich sein soll.

Vorab die technischen Daten:
Schrift Elite 180 Zeichen pro Sec.
Schrift Pica 150 ZpS
NLQ Elite 30 ZpS
NLQ Pica 25 ZpS

Character Set:
96 ASCII Characters
96 Italic Characters
32 international Charakters
32 Italic int. Characters
32 Epson Graphics Characters
96 IBM Graphics Characters

Der LX-800 verfügt über Einzel- und Endlospapiereinzug, hat 3 K Buffer und kann über Centronics betrieben werden. Wir werden das Gerät in einer unserer nächsten Ausgaben testen.

Software, meist Spiele, gab es reichlich. Unsere Aufzählung der interessantesten Neuvorstellungen wurde deshalb zur besseren Übersicht alphabetisch angeordnet:

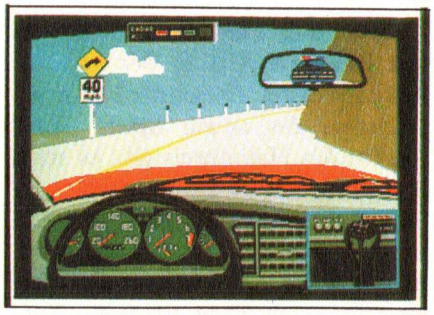
ACCOLADE:

Hardball:

Die Umsetzung des bereits vom C 64 bekannten Baseballspiels.

Testdrive:

Mit dem Ferrari oder Lamborghini geht die Testfahrt quer durch die USA – vorbei an Trucks, herabgestürzten Felsen und Baustellen immer auf der Hut vor den Polizeistreifen.



The Graphics Studio:

Mehr als ein Malprogramm oder ein Graphik-Editor. Graphiken können mit Text kombiniert und animiert werden.

ELECTRONIC ARTS:

Grand Prix:

Ein Autorennen auf den bekanntesten Rennstrecken der Welt. Mit Boxenstop und vielen Extras. Die in Chicago vorgestellte Version war komplett mausgesteuert.

„Archon III“:

Das neue Spiel von Freefall, dessen Name noch nicht feststeht. Arbeitstitel des Labyrinthspiels: „Auf dem Weg des Regenbogens“.

Von den beiden Spielen hoffen wir, in unserem nächsten Heft mehr berichten zu können. ECA hat uns auch Bilder zugesagt.

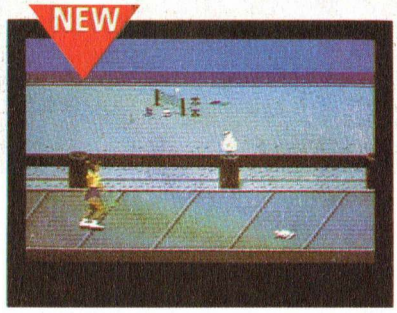
EPYX:

Street Sports Basketball:

Spielt dort, wo in Amerika die Kaderschmiede für die Nachwuchsstars steht – mitten auf der Straße. Im Hinterhof zwischen Mülltonnen und Garagen werden die Meisterschaften entschieden.

California Games:

Dies ist die Fortsetzung der World-Games-Serie. Es beinhaltet Disziplinen wie Surfen und Skateboardfahren.

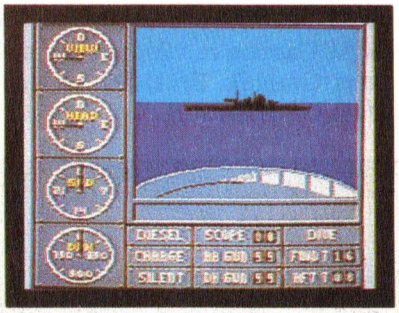


Sub Battle Simulator:

Ein U-Boot-Abenteuer im II. Weltkrieg. Destroyer – oder Jagd auf U-Boote mit dem Zerstörer.

Omnicron Conspiracy:

Captain Ace Powers wird mit der Aufgabe betraut, das mysteriöse Verschwinden eines Raumschiffes während einer Routinemission aufzuklären. Ein Action-Adventure.



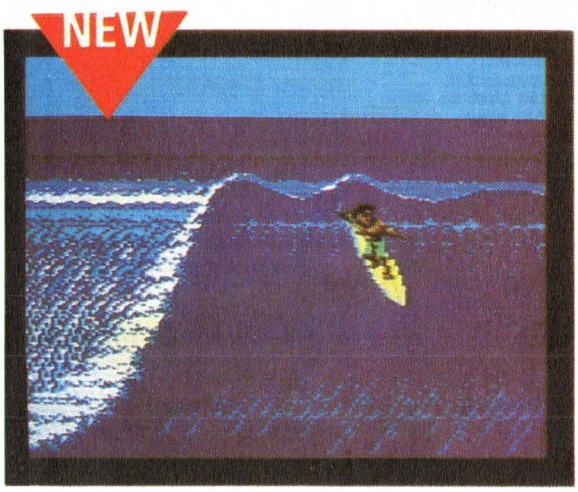
INFINITY SOFTWARE:

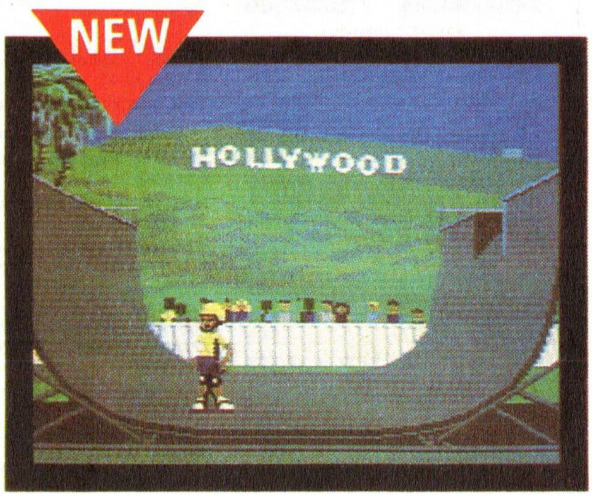
Gallileo:

Ein Astronomie-Programm mit den Daten von über 1600 Sternen. Es zeigt z. B. den Himmel von jedem Punkt der Erde. Der Programmierer Mike Smithwick arbeitet für die NASA. Preis: 99 US-Dollar.

Shakespeare:

Ein Desktop-Publishing-Programm mit iff-Graphik-Standard. Preis: 299 Dollar.





JAGUARE:

Alien Fires:

Ein Action-Strategie-Spiel zweier junger kanadischer Programmierer. Im nächsten Heft stellen wir die deutsche Version davon vor. Jeff Simpson hat uns zehn Exemplare für die Kickstart-Leser zugesagt.

MASTER DESIGNER:

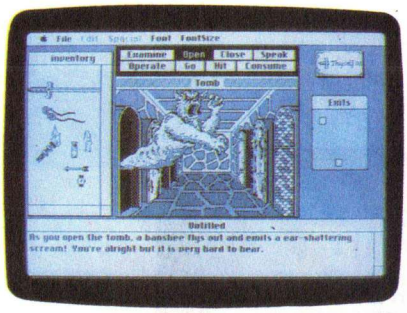
King of Chicago:

Das Neueste aus der Hitschmiede des Cinemaware-Teams. Ein Spiel um die Nachfolge der Gangsterkönigs Al Capone in Chicago mit animierter Graphik. Erhältlich ab Ende August bei Rushware.

MINDSCAPE:

Shadowgate:

Das Dritte aus der Graphik-Adventure-Serie nach „Deja Vu" und „Uninvited". Ziel ist es, in das dunkle Schloß Shadowgate einzudringen und den finsteren LORD daran zu hin-



dern, die Welt zu zerstören. Preis: 49,95 US-Dollar.

Into the Eagles Nest:

Ein vom C64 bekanntes Spiel, das im zweiten Weltkrieg spielt. Der Held muß aus einer Basis der Nazis drei Gefangene befreien. Die Umsetzung für den Amiga wird aber erst Ende des Jahres im Handel sein.

Computer Laptop Chess:

Ein neues starkes Schachprogramm unter MS DOS.

PSYGNOSIS:

Barbarian:

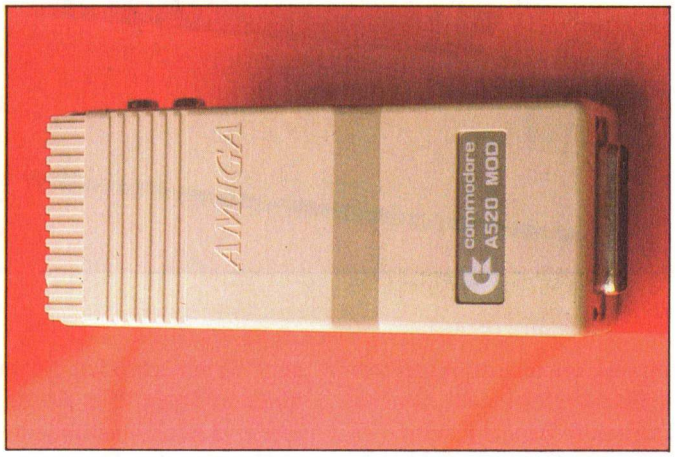
Ein Höhlenspiel aus grauer Vorzeit. Alles weitere im Spieleteil.

Am Rande der Messe, sozusagen inoffiziell, gab es dann noch die eine oder andere Information von Firmen oder Programmierern zu erhaschen. So plant MicroProse eine Umsetzung verschiedener Simulatoren, Amilink wird ein Druckerprogramm namens Amiprint herausbringen und Rick Ross plant mit Discovery Software den Sprung nach Deutschland, zukünftiger Partner: CHS. Discovery hat übrigens für den Herbst 87 drei neue Produkte in Aussicht gestellt: Einen HP Calculator, eine Arkanoid-Version für den Amiga und Marauder III.

(cpl)

AMIGA 500 ohne Monitor: Der TV-MODULATOR 520

Mit dem TV-Modulator 520 von Commodore ist es möglich, Bild und Ton des AMIGA 500 auf einem Fernsehgerät oder einem Composit-Video-Monitor auszugeben.



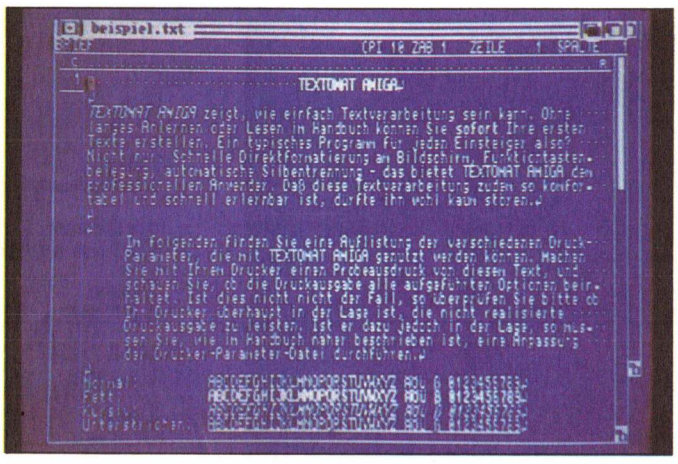
Wer mit dem Gedanken spielt, sich einen AMIGA 500 zu kaufen, der wird scharf mit dem Preis des Farbmonitors 1081 (ca. 798 DM) kalkulieren müssen: Sicher ein großer Brocken in der Geldbörse eines jungen Menschen. Doch es gibt eine Alternative, die für einige Anwendungen auch akzeptabel erscheint: Der TV-Modulator 520, der direkt von Commodore angeboten wird. Ganze 59 Mark kostet das längliche Kästchen, das direkt auf den RGB-Video-Ausgang gesteckt wird, an den sonst der Monitor angeschlossen ist.

Ein mitgelieferter Y-Adapter verbindet die beiden Tonausgänge des AMIGA mit dem Modulator. Ein weiteres Kabel stellt schließlich die Verbindung zum Antenneneingang des Fernsehers her. Nun muß nur noch der Kanal 36 justiert werden, und schon kann das Bild des AMIGA bewundert werden. Zusätzlich ist auch ein separater Ausgang für einen Composite-Video-Monitor vorhanden.

Zu dem günstigen Preis darf man sicher nicht allzuviel erwarten, aber je nach dem verwendeten Fernseher gibt es ein durchaus aktzeptables Bild zu sehen. Allerdings zeigen sich öfters störende Streifen auf dem Bildschirm, die mit einem leichten Flimmern den unruhigen Charakter des Bildes verursachen. Dies fällt natürlich bei Spielprogrammen weniger auf als bei Anwenderprogrammen wie z. B. Textverarbeitungen oder Datenbanken.

Wer auf seinem AMIGA in erster Linie Spiele laufen lassen will, dem kann der Modulator wegen des geringen Preises emfohlen werden. Zum Programmieren und für Anwenderprogramme ist er allerdings nur bedingt geeignet, denn das unruhige Bild strengt die Augen nach einiger Zeit stark an.

Bezugsquelle:
PDC, Bad Homburg
und andere Fachhändler



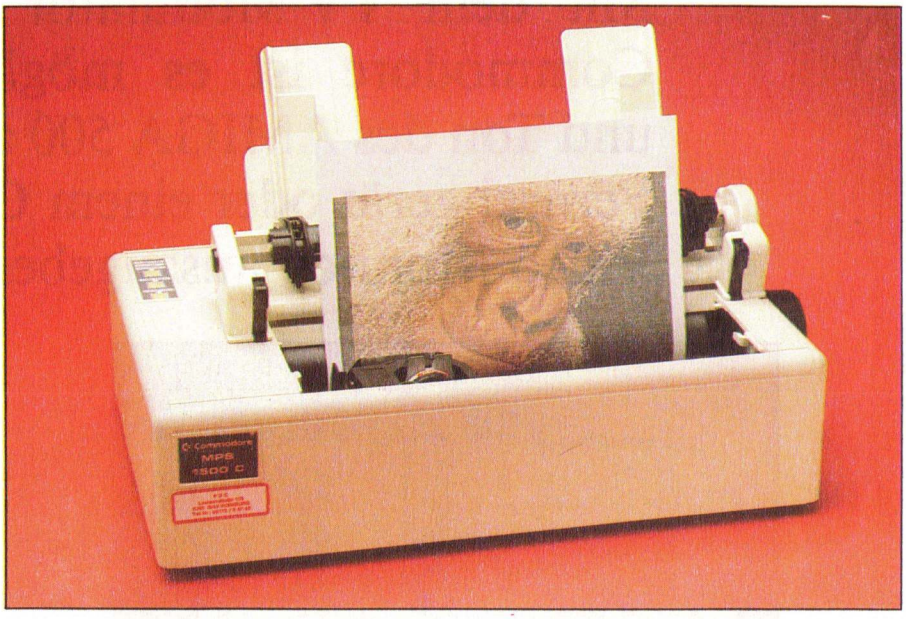
Für Textverarbeitung ist der TV-Modulator leider ungeeignet

Ein Farbdrucker für weniger als tausend Mark:

Der MPS 1500 C von COMMODORE

Überraschend präsentiert sich ein neues Druckermodell von Commodore auf dem Markt: Der MPS 1500 C. Besonders der letzte Buchstabe läßt aufhorchen, denn das C steht für 'COLOUR'. Ein Farbdrucker also, und das zu einem Preis von unter tausend Mark – das ist schon eine kleine Sensation.

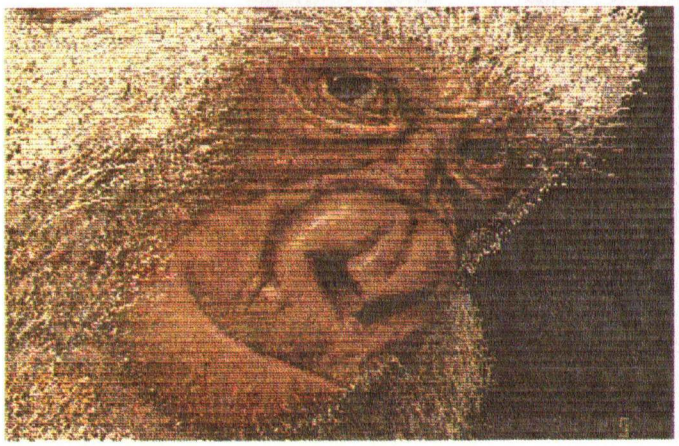
Beim ersten Blick auf diesen „Neuling" fällt das kastenförmige, etwas plump wirkende Gehäuse auf. Allerdings ist der Drucker sehr kompakt und klein in seinen Abmessungen. Der Aufbau ist einfach, wobei auch das geringe Gewicht des Kandidaten auffällt. Zum Lieferumfang gehört eine Papierführung, die das Einspannen von Einzelblättern sehr erleichtert. Sie gleiten direkt auf die Walze und können durch Drücken von 'Line Feed' eingezogen werden. Optional gibt es auch einen vollautomatischen Einzelblatteinzug, bei dem einfach ein Stapel Blätter eingelegt wird, der dann automatisch eingezogen wird. Ebenfalls zum Lieferumfang gehört ein unidirektionaler Traktor, der für die Verarbeitung von Endlospapier benötigt wird. Er ist mit einem Handgriff montiert, wobei die Blattführung nicht unbedingt abgenommen werden muß. Sie wird lediglich nach hinten umgelegt. Der Traktor ist ein sogenannter Zugtraktor. Dies hat den Nachteil, daß beim

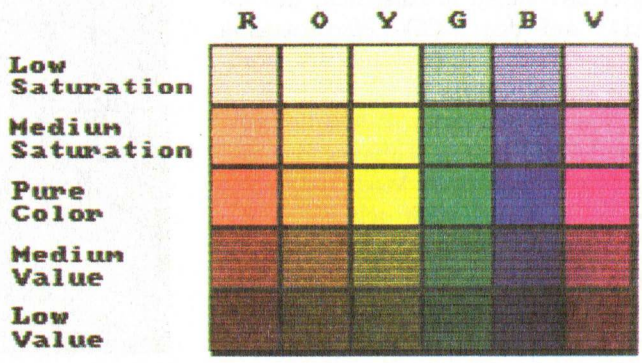


Einspannen und auch beim Abreißen jeweils ein Blatt verlorengeht, weil es für den Transport benötigt wird.

Der MPS 1500 C ist ein 9-Nadel-Drucker und baut somit die Zeichen aus einer 9★9-Matrix auf. Er kann jedoch auch in Schönschrift arbeiten; dann sorgt eine 18★9-Matrix für ein sauberes Schriftbild.

Das Gerät hat an der Frontseite nur Schalter für den Zeilen- und Blattvorschub; DIP-Schalter sind nicht vorhanden. Dies wird manchen Anwender sicher verwundern. Bei diesem Druckermodell ist man jedoch einen anderen Weg gegangen, der allerdings auch Nachteile hat: Die Einstellungen des Druckers werden im Dialogbetrieb vorgenommen, und der wird tatsächlich zwischen Mensch und Drucker vollzogen, also ohne Beeinflussung durch den Computer! In den Dialogmodus gelangt man, wenn beim Einschalten FF- und LF-Taste gedrückt werden. Auf dem Papier erscheint die Frage, welchen Drucker Sie emulieren wollen: Z. B. IBM. Sie lehnen ab (Taste FF) und es





erscheint EPSON JX-80. Sie sind einverstanden und drücken deshalb LF. Als nächstes erscheint die Frage nach dem Zeichensatz usw. Der Drukker schiebt dabei das Papier so lange vorwärts, bis die Schrift lesbar ist. Nach einem Tastendruck zieht er es wieder hinein und druckt weiter. Diese Art der Druckerprogrammierung ist sicherlich sehr eigentümlich, sie ist aber sehr leicht und kommt deshalb all jenen zugute, die sich nicht mit einer Druckeranpassung herumschlagen wollen. Freilich ist sie dann etwas lästig, wenn man nur einen Punkt (z. B. die Schriftart) ändern möchte und deshalb den ganzen Dialog durchlaufen muß.

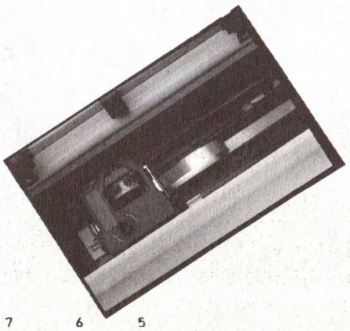
Der MPS 1500 C ist als Farbdrukker mit seinem Preis von knapp 1000 DM momentan sicher unschlagbar. Die Qualität ist ausreichend gut; sie dürfte sich mit dieser Drucktechnik kaum steigern lassen. Die Bedienung, besonders die Druckerprogrammierung im Dialogbetrieb, ist gewöhnungsbedürfig, aber sehr einfach. Auch das Einsetzen des Traktors ist ohne Schwierigkeiten möglich. Das Gerät ist zwar optisch etwas plump, ansonsten ist das Design jedoch gut gelungen. Durch die Kompabilität zum EPSON JX-80 ist der Betrieb am AMIGA kein Problem, da ein Druckertreiber bereits vorhanden ist. Der MPS 1500 C kann deshalb nur empfohlen werden; er wird sicher nicht enttäuschen.

(mn)

Schriftprobe

```
TO CONFIRM PRESS LF. TO CHANGE PRESS FF. TO END PRESS LOCAL

PRINTER EMULATED
EPSON JX 80     PROPRINTER     IBM G. P.     EPSON JX 80

CHARACTER SET
U.S.A    FRANCE    GERMANY    U.K.    DENMARK SWEDEN    ITALY    SPAIN

TYPE OF RIBBON
COLOURED     BLACK

CHARACTER DEFINITION
DRAFT     N.L.Q.

CHARACTER SPACING
10     12     15     17.1     20     24

ENABLE D.L.L.
NO 5.5K BYTE L.B     YES 2.5K BYTE L.B

LINE FEED
LF=LF+CR     LF = LF

CARRIAGE RETURN
CR = CR     CR=CR+LF

PAPER END DETECTION
YES     NO

LINE SPACING
1/6     7/72     1/8

SLASHED ZERO
NO     YES

DC1/DC3 PROCEDURE
NO     YES

FORM LENGTH
12     11     10     9     8     7     6     5

SKIP OVER PERFORATION (BOF)
0     1/3     1/2     2/3     1     2

BIDIRECTIONAL BIM
YES     NO

PROPORTIONAL SPACING
NO     YES

CHARACTER LENGTH
8 BITS     7 BITS

WOULD YOU LIKE TO STORE THESE PARAMETERS ?
YES     NO

CHANGES EXECUTED
```

Programmierung des MPS 1500



+ kompatibel mit EPSON JX-80, IBM, Prowriter
+ eigene Schrift ladbar
+ unidirektionaler Taktor
+ günstiger Preis

– NLQ sehr langsam
– keine Andruckwalze

MPS 1500 C

Daten:

Typ: 9 Nadel Matrixdrucker
Hersteller: Commodore
Kompatibel: EPSON JX-80
Druckertreiber: ja
Preis: 999 DM
PDC, Bad Homburg
und andere Fachhändler

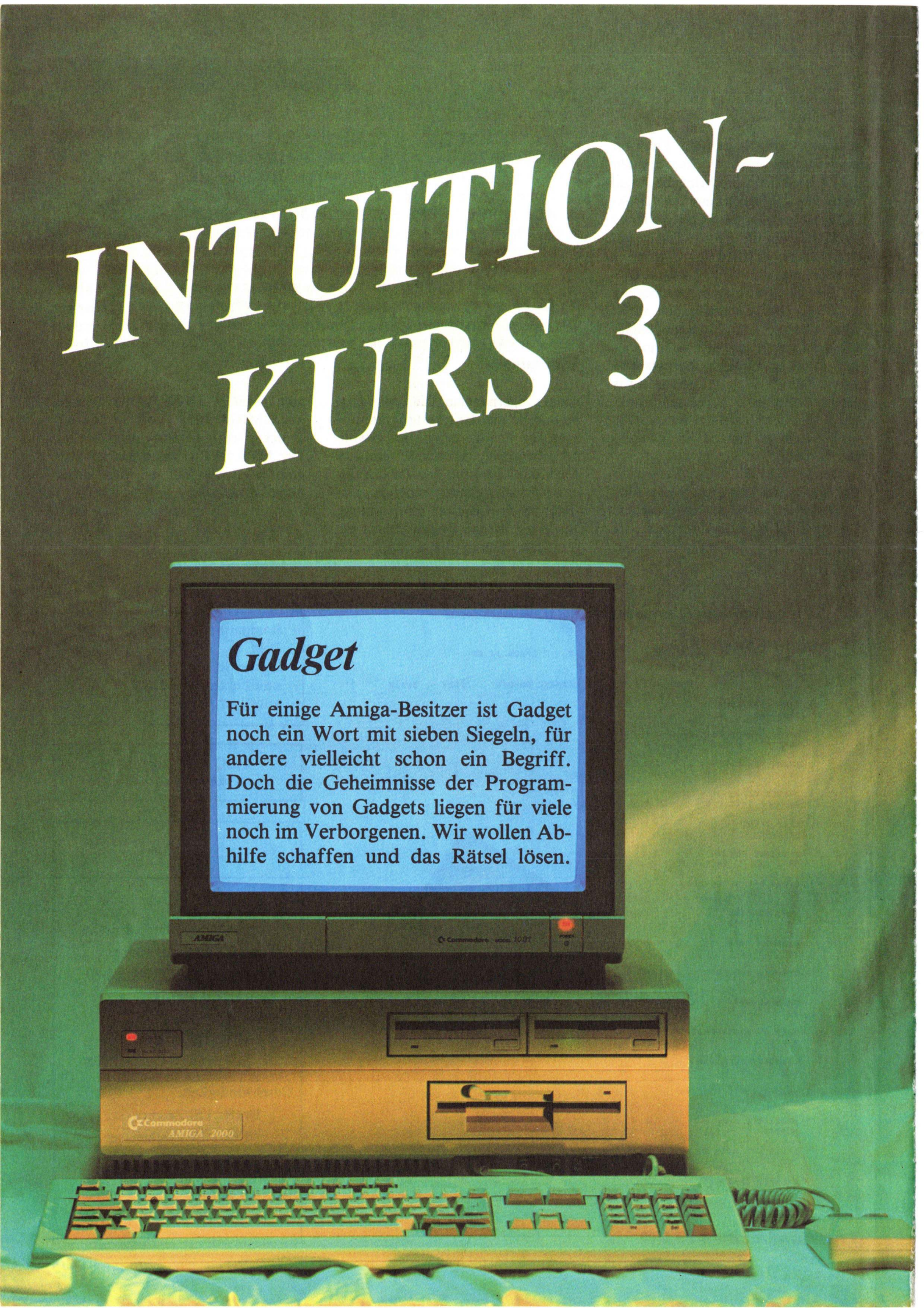
Geschwindigkeit:

Draft: 120 Zeichen/sec.
NLQ: 25 Zeichen/sec.
Grafik: ca.

Besonderheiten:

incl. Traktor

INTUITION-KURS 3



Gadget

Für einige Amiga-Besitzer ist Gadget noch ein Wort mit sieben Siegeln, für andere vielleicht schon ein Begriff. Doch die Geheimnisse der Programmierung von Gadgets liegen für viele noch im Verborgenen. Wir wollen Abhilfe schaffen und das Rätsel lösen.

Rückblick

In den vorangegangenen Teilen unseres Kurses wurde ausführlich auf Screens (Bildschirme) und Windows (Fenster) eingegangen. Bei der Behandlung von Bildschirmen waren wir noch nicht mit Gadgets in Berührung geraten. Anders war es beim Kapitel über die Fenster: Gadgets konnten einem Fenster zugewiesen werden. Dabei handelte es sich zweifellos um sogenannte Systemgadgets. Ihre Einbindung geht ohne größeren Aufwand vor sich und erfordert kein weiteres Hintergrundwissen: Ein bestimmtes Flag wurde in der NewWindow Struktur gesetzt, und schon konnte man ein Gadget bewundern.

Gadget, ein Rätsel?

Vielleicht werden viele Leser mit dem Begriff Gadget herzlich wenig anfangen können, besonders diejenigen unter Ihnen, die erst seit kurzer Zeit einen Amiga besitzen. Gadgets repräsentieren, allgemein ausgedrückt, jede Art von Feldern oder Symbolen, die mit der Maus angewählt werden und danach eine bestimmte Aufgabe erfüllen. Die einzige Ausnahme sind hierbei die Icons (Diskettensymbol oder Programmsymbol auf der Workbench); alle anderen mit dem Mauszeiger anwählbare Felder sind also Gadgets. Kaum ein Programm, das auf die grafikorientierte Benutzeroberfläche aufbaut, kommt um diese Gadgets herum. Gadgets sind zwar nicht die wichtigsten, doch sicher mit die nützlichsten Komponenten von Intuition. Das Programm 'Preferences' (Abb. 1), das sich auf jeder Workbench-Diskette findet, ist ein hervorragendes Beispiel dafür, wie vielfältig Gadgets eingesetzt werden können. Aufgabe der Gadgets ist es in diesem Programm, den Bildschirm neu zu positionieren oder die Farbe zu verändern.

Gadgets müssen immer mit einer Ausgabekomponente verkettet sein. Dies kann ein Bildschirm, Fenster oder Requester sein. Ebenso müssen sie, falls mehrere Gadgets für dasselbe Ausgabeelement bestimmt sind, untereinander zu einer Liste verbunden werden.

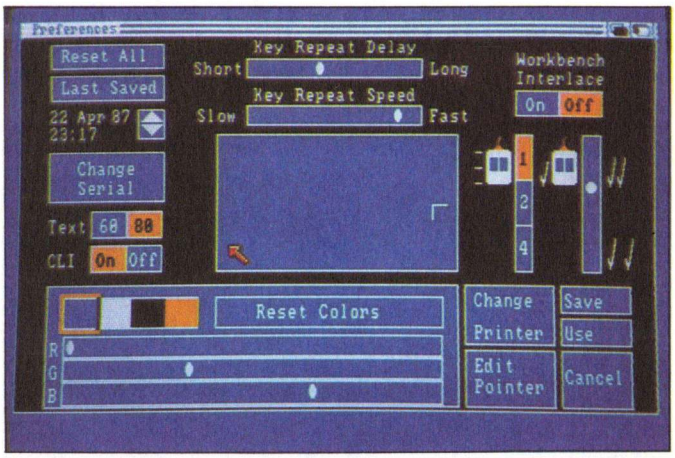


Abb.1:
Das Programm 'Preferences' beinhaltet eine Vielzahl von verschiedenen Gadget-Arten.

Vielfalt der Gadgets

Eine ganze Menge verschiedener Gadget-Arten steht dem Programmierer zur Verfügung. Man kann dabei die Gadgets in zwei Gruppen unterteilen: Die Systemgadgets (sie dürften den Lesern des letzten Kurs-Kapitels noch in Erinnerung sein) und die frei bestimmbaren Gadgets. Der Unterschied zwischen beiden besteht darin, daß Systemgadgets vom Betriebssystem des Amigas bereits fest definiert sind, beispielsweise werden sie verwendet, um ein Fenster in den Hinter- oder Vordergrund zu setzen. Die zweite Art von Gadgets kann dagegen vom Anwender frei programmiert werden. Hier gibt es wiederum vier Basis-Typen:

1. sogenannte Boolean-Gadgets, die nur ein 'True' oder 'False' bzw. 'Ja' oder Nein' als Antwort erlauben.
2. String-Gadgets, die es erlauben, einen Text, der vom Benutzer eingegeben werden kann, zu verarbeiten.
3. Integer-Gadgets sind spezielle String-Gadgets, die jedoch nur Integerwerte (also ganzzahlige Werte) erlauben.
4. Proportionale Gadgets, 'Schieber' oder andere frei bestimmbare Gadgets.

Die Systemgadgets sind immer an einen festen Platz im Fenster oder Bildschirm gebunden, es besteht auch keine Möglichkeit, ihre Form ohne größeren Aufwand zu verändern. Dagegen kann die Größe und Gestalt bei den selbsterstellten Gadgets frei gewählt werden. Die Phantasie des Programmiers kann tatsächlich in ungeahnte Sphären vordringen und dem Programm eventuell einen ungewöhnlichen Touch verleihen.

Boolean-Gadgets

Diese Gadgets sind sehr einfach aufgebaut. Sie bestehen meistens aus zwei einzelnen Gadgets, in denen ein positiver bzw. negativer Ausdruck dargestellt ist. Eine typische Verwendung dieses Typs findet man beispielsweise bei einem Systemrequester, das vom DOS ausgegeben wird, wenn ein kleinerer Fehler auftritt. Abb. 2 zeigt ein System-Requester, in dem unter anderem Boolean-Gadgets verwendet werden. Dem Anwender stehen in diesem Fall immer zwei Möglichkeiten zur Auswahl, eine positive und eine negative. Natürlich können auch mehrere Gadgets des Typs aufgerufen werden, doch kann ein einzelnes nur jeweils eine Antwort repräsentieren. Das Erstellen eines solches Gadgets setzt voraus,

daß in der Gadget-Struktur das Flag BOOLGADGET gesetzt wird. Dazu sollte man wissen, daß in der Programmiersprache C Variablen, die das Aussehen eines Gadgets festlegen, in sogenannten Strukturen zusammengefaßt werden. In anderen Programmiersprachen müssen die Werte für die einzelnen Variablen ähnlich zugewiesen werden.

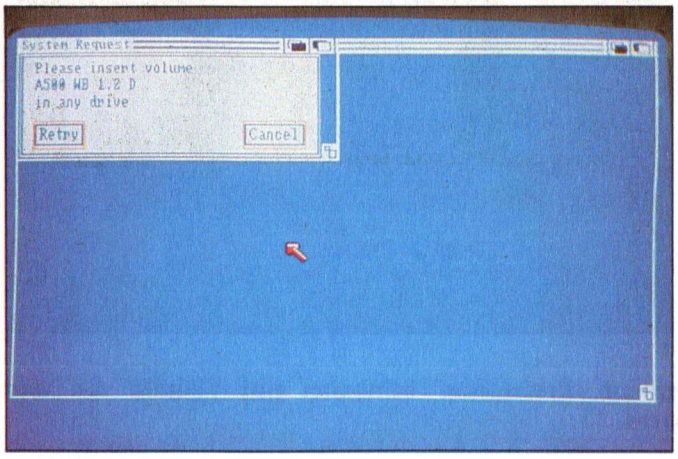


Abb.2:
Ein System-Requester mit zwei Boolean-Gadgets.

String- und Integer-Gadgets

String- oder Integer-Gadgets erlauben es einem Anwender, Text oder Integerzahlen einzugeben. Diese sehr nützliche Methode kann beispielsweise dazu verwendet werden, eine Datei von einem eigenen Programm aus zu laden. Das – gegenüber einem Boolean – etwas komplexere Gadget benötigt beim Erstellen einige Hilfsmittel. Unter anderem muß ein Puffer für die Zeichenkette reserviert werden, sowie zusätzlich ein weiterer, der im Hintergrund liegt. Der letztere dient dazu, einen gerade erstellten String in diesen Puffer zu verschieben. Die Puffer können beliebig groß gewählt werden, sollten aber nicht zuviel wertvollen Speicherplatz vergeuden. Außerdem wird eine zusätzliche Struktur verlangt: Die StringInfo-Struktur. Sie enthält einen Zeiger auf die eben erwähnten Puffer und einige andere sehr effiziente Wertzuweisungen. Weiterhin existieren Editiermöglichkeiten während der Eingabe des Strings. Die rechte Cursortaste bewegt das Editierzeichen nach rechts, die linke Cursortaste nach links. Die Shift-Taste – gleichzeitig gedrückt mit der rechten oder linken Cursortaste – bewirkt einen Sprung an das Ende bzw. den Anfang des eigegebenen Strings. 'Del' löscht das Zeichen unter dem Cursor, 'Backspace' das Zeichen rechts vom Cursor, 'Return' beendet die Eingabe. Durch Drücken der rechten AMIGA- sowie der Q-Taste wird die Eingabe rückgängig gemacht. Wird statt der Q-Taste dabei die X-Taste betätigt, wird der Eingabestring gelöscht. Ein reines String-Gadget setzt voraus, daß das Gadget-Type-Feld in der Gadget-Struktur auf STRGADGET gesetzt wird. Bei einem Integer-Gadget muß in der Activation-Variable das Flag LONGINT gesetzt werden. Abb. 3 zeigt eine typische Anwendung eines String-Gadgets.

Proportionale Gadgets

Die Proportionalen Gadgets sind die flexibelsten überhaupt. Mit ihrer Hilfe lassen sich beispielsweise im Programm „Preferences" die Farbwerte oder die Bildschirmposition ändern. Sie sehen: Man kann eine ganze Menge mit dieser Art von Gadgets anfangen. Entsprechend ist die Programmierung etwas komplizierter, da mehrere komplexe Werte festgelegt werden müssen. Abb. 4 stellt eine typische Verwendung eines Proportionalen Gadgets dar. Auf die Besonderheiten der Programmierung werde ich später bei den Erklärungen der einzelnen Strukturvariablen ausführlich eingehen.

Um ein solches Gadget zu erstellen, muß in der Gadget-Struktur (siehe Abschnitt String-Gadgets) die Gadget-Type-Variable auf PROPGADGET und das SpecialInfo-Feld einen Zeiger auf eine PropInfo-Struktur enthalten. Die PropInfo-Struktur beinhaltet weitere wichtige Variablen, die für ein Proportionales Gadget unbedingt notwendig sind.

Programmierung von Gadgets

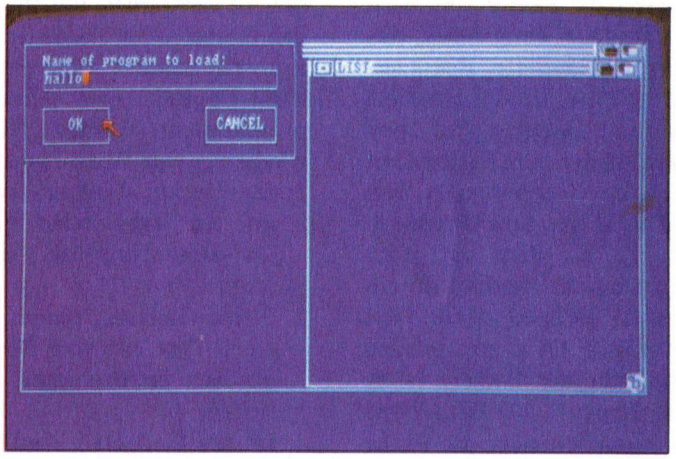


Abb.3:
Das AmigaBASIC verwendet ein String-Gadget zum Einlesen oder Abspeichern von Programmen.

Wieder stellt sich die Frage, welche Programmiersprache wohl am besten zum Erstellen von Gadgets geeignet ist. Dies ist leicht zu beanworten: C. Warum? C ist die Sprache für den Amiga, in C können alle Systemroutinen leicht aufgerufen werden. Nicht anders verhält es sich mit den Routinen, die ein Gadget zum Erstellen benötigt. Natürlich kann man Gadgets auch in Assembler produzieren, man geht damit jedoch einen komplizierteren Weg. Die Frage nach BASIC muß natürlich ebenfalls beantwortet werden. BASIC ist die Anfängersprache, leicht zu erlernen und weit verbreitet. Doch die Programmierung von Gadgets ist in BASIC alles andere als leicht, sehen wir einmal von den Sytemgadgets ab, die in einem Fenster oder Bildschirm eingebunden werden. Das BASIC des Amiga bietet auch keinerlei Befehle, die das Erstellen von Gadgets erleichtern. Für Fenster und Bildschirme sind sie vorhanden, Gadgtes hingegen wurden nicht berücksichtigt.

Gadgets in C

Jeder Amiga-Anwender kommt früher oder später zu dem Schluß, daß BASIC wohl nicht die beste Programmiersprache ist, besonders wenn er auf die Systemroutinen zurückgreifen möchte. Was bleibt, ist C oder Assembler. Diese Sprachen empfehlen sich allein durch die höheren Arbeitsgeschwindigkeiten. Aber auch für das Erstellen von Gadgets, bei dem nicht unbedingt eine hohe Geschwindigkeit benötigt wird, ist es angebracht, C zu verwenden. Laien in der Programmiersprache C müssen keine Angst haben: Der Ruf, der C vorrauseilt, nämlich eine komplizierte Sprache zu sein, ist weit übertrieben. Natürlich können C-Programme so aufgebaut sein, daß sogar ein Profi Schwierigkeiten bekommen wird, ihren Sinn zu erfassen. Doch die Programmierung von Gadgets in C ist mehr oder weniger ein recht leicht verständliche Angelegenheit, allerdings sollte man über Gadgets und deren Möglichkeiten Bescheid wissen. Aber das ist ja Sinn und Zweck dieses Kurses.

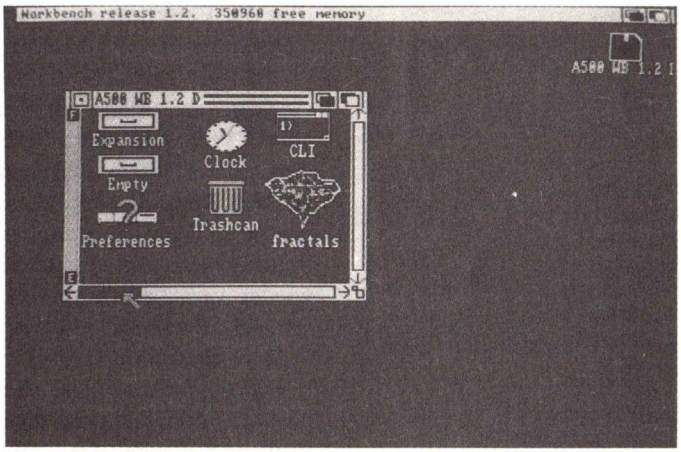


Abb.4:
Das Fenster verwendet Proportionale Gadgets, um den Inhalt zu verschieben.

Program-mierung

Vorweg möchte ich auf elementare Dinge der C-Programmierung hinweisen. In C geschriebene Programme haben immer den gleichen Aufbau. Zu Beginn finden sich stets allgemeine Zuweisungen, die Marke 'main()' läßt das eigentliche Hauptprogramm beginnen. Eine geschweifte öffnende ({) sowie eine geschweifte schließende Klammer (}) bestimmen den Umfang. C-Programme beginnen ihren Ablauf immer mit den Befehlen, die sich im 'main()' Abschnitt befinden. Natürlich können Sprünge vom Hauptprogramm in Unterprogramme führen und wieder zurück.

Noch ein allgemeiner Hinweis über die bereits häufig erwähnten Strukturen: In der Programmiersprache C stellen Strukturen eine Sammlung von Variablen dar. Eine Gadget-Struktur enthält beispielsweise alle nötigen Variablen zu dessen Erstellung. Den Variablen der Struktur können dann bestimmte Werte zugewiesen werden. Strukturen können leicht selbst erzeugt werden, in unserem Fall existieren sie bereits und sind auf der Compiler-Diskette in der Datei 'intuition.h' abgelegt.

Um Gadgets in C zu erstellen, müssen folgende Schritte unbedingt eingehalten werden.

1.: Das Erstellen einer Struktur für jedes Gadget, Wertzuweisung der Variablen.
2.: Das Festlegen einer Liste von Gadgets für jedes Ausgabeelement (Bildschirm, Fenster, Requester).
3.: Das Setzen der Gadget-Variable in der Screen-, Fenster- oder Requester-Struktur, die auf das erste Gadget der jeweiligen Liste zeigt.

Eventuell müssen noch weitere Strukturen definiert werden, jenachdem, ob Sie ein String- oder Proportionales Gadget erzeugen wollen. In dem nebenstehenden Kasten sind die einzelnen Strukturen mit den verschiedenen Variablen abgedruckt. Die Erklärungen der Variablen können Sie unter dem Titel 'Tabellarische Übersicht' nachlesen.

```
struct Gadget
{
 struct Gadget *NextGadget;
 SHORT LeftEdge, TopEdge;
 SHORT Width, Height;
 USHORT Flags;
 USHORT Activation;
 USHORT GadgetType;
 APTR GadgetRender;
 APTR SelectRender;
 struct IntuiText *GadgetText;
 LONG MutualExclude;
 APTR SpecialInfo;
 USHORT GadgetID;
 APTR UserData;
};


struct StringInfo
{
 UBYTE *Buffer;
 UBYTE *UndoBuffer;
 SHORT BufferPos;
 SHORT MaxChars;
 SHORT DispPos;
 SHORT UndoPos;
 SHORT NumChars;
 SHORT DispCount;
 SHORT CLeft;
 SHORT CRight;
```

```
 LONG LongInt;
 struct KeyMap *AltKeyMap;
};

struct PropInfo
{
 USHORT FLAGS;
 USHORT HorizPot;
 USHORT VertPot;
 USHORT HorizBody;
 USHORT VertBody;
 USHORT CWidth;
 USHORT CHeight;
 USHORT HPotRes;
 USHORT VPotRes;
 USHORT LeftBorder;
 USHORT RightBorder;
};
```

Gadget-Funktionen

Funktionen, die sich direkt auf Gadgets beziehen, sind nicht allzu zahlreich. Zwei Funktionen ermöglichen es, ein Gadget an eine bestehende Liste eines Fensters anzuhängen oder zu entfernen. Die Syntax, um ein Gadget anzuhängen, lautet:

AddGadget(AddPtr,Gadget,Position).

'AddPtr' stellt hierbei den Zeiger auf das Fenster dar, an welches das Gadget angehängt werden soll. 'Gadget' beinhaltet den Zeiger auf das neue Gadget. 'Position' definiert die Position des Gadgets in der bestehenden Gadget-Liste. Um ein Gadget aus einer bereits vorhandenen Liste zu entfernen, kann die Funktion mit folgender Syntax verwendet werden:

RemoveGadget(RemPtr,Gadget).

'RemPtr' ist bei dieser Funktion ein Zeiger auf ein Fenster, aus dem das Gadget entfernt werden soll. 'Gadget' repräsentiert den Zeiger auf das Gadget, das nicht mehr gebraucht wird. Man muß bei diesem Funktionspaar beachten, daß es nur bei denjenigen Gadgets funktioniert, die mit einem Fenster in Verbindung stehen. Gadgets, die mit einem Bildschirm oder Requester verkettet sind oder verkettet werden sollen, können mit diesen Funktionen nicht hinzugefügt oder entfernt werden.

Wenn ein Gadget nicht mehr verwendet wird, muß es nicht unbedingt mit der Funktion 'RemoveGadget()' entfernt werden. Ebensogut kann es nur ausgeschaltet werden. Dies erledigt die Funktion mit der Syntax:

OffGadget(Gadget,Ptr,Requester).

'Gadget' ist ein Zeiger auf das Gadget, das nicht mehr gebraucht wird. 'Ptr' stellt einen Zeiger auf ein Fenster dar, 'Requester' einen Zeiger auf ein Requester oder wird NULL gesetzt. Natürlich besteht die Möglichkeit, ein ausgeschaltetes Gadget wieder verarbeitungsbereit zu schalten. Die Funktion mit der Syntax:

OnGadget(Gadget,Ptr,Requester)

ermöglicht dies. Die Variablen innerhalb der Klammern sind identisch mit denen der 'OffGadget()' Funktion.

Eine weitere Funktion ermöglicht eine Art von Refresh (siehe Intuition-Kurs, Kapitel 2), doch werden nur die Gadgets neu erstellt. Das Aufrufen dieser Funktion kann mit nachstehender Syntax erfolgen:

RefreshGadgets(Gadgets,Ptr,Requester)

'Gadgets' zeigt auf das Gadget von dem der Refresh gestartet werden soll. 'Ptr' stellt einen Zeiger auf das Fenster dar. 'Requester' zeigt auf ein Requester oder wird NULL gesetzt.

Die letzte Funktion gestattet es, die Werte der ProfInfo-Struktur zu ändern. Sie wissen, daß diese Struktur für Proportionale Gadgets verwendet wird. Die Syntax hierzu lautet:

ModifyProp(Gadget,Ptr,Requester, Flags,HorizPot,VertPot,HorizBody, VertBody).

Die ersten drei Variablen sind identisch mit denen der 'OffGadget()'-Funktion. Die Erklärungen der verbliebenen fünf können Sie unter den Erläuterungen der PropInfo-Struktur nachlesen.

Programm-erläuterung

Unser Beispielprogramm erstellt in einem Fenster je ein String-, Boolean- sowie ein Proportionales Gadget. Das letztere Gadget zeigt nebenbei den momentan eingestellten Wert auf dem Bildschirm (CLI-Fenster) an. Das Programm ist in der Programmiersprache C geschrieben, so daß Sie einen C-Compiler benötigen. In meinem Fall habe ich einen Lattice-Compiler gewählt, aber das Umsetzen auf den Aztec dürfte keine großen Schwierigkeiten bereiten. Der Umfang des Programmes täuscht ein wenig, eine ganze Reihe von Zeilen sind mit Kommentar oder mit nur mit einem Zeichen belegt. Die Kommentarzeilen müssen natürlich nicht abgetippt werden. Auch die Numerierung des Listings dient nur zur besseren Eingabe und darf unter keinen Umständen mit eingegeben werden.

Doch nun zur Programmerklärung. Zu Beginn des Listings, in Zeile 11 und 12, werden zwei Dateien eingebunden. Die Datei 'types.h' befindet sich hierbei im Ordner 'exec' und enthält unter anderem Variablen-Definitionen wie 'void', 'UBYTE' oder 'int'. Es ist sinnvoll, diese Datei immer einzubinden, da die Definitionen sehr oft benötigt werden. Die zweite Datei nennt sich 'intuition.h' und beinhaltet vorgefertigte Intuition-Strukturen und dazugehörige Flagdefinitionen. In den Zeilen 17 – 23 erhalten die verschiedenen Strukturen einen Zeiger zugewiesen. Die Zeiger werden im fortlaufenden Programm unbedingt benötigt. Die Variablen-Definition ist der nächste Programmschritt. 'STRINGSIZE' legt in unserem Listing die Größe des Puffers für das String-Gadget fest. Die zweite erklärt die Versionsnummer der Libraries. Die Zeilen 32 und 33 definieren den Puffer sowie den Undo-Puffer des Stringgadget. Der nächste Programmteil beinhaltet alle Strukturen, die im Programm benötigt werden. Zuerst erscheint die IntuitText-Struktur, die noch nicht erklärt wurde, danach die String-Info-Struktur, die unbedingt für das String-Gadget gebraucht wird. Sie enthält unter anderem einen Zeiger auf die beiden Puffer. Die PropInfo-Struktur ist für das Proportionale Gadget zu verwenden. Nach diesen drei Stukturen folgen die der Gadgets. Die erste Struktur ist die des Boolean-Gadgets, dann folgen die des Proportionalen und des String-Gadgets. Beachten Sie bitte die Reihenfolge der Liste (gadget3-gadget2-gadget1). Wenn Sie ein eigenes Programm schreiben, muß die Liste genauso aufgebaut werden, ansonsten ist der Compiler nicht imstande, die Liste zu verketten. In den Zeilen 134 bis 150 befindet sich die NewWindow-Struktur für das Fenster. Einigen Le-

sern ist diese Struktur nicht unbekannt, sie wurde im letzten Teil des Intuition-Kurses ausführlich behandelt.

Mit der NewWindow-Struktur ist der sogenannte Programmkopf beendet; es folgt das Hauptprogramm mit der Zuweisung von zwei Variablen als ULONG- bzw. USHORT-Wert. Diese Variablen werden später zur Abfrage verwendet. Ihnen werden Werte der IntuiMessage-Struktur zugewiesen.

Als nächstes folgt die Intuition-Library. Wie Sie vielleicht wissen, ist das Verwenden von Routinen einer Library nur dann gestattet, wenn diese vorher geöffnet wurde. Nicht anders verhält es sich in unserem Programm. Nach dem Öffnen der Library wird eine Funktion von Intuition verwendet (OpenWindow(...)). Sie öffnet das Fenster, mit dem die Gadgets verkettet werden. Zeile 143 weist hierbei auf das erste Gadget der bestehenden Liste. Das Öffnen der Library sowie des Fensters bewirkt bei einem erfolglosen Versuch den Programmabbruch.

Die Initialisierung des Rastportes ist nicht umbedingt notwendig, kann aber bei späteren Programmerweiterungen nützlich sein. Ab Zeile 184 beginnt die Hauptschleife. In ihr befindet sich die Abfrage der Gadgets. Unter 'CLOSEWINDOW' wird das Schließen des Fensters sowie das Beenden des Programms veranlaßt. Hierbei wird in ein Unterprogramm gesprungen. 'GADGETUP' druckt die momemtanen Werte des Proportionalen Gadgets aus und 'GADGETDOWN' zeigt an, ob das Boolean-Gadget angewählt wurde. Nach diesen Abfragen werden sämtliche Klammern geschlossen und das Unterprogramm, das mit dem Label 'Schluss()' angesprungen wird, schließt alle Einrichtungen (Fenster, Intuition-Library).

Wenn Sie das Programm abtippen, können Sie in den einzelnen Strukturen einmal andere Werte sowie Flags einsetzen. Dadurch werden Sie sehr schnell die Wirkungsweise und den Nutzen der Gadgets kennenlernen.

Wie geht's weiter?

Das vierte Kapitel meines Intuition-Kurses befaßt sich ausführlich mit der Programmierung und Erläuterung von Menüs aller Art. Wenn Sie auch diesen Teil wieder verfolgen, können Sie demnächst auch Menüs und Submenüs (Untermenüs) erstellen und in Ihre eigenen Programme einbinden. Also – bis zur nächsten Ausgabe von KICKSTART!

(AK)

Tabellarische Übersicht

Die Bedeutungen der Variablen in der Gadget-Struktur. Durch gezieltes Zuweisen der Variablen kann der Verwendungszweck des Gadgets genau bestimmt werden.

NextGadget
Ein Zeiger auf das nächste Gadget in einer Liste. Das letzte hat den Next-Gadget-Wert NULL.

LeftEdge
Die X-Position der Gadget-Anwählbox.

TopEdge
Die Y-Position der Gadget-Anwählbox.

Width
Die Breite der Gadget-Anwählbox.

Height
Die Höhe der Gadget-Anwählbox.

Diese Werte können absolut oder relativ zu den Werten eines Fensters, Bildschirms oder Requesters sein. Die beiden ersten Variablen sind relativ zur Basisdimensionierung, wenn die GRELRIGHT- und GRELBOTTOM-Flags gesetzt sind. Die Breite und Höhe sind absolut, wenn die Flags GRELHEIGHT und GRELWIDTH gesetzt sind.

Flags

Es besteht die Möglichkeit, folgende Flags zu setzen:

GADGHIGHBITS
Wenn ein Gadget angewählt wird, stehen vier Möglichkeiten zur Veranschaulichung bereit.

GADGHCOMP
komplementiert alle Bits, die sich in der Select Box befinden.

GADGHBOX
erzeugt eine Box um die Gadget-Select-Box.

GADGHIMAGE
erzeugt einen alternativen Rand oder ein neues Bild.

GADGHNONE
wird gesetzt, wenn keine Veränderungen eintreten sollen.

GADGIMAGE
Verwenden Sie dieses Flag, wenn Sie die Variable GadgetRender nicht NULL gesetzt haben. Bei einer Bildumrandung wird es gesetzt, bei einem einfachen Rand (Border) nicht.

GRELBOTTOM
wird gesetzt, wenn die Gadget-Variable TopEdge relativ zum Boden des Anzeige-Elementes sein soll, ansonsten ist TopEdge relativ zur Decke.

GRELRIGHT
wird gesetzt, falls die Gadget-Variable LeftEdge ein relatives 'offset' zur rechten Ecke des Anzeigeelementes sein soll, anderweitig ist sie relativ zur linken Ecke.

GRELWIDTH
wird gesetzt, wenn die Gadget-Variable Width eine Zunahme der Breite innerhalb des Anzeigeelementes erzeugen soll, ansonsten ist Width ein absoluter Wert.

GRELHEIGHT
wird gesetzt, wenn die Gadget-Variable Height eine Zunahme der Höhe innerhalb des Anzeigeelementes erzeugen soll, anderweitig ist Height ein absoluter Wert.

SELECTED
ermöglicht die sofortige Verwendung eines Gadgets. Andernfalls ist das Gadget nicht sofort bearbeitungsbereit.

GADGDISABLED
setzt das Gadget in einen nicht zu verwendenen Zustand. Wollen Sie den Status eines Gadgets später ändern, kann das über die Funktionen OnGadget() und OffGadget() erfolgen. Dieses Flag wird nicht gebraucht, wenn Sie das Gadget immer anwählbar wünschen.

Activation Flags
Diese Variablen können in der Activation-Variablen der Gadget-Struktur gesetzt werden.

TOGGLESELECT
wird gesetzt, wenn der Ein/Aus Status des Gadgets bei jedem Anwählen wechselt.

GADGIMMEDIATE

zeigt sofort an, ob Sie das Gadget angewählt haben.

RELVERIFY
ermöglicht nur ein Verarbeiten des Gadgets, wenn der Zeiger sich über dem Gadget befindet und mit dem Anwählknopf der Maus angewählt wird.

ENDGADGET
Dieses Flag betrifft nur Gadgets, die für ein Requester bestimmt sind. Um ein Requester von der Ausgabe verschwinden zu lassen, muß ein Gadget angewählt werden, indem dieses Flag gesetzt ist.

FOLLOWMOUSE
Wird ein Gadget angewählt, wo dieses Flag gesetzt ist, möchte der Anwender die Mausposition bei jeder Bewegung erhalten.

Die folgenden Flags können gesetzt werden, wenn Sie die Größe des Fensterrandes bestimmten Ausgangskoordinaten anpassen wollen.

RIGHTBORDER
wird gesetzt, wenn die Höhe des rechten Fensterrandes als Ausgangskoordinate dienen soll.

LEFTBORDER
wird gesetzt, wenn die Höhe des linken Fensterrandes als Ausgangskoordinate dienen soll.

TOPBORDER
wird gesetzt, wenn die Breite des Randes vom Fensterkopf angepaßt werden soll.

BOTTOMBORDER
wird gesetzt, wenn die Breite des Randes vom Fensterboden angepaßt werden soll.

Die nun folgenden Flags können bei String-Gadgets gesetzt werden.

STRINGCENTER
bewirkt, daß der Text eines Strings zentriert wird, wenn er umrandet ist.

STRINGRIGHT
bewirkt, daß der Text des Strings rechts am umrandeten Gadget beginnt.

LONGINT
ermöglicht das Setzen eines 32 Bit Integer Wertes in einem normalen String-Gadget. Es muß aber ebenso wie beim String-Gadget ein Eingabepuffer mit einem Integer String initialisiert werden.

ALTKEYMAP
erklärt, daß eine alternative Key-Map vorhanden ist. Ebenso brauchen Sie einen Zeiger auf eine 'Key-Map', der in der String-Info-Struktur-Variablen AltKeyMap gesetzt sein muß.

GadgetType
Diese Variable legt fest, welcher Typ eines Gadgets ausgegeben werden soll und für welches Ausgabemedium es bestimmt ist. Sie müssen eins der folgenden Flags setzen, um den Typ festzulegen.

BOOLGADGET
erläutert, daß es sich um ein einfaches 'JA/Nein'-Gadget handelt.

STRGADGET
wird gesetzt, wenn es sich um String-Gadget oder Integer-Gadget handeln soll. Beim letzteren muß jedoch das LONGINT-Flag gesetzt werden.

PROPGADGET
wird gesetzt, wenn es sich um ein Proportionales Gadget handeln soll.

Die nachstehenden Flags teilen Intuition mit, für welches Medium das Gadget bestimmt ist.

SCRGADGET
wird gesetzt, wenn es sich um ein Screen-Gadget handelt.

GZZGADGET
wird gesetzt, wenn es für ein Gimmezero-zero-Fenster bestimmt ist.

REQGADGET
wird gesetzt, wenn es für ein Requester bestimmt ist.

GadgetRender
weist mit einem Zeiger auf eine Image- oder Border-Struktur, welche die Grafik beinhaltet.

SelectRender
weist mit einem Zeiger auf ein alternatives Image (Bild) oder Border (Rand).

GadgetText
beinhaltet einen Zeiger auf eine IntuiText Struktur, welche Text beinhaltet, der nach der Ausgabe des Gadgets ausgegeben werden soll.

MutualExclude
wird verwendet, um zu verdeutlichen welche Gadgets gemeinsam arbeiten. Intuition ingnoriert dieses Feld.

SpecialInfo
zeigt auf bestimmte Strukturen, die zur Unterstützung von Gadgets notwendig sind. Dabei handelt es sich um PropInfo- oder StringInfo-Strukturen.

GadgetID
Diese Variable ist für Ihren eigenen Gebrauch bestimmt. Sie können jeden Wert einsetzen, den Sie für sinnvoll halten. Die Variable wird von Intuition nicht beachtet.

UserData
Diese Variable kann auf selbst definierte Daten weisen, die für das Gadget bestimmt sind. Sie wird von Intuition ignoriert.

StringInfo-Variablen

Die nun folgenden Variablen sind in der StringInfo-Struktur enthalten und bestimmen somit weitere Werte für ein String-Gadget.

Die Bedeutungen der Variablen in der Struktur lauten:

Buffer
setzt einen Zeiger auf einen Puffer, welcher den Anfangsstring und den endgültigen String beinhaltet.

UndoBuffer
setzt einen Zeiger auf einen Puffer, welcher den letzten Eintrag beinhaltet. Dieser Puffer sollte groß genug gewählt werden, weil die Möglichkeit besteht, daß jede String-Info-Struktur auf diesen Puffer zurückgreifen kann.

MaxChars
ist die maximale Anzahl von Zeichen im Puffer.

BufferPos
legt die Cursorposition im Puffer fest.

DispPos
legt die Pufferposition des ersten Zeichens im Ausgabefeld fest.

Die folgenden Variablen verwaltet Intuition für Sie.

UndoPos
Zeichen Position im 'Undo' Puffer.

NumChars
Anzahl von Zeichen, die sich augenblicklich im Puffer befinden.

DispCount
Anzahl der Zeichen im Anzeigeelement.

CLeft
Linker Übertrag im Anzeigeelement.

CTop
Oberer Übertrag im Anzeigeelement.

LongInt
Nachdem der Anwender seinen Integer-Wert eingegeben hat, kann er den Wert dieser Variable überprüfen, vorausgesetzt es handelt sich um ein Integer-String-Gadget.

Die nächste Variablen werden von Intuition nicht mehr verwaltet.

AltKeyMap
Diese Variable zeigt auf Ihre eigenen erstellten Zeichen. Sie müssen jedoch ALTKEY-MAP im Activation-Flag in der Gadget-Struktur setzen.

PropInfo-Variablen

Die zweite Struktur, die zur Unterstützung von Gadgets verwendet werden kann, ist die PropInfo-Struktur, diese wird von Proportionalen Gadgets genutzt. Deren Variablen haben nachstehenden Verwendungszweck:

Flags
Die nachstehenden Flags können gesetzt werden:

AUTOKNOB
Setzen Sie dieses Flag, wenn Sie den Verschiebungsknopf in Ihrem Gadget gebrauchen möchten.

FREEHORIZ
Dieses Flag wird gesetzt, wenn der Knopf in horizontaler Richtung bewegt werden soll.

FREEVERT
Dieses Flag wird gesetzt, wenn der Knopf in vertikaler Richtung bewegt werden soll.

KNOBHIT
Bewirkt, daß der Verschiebungsknopf vom Anwender angewählt werden kann.

PROPBORDERLESS
Das Flag gibt keinen gezeichneten Rand um das Gadget aus.

Initialisieren Sie die nächsten zwei Variablen, bevor das Gadget ausgegeben wird. Nach dem 'Öffnen' des Gadgets können Sie die Werte dieser Variablen auslesen.

HorizPot
Der horizontaler Prozentanteil. Dieser kann zwischen 0 und FFFF (Hexadeziaml) gewählt werden. Setzen Sie Ihn auf 0x8000, wird der Verschiebungsknopf in der Mitte des Gadgets ausgegeben.

VertPot
Der vertikale Prozentanteil. Dieser entspricht der 'HorizPot'-Variable mit einer Ausnahme, daß in diesem Fall vertikale Werte eingeben werden.

Die nachstehenden Variablen teilen Ihnen mit, welche Prozentangabe die globale Ausgabe momentan besitzt.

HorizBody
Horizontaler Körper. Durch Festlegen dieser Variable wird die Sprungweite des Schiebers bestimmt, und zwar in horizontaler Richtung.

VertBody
Vertikaler Körper. Bestimmt die Sprungweite des Schiebers in vertikaler Richtung.

Intuition verwaltet und setzt die folgenden Variablen für Sie.

CWidth
Die tatsächliche Ausgabebreite des Gadgets.

CHeight
Die tatsächliche Ausgabehöhe des Gadgets.

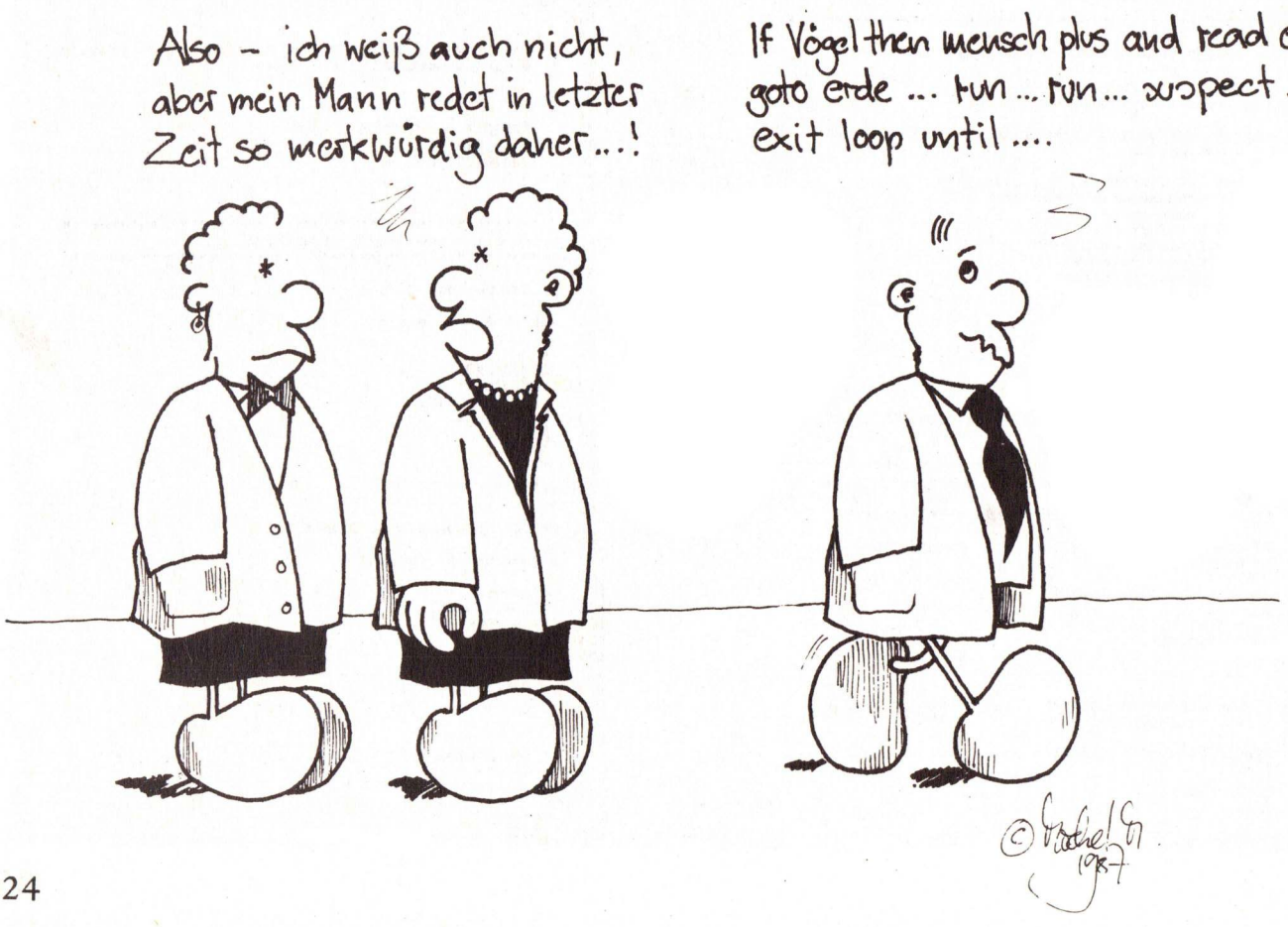
HPotRes, VPotRes
'Pot' Zunahme.

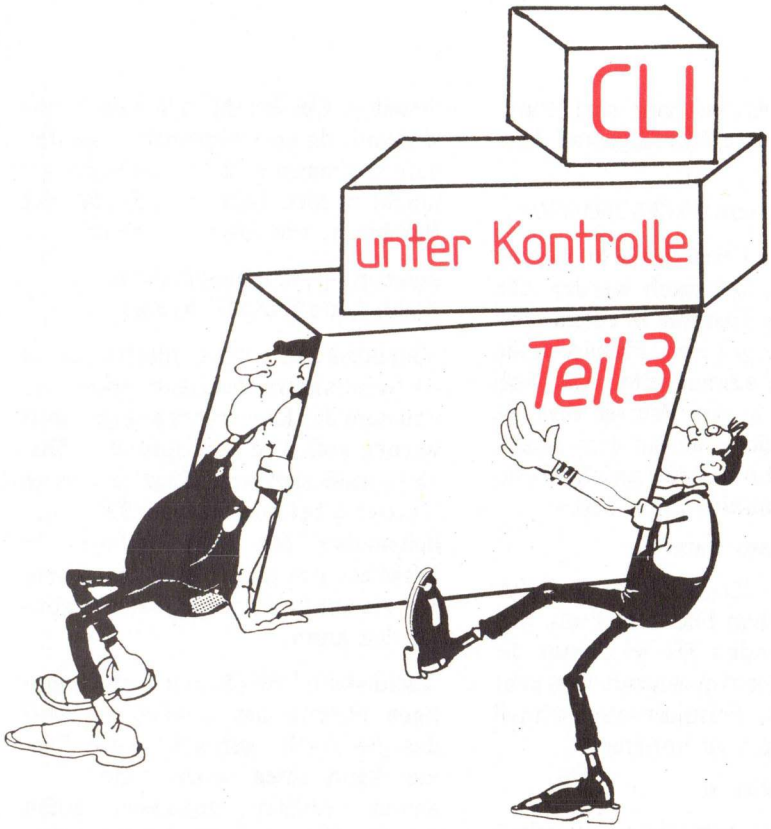
LeftBorder
Tatsächlicher linker Rand.

TopBorder
Tatsächlicher oberer Rand.

```
/*****************************************************
*                                                    *
*        Beispielprogramm von Gadgets                *
*        Autor: Andreas Kraemer                      *
*        KICKSTART 9.1987                            *
*                                                    *
*****************************************************/

/* Zwei Dateien werden eingebunden */

#include <exec/types.h>
#include <intuition/intuition.h>

/* Zuweisungen von verschiedenen Strukturen mit *
*  einem Zeiger.                                 */

struct IntuitionBase *IntuitionBase;
struct IntuiMessage *message;
struct Image Image;
struct RastPort *Port;
struct IntuiMessage *Nachricht;
struct Window *Window;
struct Message *GetMsg();

/* Variablen Definition */

#define STRINGSIZE 512
#define INTUITION_REV 29

/* Festlegung des Puffers fuer das String Gadget */

UBYTE DefString[STRINGSIZE]="STRINGGADGET";
UBYTE Undo[STRINGSIZE];

/*************************************************
* Alle notwendigen Strukturen, die fuer Gadgets  *
* notwendig sind.                                *
*************************************************/

/* Die IntuiText Struktur ist notwendig fuer das *
*  Boolean Gadget.                              */

struct IntuiText text =
{
 1,0,JAM1,0,0,
 "BOOLGADGET",
 0
};

/* Die StringInfo Struktur ist notwendig fuer das *
*  String Gadget.                                */

struct StringInfo TextString =
{
DefString,
Undo,
0,
STRINGSIZE,
0,0,
5,
0,0,0,
NULL,
0,
NULL
};

/* Die PropInfo Struktur wird fuer das Proportionale *
*  Gadget benoetigt.                                */

struct PropInfo Prop =
{
 FREEHORIZ | AUTOKNOB,
 0x8000,0x8000,
 0x800,0x800,0,0,0,
 0,0,0
};

/*************************************************
* Die einzelnen Gadget Strukturen.               *
*************************************************/

/* Das Boolean Gadget */

struct Gadget gadget3 =
{
 0,
 200,20,90,10,
 GADGHBOX,
 GADGIMMEDIATE,
 BOOLGADGET,
 NULL,0,
 &text,0,
 NULL,
 NULL,0
};

/* Das Proportinonle Gadget */

struct Gadget gadget2 =
{
 &gadget3,
 90,60,140,20,
 GADGHCOMP,
 RELVERIFY,
 PROPGADGET,
 (APTR)&Image,0,NULL,0,
 (APTR)&Prop,
 NULL,0
};

/* Das String Gadget */

struct Gadget gadget1 =
{
 &gadget2,
 10,20,120,10,
```

```
117   GADGHBOX,
118   STRINGCENTER,
119   STRGADGET,
120   NULL,
121   0,
122   NULL,
123   0,
124   (APTR)&TextString,
125   NULL,
126   0
127  };
128
129  /************************************************
130  * Die NewWindow Struktur. Mit diesem Fenster    *
131  * werden die Gadgets verkettet.                 *
132  ************************************************/
133
134  struct NewWindow Fenster =
135  {
136  0,0,
137  320,100,
138  2,3,
139   CLOSEWINDOW | REFRESHWINDOW
140  | GADGETDOWN | GADGETUP,
141  WINDOWDEPTH | WINDOWSIZING | WINDOWDRAG | REPORTMOUSE
142  | WINDOWCLOSE | SMART_REFRESH,
143  &gadget1,  /* Zeigt auf das erste Gadget der Liste */
144  NULL,
145  "KICKSTART FENSTER",
146  NULL,
147  NULL,
148  0,0,520,200,
149  WBENCHSCREEN,
150  };
151
152  /*******************************************************
153  *                                                      *
154  * Beginn des Hauptprogrammes                           *
155  *                                                      *
156  *******************************************************/
157
158  main()
159  {
160
161  /* Zuweisung von zwei Variablen als ULONG und *
162  *  USHORT Wert.                               */
163
164  ULONG NachrichtenArt;
165  USHORT code;
166
167  /* Oeffnen der Intuition Library */
168
169   IntuitionBase=(struct IntuitionBase *)
170       OpenLibrary("intuition.library",INTUITION_REV);
171   if(IntuitionBase==NULL)exit(FALSE);
172
173  /* Oeffnen des Fensters */
174
175   if((Window=(struct Window *)OpenWindow(&Fenster))==NULL)
176      exit(FALSE);
177
178  /* Initialisierung des Rastportes */
179
180  Port = Window->RPort;
181
182  /* Die Hauptschleife des Programms */
183
184  for(;;)
185  {
186
187  /*******************************************************
188  * Schleife die bestimmte Ereignisse abfragt und        *
189  * daraufhin Aufgaben durchfuehrt.                      *
190  *******************************************************/
191
192    if(Nachricht=(struct IntuiMessage *)
193       GetMsg(Window->UserPort))
194       {
195       NachrichtenArt=Nachricht->Class;
196       code=Nachricht->Code;
197       ReplyMsg(Nachricht);
198
199       switch(NachrichtenArt)
200        {
201         case CLOSEWINDOW : Schluss();
202                            break;
203         case GADGETUP    : printf("Propwerte: %x\n",Prop.HorizPot);
204                            break;
205         case GADGETDOWN  : printf("BOOLGADGET\n");
206                            break;
207        }  /* Ende switch */
208      }  /* Ende if */
209   }  /* Ende for */
210  return(0);
211  }  /* Ende des Hauptprogrammes */
212
213  /*******************************************************
214  * Das Unterprogramm zum Beenden des Programmes.        *
215  *******************************************************/
216
217  Schluss()
218  {
219     CloseWindow(Window);
220     CloseLibrary(IntuitionBase);
221     exit(0);
222     return(0);
223  }  /* Listing Ende */
```





Weiter geht's im Command-Line-Interpreter-Dschungel. Die zweite Oberfläche neben der Workbench hat so ihre Tücken, Amiga-Besitzer die noch keine Erfahrungen mit ähnlichen Oberflächen (z. B. MS Dos) gemacht haben, werden sich mit dem CLI schwertun. Das sinnvolle Anwenden der Befehle muß erst einmal gelernt sein. Doch bis dahin ist es für einen blutigen Laien ein weiter Weg. Sind diese Hürden überschritten, bietet der Amiga dem Benutzer jedoch eine Vielzahl von Anwendungen. Besonders gegenüber der Workbench, die leider nur eine begrenzte Anzahl von Befehlen zuläßt. Deshalb wurde dieser Kurs ins Leben gerufen, um neben dem mitgelieferten AmigaDos Handbuch den Command Line Interpreter näher zu erläutern.

In den zuvor behandelten Kursteilen habe ich die Befehle 'dir' 'list' bzw. die eng mit dem 'execute'-Befehl zusammenhängenen Schlüsselwörter (if,echo...) erläutert.

COPY und DISKCOPY

An dieser Stelle möchte ich auf den 'copy' und 'diskcopy' Befehl näher eingehen. Der erste Befehl ermöglicht das Kopieren von Dateien, der zweite hingegen kopiert eine ganze Diskette. Beide Befehle finden auch in der grafikorientierten Benutzeroberfläche, der Workbench, Verwendung, bieten aber kein so großes und flexibles Einsatzgebiet wie im CLI an.

Zum Syntax von 'copy':

copy [[from] Quelle][to Ziel]
[all] [quiet]

Klar, der Befehl gestattet es, einzelne (mehrere) Dateien zu kopieren. Doch was bedeuten 'all' und 'quiet'? 'all' veranlaßt, daß alle Dateien sowie alle Unterverzeichnisse (Subdirectories) des aktuellen Verzeichnisses in das mit 'Ziel' angegebene Verzeichnis (Directory) kopiert werden. Normalerweise werden die gerade kopierten Dateien auf dem Bildschirm angezeigt, der Ausdruck 'quiet' unterdrückt jedoch diese Ausgabe.

Die Standardbenutzung des Befehls 'copy' kann natürlich viele Gesichter haben. Hierzu einige Beispiele:

copy df0:c/Datei to ram:c

kopiert 'Datei' von Laufwerk 0 nach Laufwerk 1.

copy df0:Datei to df1:

kopiert 'Datei' von Laufwerk 0 im Verzeichnis 'c' ins RAM in den Ordner 'c'. Hierbei muß das Verzeichnis 'c' im RAM bereits existieren.

copy c to ram: all quiet

kopiert das komplette (all) Verzeichnis 'c' ins RAM, und zwar ohne Kopierangabe auf dem Bildschirm (quiet). Wäre 'c' nur ein File, würde nur dieses kopiert werden.

Weiterhin besteht die Möglichkeit, auch ganze Disketten mit dem 'copy'-Befehl zu vervielfältigen. Jedoch muß bei Nutzung eine längere Wartezeit gegenüber dem 'diskcopy'-Befehl hingenommen werden, außerdem ist es unbedingt notwendig, daß die Zieldiskette formatiert ist. Ein sinnvoller Einsatz von ganzen Kopien besteht darin, wenn Sie beispielsweise bereits eine Diskette besitzen, auf der sich nur wenige Programme befinden. Hier können Sie möglicherweise noch den Inhalt einer zweiten Diskette unterbringen und somit eine momentan doch noch sehr teure Diskette für andere Zwecke verwenden. Hierzu geben Sie lediglich nachstehendes ein:

copy df0: to df1:

Nicht nur einfach kopieren!

Nun, der Befehl ermöglicht nicht nur das Kopieren von Dateien oder Verzeichnissen, er kann noch mehr. Des weiteren ermöglicht der Befehl das Duplizieren von Dateien. Durch die Eingabe von

copy Datei to Datei1

wird 'Datei' verdoppelt, wobei der neu erstellten der Name 'Datei1' zugewiesen wurde. Natürlich können durch Verwendung dieses Befehlsausdruckes auch Dateien innerhalb einer Diskette kopiert werden.

copy Datei to c:Datei1

kopiert 'Datei' in das Unterverzeichnis 'c' mit dem neuen Namen 'Datei1'.

Weiterhin kann man mit dem Befehl 'copy' nicht nur Dateien in andere Verzeichnisse oder Laufwerke (df0, df1, df2, df3) kopieren, sondern auch auf andere logische Geräte. Als logische Geräte werden Laufwerke, Drucker, serieller und paralleler Port, ein Fenster, ein Gerät, welches eigentlich gar keines ist, und zwei weitere bezeichnet. Die Abkürzungen dieser logischen Geräte sind im CLI wie folgt festgelegt. 'df0, df1...' für die einzelnen Laufwerke, 'prt' für Drucker, 'ser' für den seriellen Port, 'par' für den parallelen Port, 'con'

für ein Fenster, 'nil' für das Gerät, welches keines ist. Die Hardware ist in diesem Fall ein Draht, der quasi in der Luft hängt, so daß die Daten, die an dieses Gerät geschickt werden, sozusagen im 'Nichts' verschwinden. Außerdem existiert als Geräteabkürzung noch 'raw', welches ähnlich zu handhaben ist wie das Gerät 'con und '★', das Eingeben der Tastatur ermöglicht. Im Bezug auf den 'copy'-Befehl ist das 'prt' (Drucker) Gerät natürlich sehr sinnvoll. Durch die Eingabe von 'copy Datei to prt: wird 'Datei' auf den Drucker ausgegeben. Sie sollten jedoch beachten, daß nur Dateien, die in ASCII vorliegen, ein akzeptables Druckergebnis liefern. Im binären Kode liefert der Drucker lediglich verworrene Zeichen. Manche Leser werden an dieser Stelle vielleicht stutzig, wissen sie doch, daß der Drucker über den parallelen Port mit dem Rechner verbunden wird. Und wofür dann das scheinbare Doppelspiel mit 'prt' und 'par'? Dazu ist zu sagen, daß 'par' diesen Port direkt anspricht, 'prt' hingegen lädt noch den momentan eingestellten Druckertreiber.

Das Gerät 'con' (Fenster) gestattet es, mit Hilfe des 'copy'-Befehls ein separates Ausgabefenster zu erstellen. Der Befehlsausdruck

copy Datei to con:20/20/300/100/

druckt 'Datei' in einem separaten Fenster aus. Beachten Sie wiederum, daß es sich um eine ASCII-Datei handelt, ansonsten wird der Ausdruck unübersichtlich.

Eine weitere Art mit 'con' und 'copy' zu arbeiten ermöglicht folgender Ausdruck:

copy ★ to con:20/20/300/100/

Der Stern (★) repräsentiert hierbei die Tastatur, demnach werden alle Eingaben der Tastatur in einem neuen Fenster ausgegeben. Probieren Sie es am besten einmal selber aus. Soll die Eingabe in das Fenster beendet werden, drücken Sie mit dem Mauszeiger das CLI-Fenster, anschließend die Tastenkombination CTRL-/.

copy ★ to ram:Datei

speichert die Eingaben der Tastatur im RAM in dem File 'Datei' ab. Mit CTRL-/ beenden Sie wiederum die Eingabe. Diese Angelegenheit ist sehr nützlich, um beispielsweise schnell etwas wichtiges zu notieren.

copy ram:Datei ★

gibt die eben gemachten Eingaben auf dem CLI-Fenster wieder aus.

Als letztes möchte ich die Möglichkeit besprechen, mit sogenannten Wildcards oder Jokern zu kopieren.

copy #?abcd#? to df0:s

kopiert alle Files die 'abcd' enthalten nach 'df0' in den Ordner 'c', der bereits vorhanden sein muß.
Sicherlich haben Sie bemerkt, daß man viel mehr Möglichkeiten besitzt, mit dem Befehl 'copy' zu arbeiten, als man vom Handbuch her vermutet.

DISKCOPY

Der 'diskcopy' Befehl gestattet es, ganze Kopien von einer Diskette zu erstellen. Der Befehl ist in soweit sehr sinnvoll, da von wichtigen Disketten eine sogenannte Sicherheitkopie gemacht werden kann. Der Syntax des Befehls ist wie folgt festgelegt:

diskcopy [from]Quelldiskette to Zieldiskette [NAME Name]

'Quelldiskette' stellt hierbei einen gültigen Namen des Laufwerkes dar, von dem der Kopiervorgang gestartet werden soll. Die zu kopierende Diskette muß sich unbedingt in diesem Laufwerk befinden. Setzen Sie sicherheitshalber den Schreibschutz der Diskette, um ganz sicher zu gehen, daß sie auf keinen Fall überschrieben werden kann.

'Zieldiskette' repräsentiert einen gültigen Namen des Laufwerkes, auf das die Kopie 'gezogen' wird. 'Name' kann einen neuen Namen der Kopie enthalten, ansonsten erhält das Duplikat den der Quelldiskette.

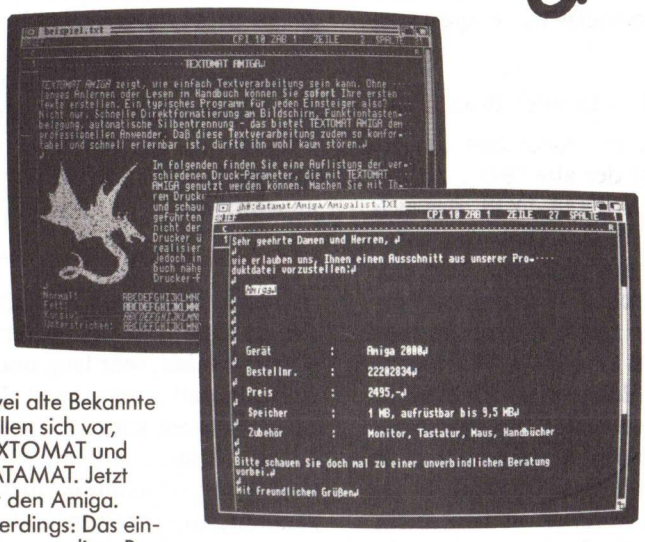
Zu beachten ist, daß sämtliche Daten, die sich möglicherweise auf der Zieldiskette befinden, gelöscht werden, da die Zieldiskette während des Kopiervorganges neu formatiert wird.

Ich hoffe, daß ich Ihnen mit diesem Teil des CLI-Kurses neue Aspekte vermitteln konnte. Vielleicht sind Sie in naher Zukunft ein wahrer CLI-Profi, wenn Sie weiterhin den Kurs aufmerksam mitverfolgen.

(AK)

Endlich

Textomat und Datamat für den Amiga!



Zwei alte Bekannte stellen sich vor, TEXTOMAT und DATAMAT. Jetzt für den Amiga. Allerdings: Das einzige, was diese Programme mit der ST- bzw. PC-Version gemeinsam haben, ist der Name – und natürlich der extrem günstige Preis. Aber das wird ohnehin sehr schnell deutlich, wenn man die phantastischen Leistungsmerkmale beider Programme betrachtet.

TEXTOMAT Amiga verfügt über eine überaus schnelle Direktformatierung mit allen Textattributen am Bildschirm (WYSIWYG). Selbst die Verknüpfung von Text und Grafik läßt sich am Bildschirm darstellen. Weitere Leistungsmerkmale: ● Hohe Geschwindigkeit bei der Eingabe und Bearbeitung von Texten ● Sämtliche Funktionen über Menüleisten oder Kurzbefehle anwählbar ● Automatische Silbentrennung ● Vielseitige Funktionstastenbelegung ● Drucklistenerstellung ● Beliebige Ausschnitte eines Amiga-Bildschirmes lassen sich abspeichern und im Text ausdrucken ● Grafiken, die im IFF-Format vorliegen, können im Text ausgedruckt werden ● Laden und Speichern über RS 232-Schnittstelle ● Beliebig viele Tabulatoren ● Multitaskingfähig ● Umfangreiche, sehr komfortable Druckeranpassungen ● Nicht kopiergeschützt ● Mit ausführlichem Handbuch.

TEXTOMAT Amiga DM 99,–

DATAMAT Amiga in Stichworten: ● Maximal 8 offene Dateien gleichzeitig ● Dateigröße: max. 2 Milliarden Zeichen ● Datensatzgröße: max. 64.000 Zeichen ● Maximal 2 Milliarden Datensätze ● Anzahl der Datenfelder: unbegrenzt ● Maximale Feldgröße: 32.000 Zeichen ● Paßwort-Schutz ● Dateien werden auf Massenspeicher (hohe Datensicherheit) bearbeitet ● Maximal 80 Indexfelder mit wählbarer Genauigkeit (1–999 Zeichen) ● Komfortable Such- und Selektierkriterien (Bereiche, Und-, Oderverknüpfung ...) ● Text-, Datums-, Zeit-, Zahlen- und Auswahl-Felder, mit Plausibilitätskontrolle ● Einlesen von IFF-Dateien ● Datenaustausch mit anderen Programmen möglich, wichtig für Serienbrieferstellung ● Programmsteuerung über Maus und Tastatur ● Frei gestaltete Bildschirmmaske bis zu maximal 5000 x 5000 Punkte groß ● Bildschirmmaske unterstützt Grafikelemente wie Rechteck, Kreis, Linie, Muster usw. ● Bildschirmmaske unterstützt verschiedene Textarten und -größen ● Mehrzeilige Textfelder mit Wortumbruch und Formatierungsmöglichkeiten ● Integrierter Druckermasken- und Listeneditor ● Nicht kopiergeschützt ● Mit ausführlichem Handbuch

DATAMAT Amiga DM 99,–

DATA BECKER

Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 31 00 10

BESTELL-COUPON

Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:

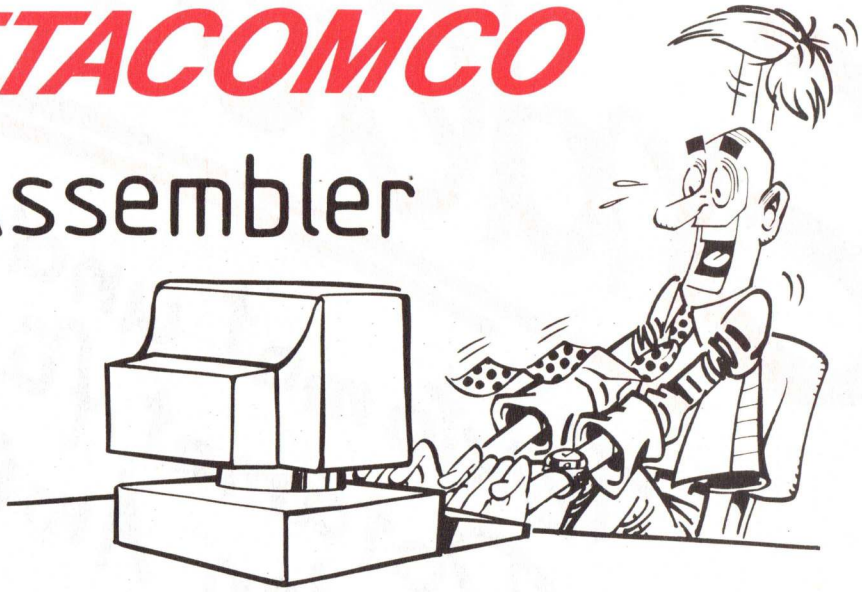
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

Name

Straße

Ort

Der METACOMCO Makro-Assembler



Wer hat nicht schon die aufwendigen Spiele bestaunt, die bunt und rasend schnell über den Bildschirm flimmern? Wer hat nicht mit Hochachtung, aber auch mit ein wenig Neid den schnellen Ablauf von Malprogrammen bewundert? Welche jungen Programmierer haben nicht versucht, ein schnelleres Spiel in irgendeiner höheren Sprache zusammenzubasteln, und wurden enttäuscht, daß alles zu langsam lief? Das Zauberwort heißt Assemblerprogrammierung. Damit ist aber längst noch nicht alles getan. Um in Assembler programmieren zu können, bedient man sich normalerweise eines Assembler-Paketes, das das Schreiben in Maschinensprache erleichtert.

In der Premierenausgabe von KICKSTART hatten wir den SEKA-Assembler vorgestellt und über seine guten und schlechten Eigenschaften berichtet. Nun testeten wir den Makro-Assembler von METACOMCO. Lesen Sie hier das Ergebnis dieses Tests.

Dem Assembler-Programmierer, der das SEKA-Programm besitzt, wird der grundlegende Unterschied zum METACOMCO-Assembler sofort auffallen. Beim SEKA handelt es sich um ein integriertes Assembler-Paket, das alles in einem beinhaltet: Editor, Assembler und Debugger werden geladen und je nach Bedarf benutzt. Der METACOMCO-Assembler geht einen anderen Weg. Editor, Assembler und Linker werden über den CLI nacheinander aufgerufen. Dies zeigt eindeutig, daß METACOMCO immer noch an ältere Rechnerkonzeptionen denkt, die mit sehr geringem Speicherplatz ausgestattet sind.

Der EDITOR: Ein alter Bekannter

Der bei diesem Assembler enthaltene Editor ist der alte 'ED', ein Bestandteil des AMIGA-DOS, das sich auf der Workbench-Diskette befindet. 'ED' ist ein typischer Vertreter eines Vollbildschirm-Editors, als Programm-Editor reicht er vollkommen aus. Er besitzt eine Fülle von Features, die die Editierung erleichtern. Der Editor beherrscht die deutschen Umlaute, was bei Kommentaren in deutscher Sprache recht praktisch ist. Seine Geschwindigkeit ist zwar nicht atemberaubend, doch man braucht bei der Ausführung einer Funktion keine Kaffeepause einzulegen.

Der Editor kennt zwei Arten von Befehlen: Die direkten Befehle und die Escape-Befehle. Entweder wird direkt nach der Betätigung einer Taste eine Funktion aufgerufen und ausgeführt, oder es wird zuerst die 'Esc'-Taste gedrückt und danach irgendeine zugelassene Taste. Befehle wie Block löschen, Block verschieben, Finden und Ersetzen usw. erleichtern das Leben des Programmierers.

Der Assembler: Ein Motorola-Standard

Hat man seinen Quelltext editiert, muß man ihn assemblieren. Dadurch wird ein Code erzeugt, der später mit dem Linker weiterverarbeitet wird. Der Assembler von METACOMCO ist eine Weiterentwicklung des ASSEM-Assemblers, der Bestandteil des Entwicklungs-Paketes von Commodore ist. Es handelt sich um ein ausgereiftes und makrofähiges Programm. Der erzeugte Code ist kompakt, die Geschwindigkeit sehr annehmbar.

Makros für aufwendigere Programme

In die Regel werden professionelle Programme, die in Assembler geschrieben sind, sehr lang und kompliziert, so daß man schnell den Überblick verlieren kann. Deswegen ist es immer ratsam, größere Programme in kleinere Abschnitte, auch Module genannt, zu schreiben und dann zusammenzulinken. Hat man bestimmte Module, die sich häufig wiederholen, so werden zwei verschiedene Varianten verwendet, um das lästige mehrmalige Eintippen eines Programmabschnittes zu verhindern.

Die am meisten verbreitete Lösung ist das Implementieren einer Unterroutine, die nur einmal im Programm erscheint und bei Bedarf immer wieder aus dem Hauptprogramm angesprungen werden kann.

Die zweite Lösung: Makros. Ein Makro ist im Prinzip nichts anderes als eine Unterroutine, also eine Folge von Befehlen, die der Reihe nach verarbeitet werden. Erst der Assembler macht bei der Interpretierung des Quelltextes den Unterschied zwischen Makros und Unterroutinen klar. Immer, wenn im Quelltext ein Makro aufgerufen wird, kopiert der Assembler an diese Stelle die gesamten Fol-

gebefehle, so daß kein Sprung an irgendwelche Stellen erforderlich ist.

Makros haben gegenüber Unterroutinen viele Vorteile und nur wenige Nachteile, zum Beispiel die Geschwindigkeit bei der Ausführung des Programms. Makros übernehmen Paramter auf eine sehr einfache Art. Subroutinen dagegen werden in der Regel indirekt adressiert, um Parameter weiterzugeben. Der größte Nachteil von Makros ist, daß die Programme erheblich länger werden. Der METACOMCO-Assembler erlaubt die Möglichkeit, ein Programm mit Makros zu versehen. Dafür gibt es im Sprachumfang des Assembler-Paketes genügend Anweisungen, mit denen sich solche Makros auf eine sehr komfortable Art und Weise programmieren lassen.

Jede Makro-Definition muß mit einem Namen versehen werden. Direkt nach dem Namen muß die Anweisung MACRO erfolgen. Die Definition wird mit ENDM (END Makro) abgeschlossen. Ferner bietet dieses Entwicklungspaket in Zusammenhang mit der sogenannten 'Conditional'-Anweisung die Möglichkeit, ein Makro gegebenfalls problemlos zu verlassen.

Der Linker: Ohne ihn läuft nichts

Dritter im Bunde dieses Programmpaketes ist ein Linker. Erst durch ihn wird ein ausführbares Programm erzeugt. Das Aufrufen des Linkers muß immer unabhängig davon erfolgen, ob man ein externes File mit dem Hauptprogramm aneinanderhängen möchte oder nicht. Das ist mit Sicherheit eine lästige Prozedur, wenn es sich um ein kleineres Programm handelt. Um die Bedienung des Assemblers und des Linkers zu erleichtern und zu automatisieren, existiert eine sogenannte Batch-Datei mit allen nötigen Parametern für den Assembler bzw. Linker. Man braucht nur diese Datei auszuführen und der Quellcode wird assembliert. Der Objektcode, der daraufhin erzeugt wird, wird von dem Linker übernommen und damit ein ausführbares Programm erzeugt.

Was noch mitgeliefert wird – und was fehlt!

Mit dem Assemblerpaket werden sämtliche Include-Files, die für die Entwicklung benötigt werden, geliefert. Ein Debugger fehlt völlig, was sehr schade ist und den Wert dieses Produkts mindert. Ein Debugger ist mit Sicherheit eine der wichtigsten Utilities während der Entstehung eines komplizierten Programms, da ja nur mit ihm eine vernünftige Fehlersuche möglich ist.

Die Dokumentation, die zur Zeit nur in englischer Sprache erhältlich ist, erklärt die Bedienung des Assemblers sehr knapp. Es ist kaum vorstellbar, daß ein Anfänger, der seine ersten Schritte in der Programmierung von Maschinensprache macht, mit dieser mitgelieferten Anleitung zurechtkommen könnte.

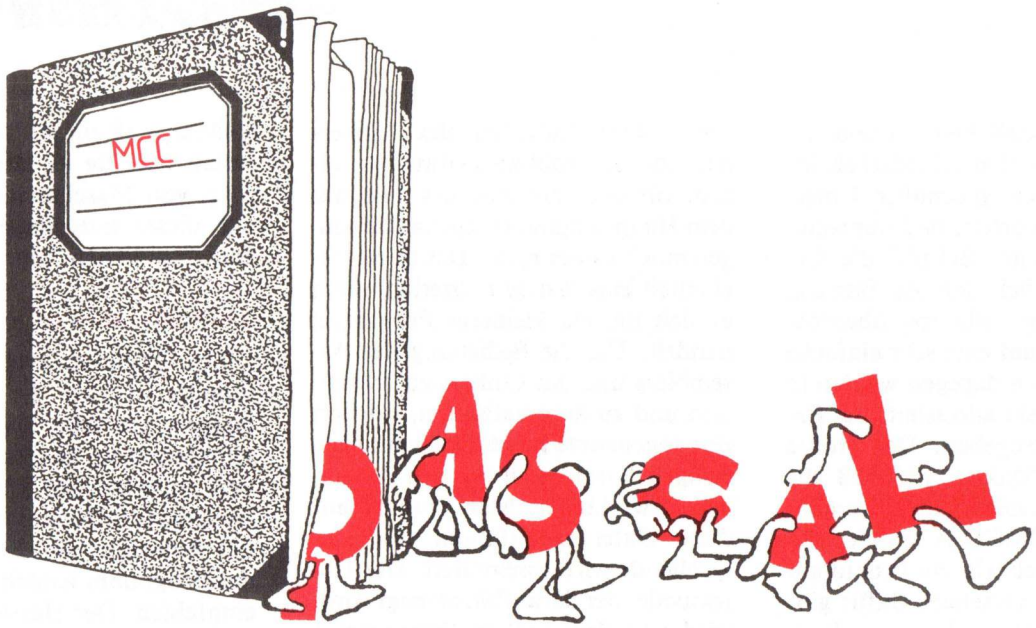
Ende gut, alles gut...

Der METACOMCO Makro-Assembler ist trotz des fehlenden Debuggers ein sehr komfortables Assemblerpaket, das zur Entwicklung von aufwendigen Programmen in Maschinensprache sehr geeignet ist. Für Einsteiger auf diesem Gebiet ist das Programm jedoch sicher nicht zu empfehlen. Der Hersteller sollte sich die Mühe machen, dieses Paket um eine bessere Dokumentation zu erweitern und einen Debugger einzufügen.

MCC Assembler, Preis: 198,– DM.

Philgerma GmbH
Ungererstr. 42
8000 München 40
Tel.: 089/395551

```
        INCLUDE "libraries/dos_lib.i"
        XREF    _DOSBase
        XDEF    _main

CALL    MACRO
        MOVE.L  A6,-(SP)
        MOVE.L  _DOSBase,A6
        JSR     _LVO\1(A6)
        MOVE.L  (SP)+,A6
        ENDM

_main   MOVE.L  SP,SAVESP
        CALL    Input
        MOVE.L  D0,D7           D7 holds stdin
        CALL    Output
        MOVE.L  D0,D6           D6 holds stdout

        LEA.L   PROMPT,A0       String ptr
        BSR.S   WRSTR           Print message

        LEA.L   BUFFER,A3       Get name buffer
        MOVE.L  D7,D1           Use stdin handle
        MOVE.L  A3,D2           Read buffer pointer
        ADDQ.L  #1,D2           Skip length byte
        MOVEQ   #30,D3          Maximum name size
        CALL    Read            Fetch name
        MOVE.B  D0,(A3)         Insert length

        LEA.L   MESS,A0         String ptr
        BSR.S   WRSTR           Print message
        LEA.L   BUFFER,A0       Ptr to name
        BSR.S   WRSTR           Print message

        MOVEQ   #0,D0           Zero return code
EX      MOVE.L  SAVESP,SP
        RTS

ERR     MOVEQ   #20,D1          Non zero return code
        BRA.S   EX

WRSTR   MOVE.L  D6,D1           Stdout  file handle
        MOVEQ   #0,D3           Clear length register
        MOVE.B  (A0)+,D3        Get byte length of string
        MOVE.L  A0,D2           String is write buffer
        CALL    Write
        RTS

        DATA
SAVESP  DS.L    1
BUFFER  DS.B    1.
        DS.B    30
MESS    DC.B    6,'Hello '
PROMPT  DC.B    18,'Enter your name : '
        END
```



Parlez-vous Pascal?

Wer vom PC auf den AMIGA umsteigt, will vielleicht die Hofsprache Pascal ebenfalls übernehmen. Wir testeten ein Pascal nach dem ISO-Standard, das dafür von Metacomco angeboten wird.

Pascal gehört auf dem PC zu den am meisten verbreiteten Sprachen. Dies ist sicher auch ein Qualitätsmerkmal. Die Sprache ist leicht erlernbar und ermöglicht strukturiertes Programmieren, zudem sind die Programme von System zu System leicht portierbar, wenn man denselben Dialekt benutzt. Wie bei allen verbreiteten Sprachen entstanden diverse Dialekte, die untereinander inkompatibel sind. Um diesem „Babylon" ein Ende zu setzen, wurde 1982 der ISO-7185-Standard entwickelt. An ihn hält sich das MCC-Pascal von Metacomco. Zwar wurden noch einige Befehle über den Standard hinaus definiert, doch kann man bei Programmen, die portabel sein müssen, auch gut ohne sie auskommen.

```
program hello (output);

begin
   writeln(' hello ');
end.
```

```
program sieve(output);

const SIZE = 8190;

var    flags : array[0..size] of boolean;
       i,j,prime,k,count,iter : integer;

begin
  writeln ('10 Iterationen');
  for iter := 1 to 10 do
    begin
      count := 0;
      for j := 0 to SIZE do
        begin
          flags[j] := true;
        end;
      for i := 0 to SIZE do
        begin
          if flags[i] then
          begin
            prime := i*2+3;
            k := i + prime;
            while k <= SIZE do
              begin
                flags[k] := false;
                k := k + prime;
              end;
            count := count + 1;
          end;
        end;
    end;
writeln (count,' Primzahlen gefunden');
end.
```

MCC-Pascal wird in der für Metacomco typischen Video-Buchhülle ausgeliefert. Zum Lieferumfang gehören eine Diskette und das Handbuch. Die Diskette ist bootfähig, meldet sich dann aber im 60-Zeichen-Modus und im grellsten Blau des Monats (der Programmierer war für die Farbeinstellungen wohl blind). Ein Tip am Rande: Wenn Sie die Datei „system-configuration" von einer anderen Workbenchdiskette auf MCC-Pascal kopieren, haben Sie die gleichen Voreinstellungen. Außerdem findet man auf der Diskette noch einige Textdateien, in denen nützliche Informationen über ED und das Arbeiten mit Pascal im CLI enthalten sind.

Das Handbuch

Das Heft, das mit MCC-Pascal ausgeliefert wird, kann getrost als Volltreffer bezeichnet werden. Es ist optisch gut gestaltet, inhaltlich in Ordnung und außerdem noch ausführlich. Dem Anfänger kann es als ausführliche Pascal-Erklärung dienen, dem Fortgeschrittenen ermöglicht es rasches Nachschlagen. Mit Anhängen zu Syntax, Compiler-Fehlermeldungen, Laufzeitfehlern und ISO-Standard gut bestückt, kann das Handbuch über ein kleines Manko leicht hinwegtrösten: Es wird auch in Deutschland in englischer Sprache ausgeliefert, ist aber insgesamt auch mit Schulenglisch zu verstehen.

Der Editor...

...muß eigentlich nicht mehr erläutert werden. Es handelt sich um den AMIGA-eigenen ED. Über die Bedienung dieses Editors gibt es meiner Meinung nach nur ein Urteil: Umständlich. Da der Editor jedoch keine besonderen Funktionen im Pascal-System erfüllt, kann er leicht gegen jeden anderen (z.B. EMACS auf der Extras-Diskette) ausgetauscht werden. Leider ist es auf diese Weise versäumt worden, dem System eine komfortable Fehler-Editierung mitzugeben.

Der Compiler

Der Compiler wird vom CLI aus aufgerufen. Er versieht seine Arbeit klaglos, solange man ihm fehlerfreie Programme vorsetzt. Falls nicht, fängt er an zu klagen, und zwar im Klartext. Man hat es ja schon gesehen: Fehlermeldung „error 0815 in line 4711, compilation aborted". Damit sind dann alle Klarheiten beseitigt. MCC-Pascal hat da wesentlich mehr zu bieten. Man erhält natürlich auch eine Fehlernummer, darüberhinaus aber noch die fehlerhafte Zeile angezeigt, eine Fehlermeldung im Klartext und der Fehler wird in der Zeile genau lokalisiert. Dann hat man die Auswahl, weiterzucompilieren oder abzubrechen. Die Nützlichkeit dieses Features ist nicht zu unterschätzen, da man nicht jeden Fehler einzeln editieren muß, sondern alle Fehler von Compiler finden läßt und dann in einer ED-Sitzung alle Schnitzer beseitigen kann. Wünschenswert wäre nur ein Editor, der die Fehlerbehebung unterstützen würde. Außerdem können beim Compilieren noch einige Optionen wie zum Beispiel eine List-File-Erstellung eingestellt werden, oder es kann die Verwendung der nicht ISO-Standard-Prozeduren Reset und Rewrite angezeigt werden. Der Compiler ist ausreichend schnell, erzeugt aber nur einen mäßig schnellen Code. Im Vergleich zu anderen Sprachen schneidet MCC-Pascal sehr schlecht ab. In der Tabelle sind einige Werte wiedergegeben. Die Daten der anderen Compiler sind jedoch nicht ohne weiteres vergleichbar, da Modula ja eine andere Sprache ist und die PC-Pascals an sich hardwaremäßige Nachteile haben. Trotzdem ist das MCC-Pascal mit Abstand am langsamsten. Hier wären noch einige Verbesserungen nötig. Vor allem, wenn man das schon legendäre Turbo-Pascal vor der AMIGA-Türe stehen sieht! Borland hat für dieses Jahr eine AMIGA-Version angekündigt, die MCC-Pascal das Leben schwer machen könnte.

Der Linker...

...ist ebenfalls ein alter Bekannter. Es handelt sich um den Alinker V 2.30, der auch im DOS-Handbuch des AMIGA beschrieben ist. Auch er zeichnet sich nicht gerade durch Schnelligkeit aus. Er kann ohne Probleme gegen andere ersetzt werden, mit denen man leicht die doppelte bis dreifache Geschwindigkeit erzielt. Die zum Linken benötigten Files befinden sich im l-Ordner. Wenn man das System nicht mit Pascal anstartet, muß man also darauf achten, daß man den l:-Ordner umassignt, da der Linker seine Link-Dateien ansonsten nicht findet.

Fazit

Wer jetzt fragt, ob man MCC-Pascal empfehlen könne, bekommt die klare Antwort: Ja! Zwar ist es in manchen Punkten etwas flügellahm. Aber Pascal ist eine beliebte (und an Hochschulen vielgelehrte) Sprache, und die MCC-Version ist die einzige mir bekannte Amiga-Adaption. Metacomco hat sich an einen Standard gehalten, und das wichtigste: MCC-Pascal versieht still und treu seinen Dienst. Und Zuverlässigkeit ist eine Stärke, die für manche Sekunde Compilierzeit entschädigen kann, wenn man dafür nicht ständig vor einem unbegründeten Absturz Angst haben muß. Sozusagen der Käfer unter den Programmiersprachen: Nicht der schnellste, aber er läuft und läuft und...

(chk)

Philgerma
Ungererstraße 42
8000 München 40

Preis:
MCC Pascal ISO V1.25 248 DM

+ Standardpascal (gute Portierbarkeit der Programme)
+ gutes Handbuch
+ Batchdatei zum kompletten Compilieren und Linken
+ Fehlermeldungen im Klartext
+ externe Routinen möglich
+ sehr zuverlässig

– kein eigener Editor
– nicht in die Workbench eingebunden
– kein direkter Zugriff auf das Betriebssystem
– sehr langsamer Code

Der Sprachumfang von MCC-Pascal

vordefinierte Konstanten:
maxint, true, false;

erlaubte einfache Variablentypen:
boolean, integer, char, real;

strukturierte Variablent.:
set, array, record, file, pointer;

mathematische Operationen:
abs, arctan, cos, div, exp, ln, mod, ord, pred, round, sin, sqr, sqrt, succ, trunc, +, −, ★, /;

logische Operationen:
and, or, not, =, <>, <, >, <=, >=, in;

nichtmathematische Standardfunktionen:
chr, eof, eoln;

Standardprozeduren:
dispose, get, new, pack, page, put, read, readln, reset, rewrite, unpack, write, writeln;

Programmanweisungen:
begin, case..of, end, for..to(downto), forward, goto, if..then..else, repeat..until, while..do, with.

	MCC-Pascal	MODULA-2	Turbo-P.	Compas-P.
Länge sieve	8364 Byte	12564 Byte	11701 Byte	9355 Byte
Länge hello	7172 Byte	5508 Byte	-------	-------
Compilierz.	19 sec	42 sec	< 1 sec	4 sec
Linkzeit	81 sec	101 sec		12 sec
Ausführungz.	65 sec	7 sec	14 sec	18 sec

Die File-Längen von Turbo-Pascal und Compas-P. können nicht direkt mit MCC-Pascal verglichen werden, da MS-DOS-Versionen, Laufzeiten mit einer Sidecar(V20-Prozessor) ermittelt.

Neue Version:

UBM-TEXT V2.2

Die deutsche Firma UBM hat ihr Textverarbeitungsprogramm überarbeitet und bietet nun die neue Version zum alten Preis an. In der Hoffnung, endlich einmal eine gute Textverarbeitung in Händen zu halten, wurde unser Test begonnen...

Der Start

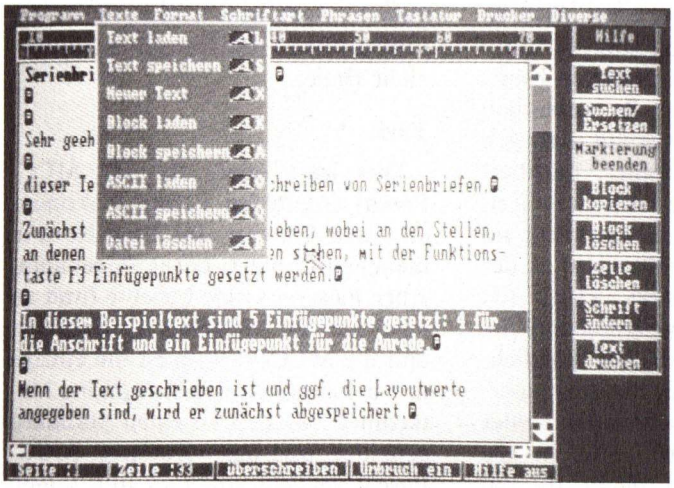
Nach dem Starten erscheint der schon von der ersten Version bekannte, übersichtliche Bildschirmaufbau. Die Kopfspalte zeigt die Tabulatormarken sowie Zeilenanfang und -ende an. Am unteren Bildschirmrand befindet sich eine weitere Anzeigezeile. Hier werden der momentane Modus (Einfügen/Überschreiben), die genaue Cursorposition und die Umbruchoption angezeigt. Auf der rechten Seite befinden sich noch einige Hilfsfunktionen, die sich jedoch auch unter dem Menüpunkt „Diverse" wiederfinden. Diese Funktionen werden bei der 80-Zeichen-Darstellung, die sicher die gebräuchlichste ist, überdeckt (siehe Bild #). Nun kann mit dem Schreiben begonnen werden. Soll ein schon bestehender Text geladen werden, erscheint eine komfortable Auswahlbox, die das gesamte Inhaltsverzeichnis einer Diskette anzeigt.

Die Voreinstellungen

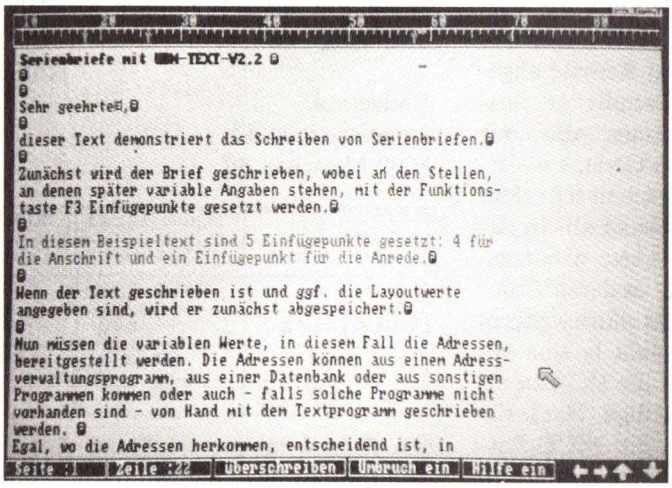
Bevor man beginnt, einen Text zu schreiben, sollte man noch einige Einstellungen vornehmen. Die erste betrifft die Umschaltung vom voreingestellten 60- in den 80-Zeichen-Modus. Danach wird die Position des linken und rechten Randes eingestellt. Im Menü „LAYOUT" wird nun die äußere Form des Textes bestimmt, die besonders für den späteren Ausdruck wichtig ist. Blattlänge und Position der Kopf- bzw. Fußzeile können verändert werden. Jetzt könnte man noch die Farben einstellen bzw. in den Monochrom-Modus gehen, dann ist wirklich alles eingestellt. Oder wollen Sie etwa eine ausländische Tastaturbelegung wählen, von denen es immerhin acht verschiedene gibt?

Die Bedienung

Die Handhabung von UBM-TEXT ist recht einfach. Allerdings ergeben sich auch einige Einschränkungen, die gerade die Bedienung und auch spezielle Funktionen betreffen. Zum einen ist das die träge Geschwindigkeit des Textsystems, die beim „normalen" Schreiben kaum auffällt (der Zeilenumbruch geht jedoch gemächlich vonstatten), richtig einschläfernd wird es freilich, wenn in einem größeren Absatz etwas gelöscht oder eingefügt werden soll. Aber nicht nur, daß das Ganze langsam geht, es gibt zudem einen nachlaufenden Tastaturpuffer, der auch dann noch munter löscht, wenn man die DEL-Taste schon lange nicht mehr gedrückt hält. Im Handumdrehen verschwinden dann, wie von Geisterhand, ganze



60-Zeichen-Bildschirm mit Menüleiste



80-Zeichen-Bildschirm

Satzstücke auf Nimmerwiedersehen. Es scheint, als hätten die Programmierer die Fähigkeiten des AMIGA und seiner schnellen CPU nicht recht ausgenutzt, wie dies bei einem Update zu erwarten gewesen wäre.

Recht gut ist insgesamt die Belegung der Funktionstasten und spezieller anderer Tasten oder Tastenkombinationen gelungen. Dies wurde hier ausgiebig verwendet, denn fast alle Funktionen lassen sich über solche Tasten anwählen. Doch spätestens, wenn man versucht, ein System in dieser Belegung zu finden, wird man einige Probleme haben. Man hätte sich dabei sicherlich ein Beispiel an WORDSTAR nehmen können oder gleich sinnvolle deutsche Abkürzungen verwenden sollen. Doch bei UBM-TEXT wird mit CTRL-, SHIFT- und anderen Sequenzen gearbeitet, ohne daß sich ein Zusammenhang zu dem gewünschten Befehl erkennen läßt; und auf die deutschen Bezeichnungen wurde keinerlei Rücksicht genommen, obwohl es sich hier doch ausdrücklich um ein deutsches Produkt handelt.

Die Features

UBM-TEXT kann wahlweise ASCII-Texte laden und speichern. Dadurch können z. B. Texte mit anderen Programmen ausgetauscht bzw. Dateien für andere Programme erstellt werden. Auch das Erstellen von Source-Codes ist denkbar. Eine Funktion, die viel Zeit spart, ist das Suchen/Ersetzen von Wörtern oder Texten. Nach Eingabe des zu suchenden und neuen Wortes kann gewählt werden, ob die Groß/Kleinschreibung beachtet und ob der Vorgang automatisch oder nur auf Anfrage ausgeführt werden soll. Texte können zentriert, im Blocksatz, links- oder rechtsbündig ausgegeben werden. Zeilen- und Seitenumbrüche werden automatisch vorgenommen, sie können jedoch auch von Hand ausgeführt werden. Hierbei fällt auf, daß es keine Trennhilfe gibt, wodurch am rechten Textrand bzw. im Text (bei Blocksatz) große Lücken auftreten können. Zumindest eine halbautomatische Trennhilfe, bei der z. B. bei einem Zeilenüberlauf die jeweilige Trennstelle mit der Maus angegeben wird, wäre sehr angebracht. Eine nützliche Sache sind die Blockoperationen. Nach Anwählen der Markierungsfunktion wird der Block mit dem Mauszeiger festgelegt. Allerdings kann das Blockende nicht außerhalb des sichtbaren Bereichs liegen, wodurch sich eine unnötige Einschränkung dieses Befehls ergibt. Der markierte Textteil kann nun gelöscht, abgespeichert oder kopiert werden. Es ist aber auch möglich, die Schriftart oder das Format des gesamten Blocks zu ändern. Sehr praktisch ist die Pufferung des Blocks, so daß versehentlich ausgeführte Funktionen rückgängig gemacht werden können.

Das Erstellen von Serienbriefen

Serienbriefe können entweder manuell oder automatisch erstellt werden. Der Text wird dazu an den entsprechenden Stellen mit Einfügemarken versehen, die dann beim Drucken durch Text ersetzt werden. Die manuelle Methode ist dabei recht umständlich und nur bei sehr wenigen Exemplaren geeignet. Die automatische Variante ist natürlich besser, denn die benötigten Eingaben werden aus einer Datei (z. B. Adreßdatei) geholt, der Ausdruck einer beliebigen Anzahl von Briefen geht dann ohne weiteres Zutun vonstatten.

Der Ausdruck

Gedruckt wird entweder über den parallelen oder den seriellen Port. Wahlweise kann entweder der ganze Text oder nur der markierte Block ausgedruckt werden. Außerdem kann eine spezielle Druckeranpassung einfach erstellt werden. Die Datei wird dazu eingeladen und dann entsprechend den Codes im Druckerhandbuch abgeändert. Beim Ausdruck fällt allerdings sofort auf, daß keine automatische Seitennumerierung vorgenommen wird. Hier muß wieder von Hand und auf jeder Seite ein spezieller Code eingefügt werden, an den dann die Seitennummer ausgegeben wird.

Nun...

UBM-TEXT ist ein Textverarbeitungsprogramm, mit dem eine normale Privatperson ihre Texte erstellen kann. Für den kommerziellen Einsatz, also z. B. in einem Büro, ist diese Textverarbeitung kaum geeignet. Dazu ist sie zu langsam und unflexibel, außerdem bietet sie zu wenig Komfort. Der Preis von 249 DM ist nach meiner Ansicht viel zu hoch angesetzt und für einen privaten Anwender kaum akzeptabel. Ein Blick auf den Atari ST und das dort eingesetzte Textverarbeitungsprogramms 1ST WORD bzw 1ST WORD PLUS zeigt, daß man für einen niedrigeren Preis weitaus bessere Produkte erstehen kann.

Preis: 249,– DM (im Umtausch gegen die alte Version: 20,– DM)

Konfiguration:
AMIGA 1000 (Kickstart 1.2, 512K RAM),
AMIGA 500, AMIGA 2000

Dokumentation: deutsch

- langsame Geschwindigkeit des gesamten Textsystems
- keine Fußnotenverwaltung
- keine Grafikeinbindung
- keine Silbentrennung
- keine Funktionstastenprogramierung
- unübersichtlichte Tastenbelegung
- hoher Preis

+ Funktionstastenbelegung
+ Serienbriefschreibung
+ individuelle Druckeranpassung
+ Phrasenspeicher (6x je 200 Zeichen)
+ kein Kopierschutz

Endlich: Der TEXTOMAT für den AMIGA

Darauf haben AMIGA-Anwender lange gewartet: Eine Textverarbeitung, die einfach zu bedienen ist (eigene Tastaturbelegung), etwas Komfort bietet (automatische Trennhilfe, Suchen/Ersetzen usw.) und die vor allem einen günstigen Preis hat. TEXTOMAT erfüllt diese Wünsche und bietet zudem noch die Möglichkeit, Grafiken in den Text einzubinden. Unser Test soll zeigen, was man für knapp hundert Mark erwarten kann.

Kinderleicht!

Die Bedienung von TEXTOMAT ist erstaunlich einfach und schnell zu erlernen. Ein ausführliches, leicht verständliches Handbuch hilft vor allem dem AMIGA-Anfänger, sich einzuarbeiten. Dabei werden auch die Grundbegriffe des AMIGA und einige wichtige Hilfsprogramme, z. B. die 'PREFERENCES' erklärt.

Nach dem Laden erscheint ein Übersichtlicher Bildschirm mit einer separaten Informations- und Tabulatorzeile. Die Informationszeile gibt die momentane Cursorposition, die Zeichendichte, den Zeilenabstand und die jeweiligen Modi an. Eine Zeile darunter stehen die Tabulatormarken, die mit der Maus einfach gesetzt und gelöscht werden können.

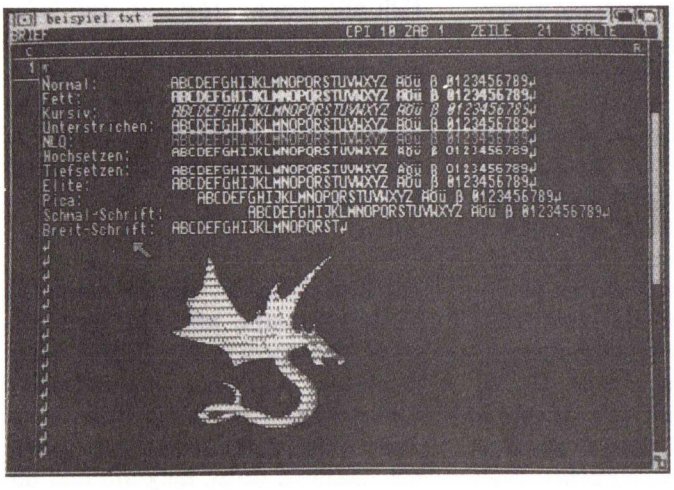
Automatisches Trennen

Der Bildschirm kann bis zu 27 Zeilen je 73 Anschläge darstellen. Allerdings können bis zu 999 Anschläge pro Zeile eingegeben werden, wobei dann der Bildschirm durchscrollt. Am Zeilenende wird automatisch ein Wordwrap vorgenommen: Ein Wort, das nicht mehr in die Zeile paßt, wird in die nächste übernommen. Damit dabei nicht so große Lücken entstehen, kann zusätzlich noch die automatische Trennung eingeschaltet werden. Dann wird versucht, das Wort zu trennen. Vor allem bei zusammengesetzten Wörtern können dabei Fehler auftreten, die sich jedoch leicht durch einen Trennvorschlag beheben lassen.

In der Grundeinstellung arbeitet TEXTOMAT im Überschreibemodus, sinnvoller ist aber sicherlich der Einfügemodus. Wenn damit innerhalb eines Textes eingefügt wird, dann verschiebt sich die Zeile in die folgende. Da hierbei keine Rücksicht auf Wordwrap und Trennen gelegt wird, geht dies recht schnell. Um danach die grammatikalische Ordnung wiederherzustellen, ist es notwendig, den Absatz neu zu formatieren. Dies ist jedoch mit einem Tastendruck ('Help') schnell erledigt.

Alles wird in Form gebracht

Die Texte werden schon während des Eingebens formatiert, wahlweise rechts-, linksbündig oder zentriert. Natürlich ist der gebräuchliche Blocksatz ebenso vorhanden wie die Möglichkeit, Texte einzurücken. Absatzweise können diese Formatierungen auch noch nachträglich vorgenommen werden. Leider gilt dies nicht für den ganzen Text oder einen markierten Block.



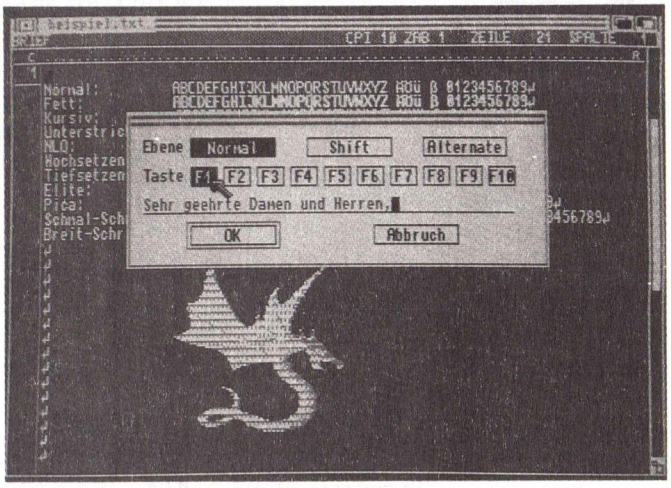
Textomat Darstellung der Schriftarten auf dem Bildschirm

Blocken

Ein sehr wichtiger Bestandteil einer Textverarbeitung sind die Blockoperationen. Sie sind hier sehr anwenderfreundlich gelöst, denn alles wird komfortabel mit der Maus erledigt. Blöcke werden einfach mit der Maus markiert, ohne eine andere Funktion aufrufen zu müssen. Dieser Block kann dann kopiert, verschoben oder gelöscht werden, wozu nur ein Menüaufruf oder ein Tastendruck notwendig ist.

Sequencen (Seite nach oben, Anfang nächster Absatz usw.) können ebenfalls auf die Funktionstasten gelegt werden. Durch die Tatsache, daß die Tastenbelegung mehrfach abspeicherbar ist, kann damit auch z. B. eine kleine Adreßdatei erstellt werden.

Die wichtigsten Einstellungen, dazu gehören neben der Funktionstastenbelegung noch Zeichensatz, Druckerparameter, Kopf- und Fußzeilen, Tabulatoren und einige Modi, kön-



Funktionstasten programmierung

Wer sucht, der findet...

Es geht aber auch einfacher, denn wofür hat man schließlich einen Computer, wenn er einem nicht die Arbeit abnimmt oder zumindest erleichtert. Suchen und Ersetzen ist eine Funktion, die dann interessant wird, wenn in einem längeren Textstück bestimmte Worte oder Ausdrücke durch andere ersetzt werden sollen. Dazu wird einfach der Such- und der Ersetzbegriff eingegeben und der Rest geht von alleine. Wenn man sich nicht ganz sicher ist, ob auch alles richtig gemacht wird, dann ist vor jedem Ersetzen eine Sicherheitsabfrage oder ein Abbruch möglich.

Alles auf Tastendruck

Komfortabel und nützlich ist die Funktionstastenbelegung. Jede Taste kann dreifach (normal, mit Shift, mit Alternate) mit bis zu 160 Zeichen belegt werden. Dies können z. B. bestimmte häufig vorkommende Anreden oder Redewendungen sein, aber auch sogenannte Tastaturmakros sind möglich, d. h. bestimmte Control-

nen als 'Options' abgespeichert werden. Diese Datei wird bei jedem Starten von TEXTOMAT mitgeladen, wodurch man sich viel Arbeit sparen kann.

Auch die wichtigsten Menüpunkte sind mit einer Tastenkombination aufrufbar. Dazu wird die 'AMIGA'-Taste zusammen mit dem Anfangsbuchstaben der Funktion gedrückt. Allerdings muß man dazu die englischen Begriffe wissen, denn die Kennungen sind von ihnen abgeleitet.

Besonderheiten

In TEXTOMAT sind einige Features eingebaut, die nicht jeder nutzen wird, die jedoch sehr interessant sein können. Das erste betrifft Uhrzeit und Datum: Auf Tastendruck werden die jeweils aktuellen Daten in den Text eingefügt und, das ist das Besondere dabei, sie werden bei jeder weiteren Bearbeitung aktualisiert. Wird also ein Brief geschrieben und erst einige Tage später ausgedruckt und weggeschickt, dann erhält er beim Ausdruck das aktuelle Datum.

Eine weitere Besonderheit ist der C-Source-Modus. Bei jeder geschweiften Klammer wird automatisch um drei Stellen ein- bzw. ausgerückt. Dadurch ist immer eine Kontrolle über die noch offenen Programmblöcke gegeben. Um als Editor genutzt werden zu können und um mit anderen Programmen Daten auszutauschen, gibt es die Möglichkeit, ASCII-Texte einzulesen und auszugeben.

Um Tabellen anzulegen, gibt es die Tabulatormarken. Sie erlauben ein linksbündiges Formatieren der eingegebenen Daten. Doch speziell für Zahlen ist ein rechtsbündiges Format erwünscht, oder noch besser die Einhaltung der dezimalen Stellen. Dafür ist ein spezieller Dezimaltabulator eingebaut, der die Dezimalpunkte sauber untereinander anordnet.

Ebenfalls erwähnenswert ist der vergrößerte Zeichensatz mit einer 12★12 Matrix (gegenüber 8★8), der für Benutzer eines Fernsehgeräts gedacht ist. Die Zeichen sind dann ausreichend groß, um sie auch auf einem über einen TV-Modulator angeschlossenen Fernseher gut erkennen zu können.

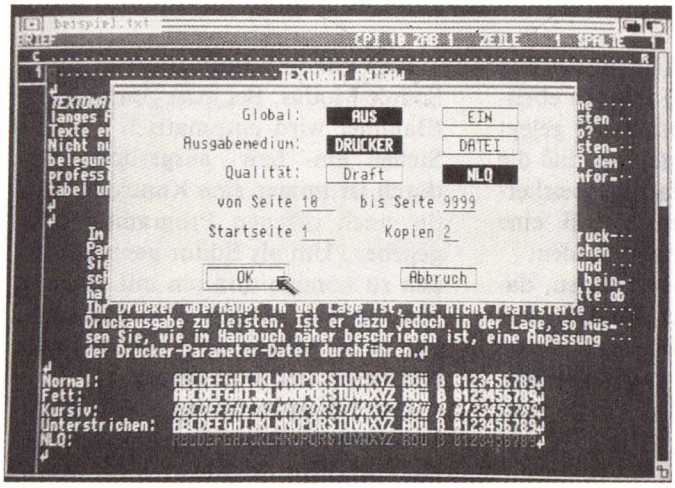
Jetzt wird es druckreif

Das Seitenlayout wird im Menü FORMULAR festgelegt. Dazu gehören Papierlänge und Zeilenabstand sowie die Position der Kopf- und Fußleiste, deren Text getrennt für links, mitte und rechts eingegeben wird. Die Schriftarten und -typen werden direkt beim Eingeben des Textes bestimmt. Gleiches gilt für unterschiedliche Zeichen- und Zeilenabstände.

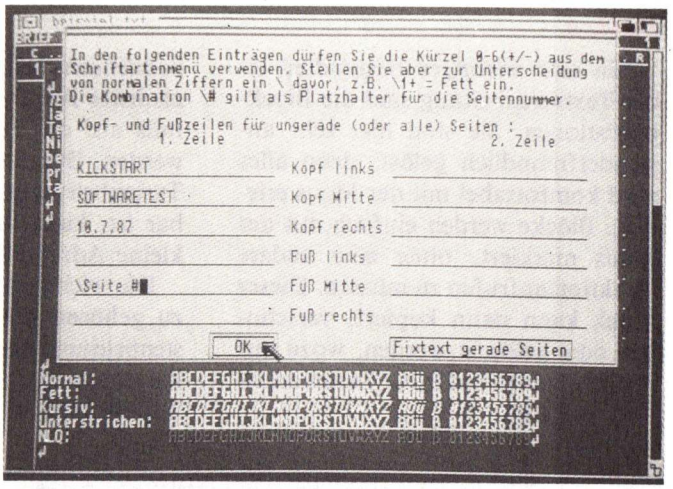
Printer ready?

Der Text kann direkt zu Papier gebracht werden, wahlweise im Schnell- oder Schönschriftdruck und als Teilbereich (von ... bis Zeile). Es ist aber auch möglich, mehrere Texte in eine Ausgabeliste einzutragen, in der jederzeit Änderungen und Löschungen vorgenommen werden können.

Die Druckeranpassung ist ebenfalls sehr flexibel, denn sie ist als ASCII-Datei abgelegt und dadurch leicht den eigenen Anforderungen anzupassen. Für die gängigen Druckermodelle ist sie bereits vorhanden, aber auch vor einer Änderung muß man keine



Druck-Menü



Eingabe der Kopf- und Fußzeilen

Angst haben, denn sie wird ausführlich im Handbuch erklärt. Auch den unerfahrenen Benutzer wird die Anpassung nicht allzu viel Zeit kosten.

Vermischtes: Grafik im Text

Wie das bei Tests so ist, steht das Interessanteste immer am Schluß. Dies dürfte bei TEXTOMAT die Einbindung von Grafik in den Text sein. Dabei treten gleich mehrere Einschränkungen auf, aus denen man den Schluß ziehen kann, daß es sich hier nur um eine einfache Möglichkeit bzw. um einen ersten Ansatz handelt, dieses Gebiet anzugehen.

Die erste Einschränkung ist, daß nur s/w-Bilder von Textomat verarbeitet werden können. Allerdings werden farbige Bilder, die im IFF-Format (z. B. Deluxe Paint) vorliegen, beim Einlesen automatisch konvertiert. Dabei können die einzelnen Farben in Graustufen umgesetzt werden; allerdings sind die dabei erzielten Ergebnisse dem Original meistens nicht sehr ähnlich. Wenn ein Bild eingeladen ist, dann kann es nur noch vertikal verschoben werden. Text läßt sich in den Zeilen, in denen sich das Bild befindet, werder eingeben noch editieren. Dazu muß das Bild erst wieder gelöscht werden.

Ein Foto bitte...

Nebenbei, es müssen nicht immer ganzseitige Bilder eingelesen werden, denn es wird ein spezielles Programm mitgeliefert, das es erlaubt, Bildausschnitte festzuhalten und abzuspeichern. Dieses Programm heißt BT-Snap und wird sinnigerweise durch ein Kamerasymbol dargestellt. Nach dem Starten bleibt dieses Programm im Speicher und kann durch <A> + <Help> von den meisten Programmen aus aktiviert werden. Die Bilder werden auf der RAM gespeichert und sind von dort einzulesen oder zu kopieren.

Ein Lob

TEXTOMAT ist eine durchaus brauchbare und benutzerfreundliche Textverarbeitung. Alle wichtigen Funktionen sind vorhanden und sehr komfortabel. Etwas unbefriedigend ist lediglich die Konvertierung von Farbgrafiken, denn die Qualität ist nicht besonders gut. Bei einem Preis von 99 Mark ist das Preis-/Leistungsverhältnis wirklich hervorragend. Keine derzeit auf dem AMIGA-Markt erhältliche Textverarbeitung kann Vergleichbares bieten. Man darf deshalb gespannt auf die 'Profi-Version' (BeckerTEXT) warten, die bald zu einem Preis von DM 199,– erhältlich sein wird und mit einer Vielzahl zusätzlicher Features aufwarten soll.

(mn)

+ automatische Silbentrennung
+ eigene Funktionstastenbelegung
+ Shortcuts
+ einfache Druckeranpassung
+ Installationsprogramm für Festplatte
+ kein Kopierschutz
+ günstiger Preis

– mäßige Konvertierung von Farbgrafiken in s/w-Bilder
– keine Serienbriefschreibung (Mail Merge)
– keine Fußnotenverwaltung

Anbieter: DATA BECKER

Preis: DM 99,–

Der Amiga setzt Grafikstandards.

Dieses Buch auch:

Es gibt viele Rechner, aber nur einen Amiga. Hier hält die Werbung wirklich das, was sie verspricht: „Unschlagbar in der Kombination Text und Grafik/Farbe." Eine Reihe Bücher widmet sich daher speziell diesem Thema; aber sicherlich keines so ausführlich wie Amiga Supergrafik. Denn hier finden Sie tatsächlich das gesamte Know-how zu Grafik und Animation auf dem Amiga – Wissen, das Sie für eigene, eindrucksstarke Programme nutzen können. Wie lassen sich die phantastischen Möglichkeiten dieser Grafikmaschine bis zum Letzten ausreizen? Wie erreicht man die Grenze des Machbaren? Dieses Buch verrät Ihnen die Antwort. Wer sich hier soweit vorwagt, wird in Dimensionen stoßen, die er selbst einem Amiga nicht zutrauen würde. Aber keine Sorge: Amiga Supergrafik wurde nicht nur für Profis geschrieben. Es bietet jedem etwas. So gibt es zahlreiche Einsteiger-Programme, die das nötige Grundwissen vermitteln, ebenso wie Programme für den fortgeschrittenen BASIC-Programmierer. Profis hingegen erfahren, wie sie die Grafik von C aus ansprechen können. Kurzum ein Buch, das Ihnen genau das Know-how vermittelt, das Sie brauchen: Grafikprogrammierung mit den vorhandenen BASIC-Befehlen, Nutzung der Libraries, die Register der Grafik-Chips, Aufbau und Programmierung von Screens, Windows, HAM, Halfbrites und Interlaces aus BASIC und C, 1024 x 1024 Punkte Superbitmap, gepufferte Multitasking-Hardcopy-Routine – zum Thema Grafik werden Sie in Amiga Supergrafik nichts vermissen.

Amiga Supergrafik
Hardcover, 686 Seiten, DM 59,–

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

BESTELL-COUPON
Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:

☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

Name

Straße

Ort

P I N B

Autoren gesucht

Die KICKSTART-Redaktion sucht Programmierer, die sich mit dem AMIGA auskennen und darüber zu berichten wissen, sei es über Soft- oder Hardware.

Wenn Sie eigene Programme geschrieben haben, die Sie gerne veröffentlichen würden, so lassen Sie sie nicht in der Schublade verschwinden, sondern schicken sie uns!

Neben Programmlistings suchen wir auch Tips & Tricks über den AMIGA. Angefangen bei Assembler bis hin zu Basic, C, Pascal oder anderen Sprachen. Auch die Bastler sind aufgerufen, über ihre Fachgebiete (z.B. Floppies, Drucker, Harddisk, Monitor) zu berichten. Das Honorar erfolgt nach Vereinbarung.

Über Ihre Beteiligung würden wir uns sehr freuen.

KICKSTART-Redaktion
'Programmeinsendung'
Industriestr. 26
6236 Eschborn
Tel. 06196/41245

Comiczeichner/innen gesucht

Mögen auch Sie etwas Auflockerndes und Heiteres in Computerzeitschriften?

Wir möchten in Zukunft regelmäßig Cartoons und kleine Comicstrips veröffentlichen und suchen dazu noch begabte Zeichner. Wenn Sie eigene Ideen haben, um eine Serie zu gestalten, diese Ideen in Bilder umsetzen können und andere Leute damit auch zum Lachen bringen, dann setzen Sie sich bitte augenblicklich mit uns in Verbindung und schicken Sie einige Probezeichnungen!

Nach Möglichkeit sollten die Comics eine oder mehrere immer wiederkehrende Personen zum Inhalt haben und computerbezogen (nicht unbedingt AMIGA) sein.

Nicht nur Comics sind denkbar, auch sonstiges in Wort und Bild – seien es Karikaturen, Kuriositäten, Gedichte, lustige Erfahrungen aus dem AMIGA-Leben, humorvolle Berichte bis hin zu geballtem Unfug. Auch hier soll natürlich der Bezug zu Computern bestehen.

Bei Interesse wenden Sie sich bitte an die Kickstart-Redaktion, Telefonnummer nicht vergessen!

KICKSTART-Redaktion
'Comic'
Industriestr. 26
6236 Eschborn
Tel. 06196/41245

BOARD

Public Domain gesucht

Für viele ist Public-Domain bereits ein Begriff. Es sind Programme, die von den Autoren freigegeben wurden und somit frei kopiert werden dürfen. Damit stehen sie jedem Anwender zur Verfügung.

KICKSTART hat sich entschlossen, eine eigene Public-Domain-Sammlung zu schaffen. Darin werden neben einer Auswahl an ausgewählter, schon existierender Software auch eigene Programme enthalten sein. Wir möchten mit dieser Sammlung gerade für Neulinge der AMIGA-Szene die Möglichkeit schaffen, gute Software für ihren Rechner zu bekommen, ohne tief in den Geldbeutel greifen zu müssen.

Schon jetzt hat sich gezeigt, daß die Qualität der privaten Programmierkunst an manches professionelle Produkt heranreicht und sich davor nicht verstecken muß. Zwar ist meist nicht die Leistung einer Softwarefirma zu erreichen, doch bestätigen Ausnahmen bekanntlich die Regel.

Nicht nur große Programme sind dabei gefragt; auch Utilities, die im Einzelfall von unschätzbarem Wert sein können, sind wichtige Bestandteile der Public-Domain. Auch ausgefeilte Routinen und Funktionsbibliotheken können für andere Programmierer sehr wichtig sein. Insgesamt sind dem Einfallsreichtum keine Grenzen gesetzt.

Was bei anderen Rechnersystemen funktioniert, sollte beim AMIGA schon lange funktionieren – was sind schon 500 MS-DOS Disketten oder die ST-Sammlung! Deshalb der Aufruf an alle Programmierer, die sich des Rechners mächtig fühlen: Tun Sie Ihren Teil dazu, um die AMIGA-Public-Domain Szene anzukurbeln und sie in der Computerwelt zu einem Begriff zu machen.

Wenn Sie sich an der KICKSTART-Sammlung beteiligen und Ihre Programme gerne der Allgemeinheit zur Verfügung stellen möchten, so schikken Sie uns das/die Programm/e einfach auf Diskette zu. Bitte bestätigen Sie, daß das Programm von Ihnen geschrieben wurde und daran keinerlei kommerzielle Rechte bestehen.

Wir halten die Public-Domain Software für eine sehr nützliche Kreation und möchten unseren Teil zu ihrer Verbreitung tun. Die Sammlung wird ab sofort zum Selbstkostenpreis über die Redaktion zu erhalten sein.

KICKSTART Redaktion
'PD-Einsendung'
Industriestr. 26
6236 Eschborn
Tel. 06196/41245

AMIGA-Clubs bitte melden

Ein Aufruf an alle AMIGA Clubs und die USER-Vereinigungen

Wir möchten allen Amiga-Usern die Möglichkeit geben, sich an Clubs anzuschließen. Dort finden Sie aller Voraussicht nach auch andere Anhänger des AMIGA, die Sie verstehen und gleiche Interessen haben.

Doch dazu benötigt man die Adressen, die wohl äußerst selten im Telefonbuch stehen. Deshalb rufen wir alle AMIGA-Clubs auf, uns ihre Kontaktadresse zu nennen. Wir werden sie in der KICKSTART veröffentlichen und hoffen, damit den Clubs die Chance zur Repräsentation geben zu können. Bitte geben Sie eine kurze Information über die Tätigkeiten der Clubs und ihre Besonderheiten (z.B. Größe, Interessen, Aktivitäten).

Bitte schicken Sie die Info an:

KICKSTART-Redaktion
'Clubinfo'
Industriestr. 26
6236 Eschborn
Tel. 06196/41245

Komfortabel: GO AMIGA DATEI

GO AMIGA DATEI ist eine einfach zu bedienende Dateiverwaltung. Alle Funktionen sind menü- oder mausgesteuert; das Handbuch kann also rasch beiseite gelegt werden. Als Besonderheit bietet das Programm Grafik- und Tonverwaltung sowie die Möglichkeit zum Erstellen einer vertonten Diashow.

Aufbau einer Datei

Um zu zeigen, wie einfach das Erstellen einer Adreßdatei ist, wollen wir zunächst die einzelnen Schritte kommentieren: Nach Aufruf von „Neu erstellen“ erscheint ein neues Dateiblatt, auf dem die Datei aufgebaut wird (siehe Bild 1). Anschließend wird „Feld erstellen“ angewählt, sofort erscheint ein Requester, der die Eingabe des Namens und des Feldtyps (Text, Zahl, Telefon, Ton, Bild usw.) verlangt. Nach dem Quittieren mit 'Return' erscheint das Dateiblatt mit der neu erstellten Spalte. Diese Spalte ist acht Zeichen breit, ihre Größe läßt sich jedoch durch einfaches 'Ziehen' mit der Maus variieren. Auch die Anordnung der Spalten kann zu jedem Zeitpunkt durch Verschieben mit der Maus verändert werden.

Wenn die sogenannte 'Dateimaske' erstellt ist, kann mit der Dateneingabe begonnen werden. Ein kurzer 'Klick' mit der Maus in eine leere Zeile, und schon ist man im 'Formularmodus', in dem alle Eingaben und Änderungen vorgenommen werden (siehe Bild 2). Jedes Eingabefeld besteht aus einem Namen und einem Eingabebereich, dessen Größe mit der Maus verändert werden kann. Auch die einzelnen Felder können noch jederzeit vertauscht werden, wobei die Eingabe immer beim oberen Feld anfängt. Wenn alle Daten eingeben sind, wird in den 'Listenmodus' zurückgekehrt. Nun sieht man einen bestimmten Ausschnitt der Gesamtdatei vor sich. Wenn die Spalten und die Zeilen nicht mehr auf den Bildschirm passen, wird mit den 'Scrollbalken' der sichtbare Bereich verschoben und die verdeckten Daten werden sichtbar.

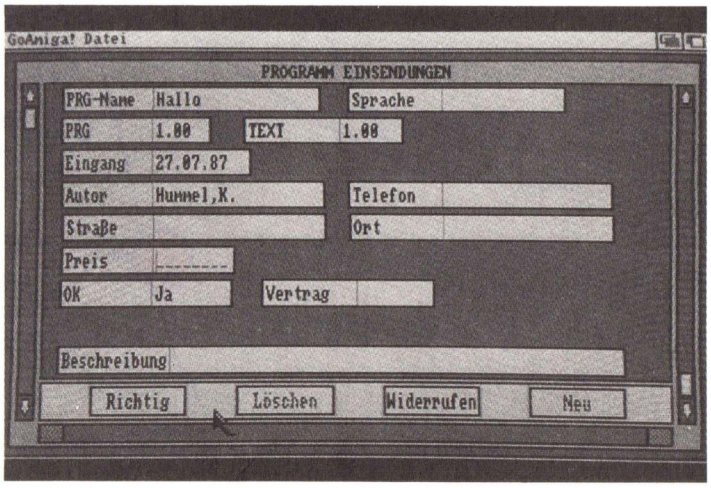
Die Datei ist nun erstellt. Es können aber immer noch Änderungen vorgenommen werden; die einfachsten Änderungen betreffen die Reihenfolge der Spalten und die Größe der Ein- bzw. Ausgabefelder.

Ändern und Löschen in einer Datei

Obwohl Änderungen eigentlich die Ausnahme sein sollten, können sie sowohl an den Daten als auch an der Struktur der Datei vorgenommen werden. Das Gleiche gilt auch für Löschungen.

Berechnen von Feldern

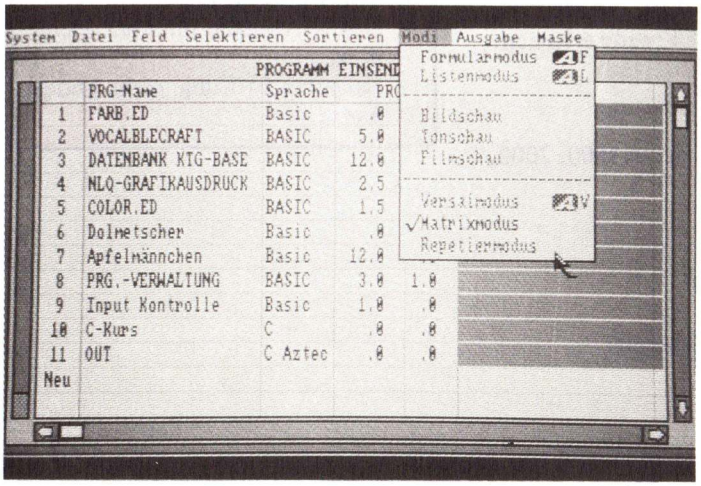
Bisweilen muß der Inhalt eines Feldes aus anderen Feldern errechnet



Eingeben der Daten im 'Formularmodus'

werden. Dazu wird eine ganz normale Rechnung mit den Spaltennamen aufgestellt (z.B. summe = beitrag + spende). Als Operationen stehen +, −, ★ und / zur Verfügung.

nach einem der zur Verfügung stehenden Kriterien (=, <>, <, >, <=, >=, ★) untersucht wird. Somit können dann z. B. alle Adressen mit der Postleitzahl



Datei bearbeiten im 'Listmodus'

Sortieren (Einzel- und Mehrfachgliederung)

Um eine Datei einfach, also nur nach einem einzigen Kriterium (z. B. Namen) zu sortieren, wird diese Spalte aktiviert und danach der Menüpunkt 'Aufsteigend' oder 'Absteigend' angewählt. Natürlich kann es vorkommen, daß man nach diesem Sortiergang feststellt, daß es mehrere Personen mit dem Namen MEIER in der Datei gibt. Um diese Gruppen zusätzlich noch nach den Vornamen zu sortieren, wird die Option 'Mehrfachgliederung' angewählt, die Spalte 'Vornamen' aktiviert und nochmals aufsteigend sortiert.

Selektieren

Wenn eine Datei sehr umfangreich ist, wird sie träge und unübersichtlich. Deshalb ist es oft sinnvoll, einen bestimmten Teilbereich herauszufiltern, der dann getrennt angezeigt wird. Dabei gibt es zwei Typen:

1) Selektieren nach Bereich: Nachdem eine Spalte aktiviert wurde, muß in zwei Eingabefeldern ein Von- und Bis-Wert eingegeben werden, nach dem die Daten herausgesucht werden.
2) Selektieren nach Kriterium: Auch hier muß erst eine Spalte aktiviert werden, bevor dieser Menüpunkt angewählt werden kann. Danach wird ein Wert eingegeben, der

6000 in die Selektionsdatei übertragen werden.

Ausdrucken

Natürlich ist das Ausdrucken von Dateien kein Problem, es gibt aber noch einige Punkte, die es komfortabler machen. So ist es möglich, beim Ausdruck bestimmte Spalten, deren Inhalt nicht jeder sehen soll, zu verstecken. Eine solche Ausgabeliste, in der nur bestimmte Daten enthalten sind, nennt man auch Report. Zudem kann bei der Ausgabe der Seitenaufbau bestimmt werden, also z. B. Abstand zu den Rändern, Seitenlänge, Schrifttyp.

Maskenverwaltung

Die komfortable Maskenverwaltung von GO AMIGA DATEI erlaubt es, eine Datei den verschiedensten Bedürfnissen des Benutzers anzupassen. So kann z. B. aus einer kompletten Adreßdatei eine Telefondatei generiert werden, in der nur Name und Telefonnummer stehen, oder eine Datei, in der sich lediglich Name und Wohnort finden. Für eine Datei können beliebig viele Masken erstellt werden, die alle auf diese Datei zugreifen.

Grafik und Ton

Auch wenn die Verwaltung von Bildern und Tönen sicher nicht allzu häufig benutzt wird, können damit sehr schöne Shows erstellt werden. Geladen werden können Bilder, die im IFF-Format (z. B. Deluxe Paint, Aegis Images) vorliegen oder die digitalisiert wurden (z. B. DigiView). Auch für Töne gilt, daß sie im IFF-Format vorliegen müssen (z. B. Future Sound, Perfect Sound). Grafiken und Bilder werden nicht ständig im RAM gehalten, nur ihre Namen werden verwaltet. Bei einem Aufruf werden sie dann von Diskette nachgeladen. Ein besonderes Feature von GO AMIGA DATEI ist eine vertonte Diashow, die sich sehr leicht erstellen läßt. Dies könnte für Werbedemos oder Lernzwecke nützlich sein; man kann damit aber auch einfach eine Bildershow mit Musik unterlegen.

Informationen & Voreinstellungen

GO AMIGA DATEI bietet drei Menüpunkte an, die Informationen über den aktuellen Stand des Systems anzeigen. 'STATUS' zeigt die Anzahl der Datensätze an, die sich in der Datei befinden, aber auch die Datensatzkapazität des Speichers und der Diskette.

Das nächste wichtige Feld ist 'DATUM/ZEIT'. Hier sollte eigentlich das Systemdatum erscheinen, allerdings hat nicht jeder eine eingebaute Uhr; deshalb können hier diese Einstellungen vorgenommen werden. Dieser Vorgang ist deshalb recht wichtig, weil die Datei mit diesem Datum abgespeichert wird.

Vor dem Erstellen einer Datei sollten einige weitere wichtige Einstellungen vorgenommen werden. Sie betreffen in erster Linie das Zahlenformat (Anzeigeformat, Nachkommastellen, Währung), das Datumsformat und die Standardwerte (RAM-Type, prozentuale RAM-Nutzung, Matrixmodus, Repetiermodus, Versalmodus, Pfadname für Datei/Ton/Bild).

Das Einstellen aller Werte ist sehr komfortabel, denn es geschieht ausschließlich mit der Maus. Mancher Anwender mag dies vielleicht etwas zu verspielt finden, doch diese Methode hat eigentlich nur Vorteile: Eine falsche Eingabe etwa ist gar nicht erst möglich.

Datenformat

Um mit anderen Programmen zusammenarbeiten zu können, werden ASCII-Dateien abgespeichert. Zusammen mit einem Textverarbeitungsprogramm ist dann ein komfortables Mail Merge möglich, denn die Ausgabedatei kann beliebig selektierte Teilmengen der Gesamtdatei enthalten. Man kann also zum Beispiel alle Vereinsmitglieder anschreiben, bei denen im Beitragsfeld noch kein 'JA' steht, ein Rundschreiben an alle Kunden im Raum '6000' aussenden und anderes mehr.

Fazit:

GO AMIGA DATEI ist eine sehr leicht zu bedienende Dateiverwaltung. Alle Eingaben und Einstellungen werden mit Menüs, Dialogboxen (Requestern) und der Maus vorgenommen. Die Einarbeitungszeit ist dadurch extrem kurz; selbst Anwender, die noch keine Erfahrung mit einer Dateiverwaltung haben, werden sie sehr schnell und mit Freude erlernen.

Konfiguration:

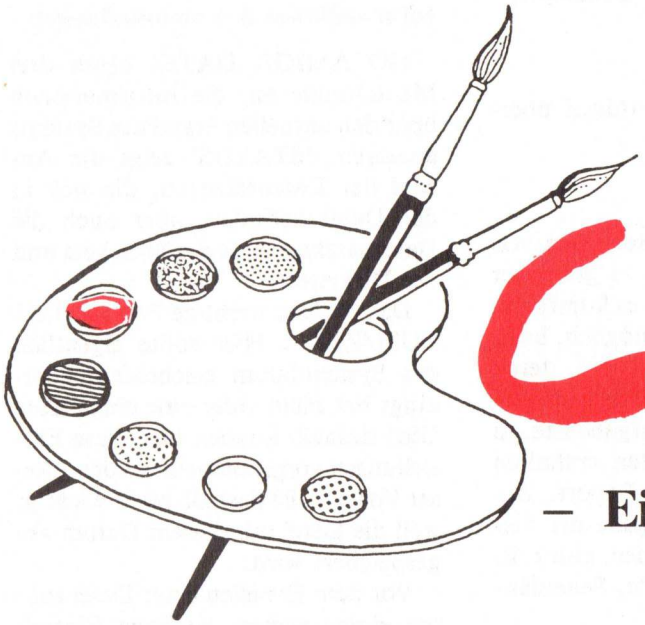
Läuft auf jedem AMIGA (500, 2000, 1000).

Anbieter:

SOFTWARELAND
Franklinstraße 27
CH-8050 Zürich (Schweiz)

Preis: 199 DM.

+ Steuerung mit Tastensequenzen
+ einfache Bedienung mit Maus und Menüs
+ deutsche Bezeichnungen/Anleitung
+ kein Kopierschutz
+ ASCII-Ausgabe für Mail Merge u. a.
– Zeilenbeschriftung nicht änderbar (1..)



COLOR.ED – Ein Editor für Farben und Muster

4096 Farbkombinationen bietet der Amiga. Daraus die gewünschte zu finden, ist nicht immer ganz einfach. Das Programm COLOR.ED hilft Ihnen dabei. Sie können mit der Maus die Zahlenwerte ermitteln, die Sie für den PALETTE-Befehl benötigen.

Wenn Sie das Programm gestartet haben, sehen Sie oben links drei Kästchen, in denen die Rot-, Grün- und Blauanteile der Farben enthalten sind. Darunter befinden sich 9 Farbfelder. Sie können nun eines dieser Farbfelder anklicken und sehen, wie sich in den oberen drei Kästchen die Rot-, Grün- und Blauanteile verändern. Das momentan angezeigte Farbfeld wird umrahmt. Zum verändern der Farbe fahren Sie mit dem Mauszeiger in eines der drei Kästchen und drücken die linke Maustaste. Fahren Sie nun mit gedrückter Maustaste nach links, wenn Sie von dieser Farbe weniger Anteile und nach rechts, wenn Sie mehr Anteile wünschen. Gleichzeitig verändern sich neben dem Kästchen die Zahlenwerte. Es sind dies die Zahlen, die mit dem PALETTE-Befehl eingegeben werden.

In der Mitte des Bildschirm sehen Sie eine durchgezogene Linie und 16 Kästchen. Daneben steht die Hexadezimale Zahl &HFFFF. Es ist dies der Wert für eine durchgezogene Linie. Ein Linienmuster wird mittels einer 16-Bit Maske definiert. Ein gesetztes Blatt in dieser Maske zeichnet die Linie hier in der aktuellen Vordergrundfarbe und ein nicht gesetztes Bit in der aktuellen Hintergrundfarbe. Diese Werte werden mit dem Befehl COLOR eingestellt. Es kann somit ein neuer Hintergrundfarbwert eingestellt werden, ohne das die sichtbare Hintergrundfarbe sich verändert. Damit lassen sich hübsche Effekte erzielen. Zum löschen eines Bits in dieser 16-Bit Maske fahren Sie mit dem Mauszeiger auf eines der 16 Kästchen und klicken es mit der linken Maustaste an. Das Feld nimmt nun die Hintergrundfarbe an und gleichzeitig verändert sich die Linie. Durch nochmaliges Anklicken wird dieses Bitfeld wieder gesetzt. Die

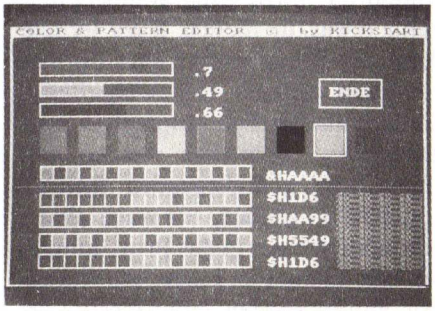
Zahl neben den 16 Kästchen stellt den Wert dar, den Sie mit dem PATTERN-Befehl in Ihren eigenen Programmen übernehmen.

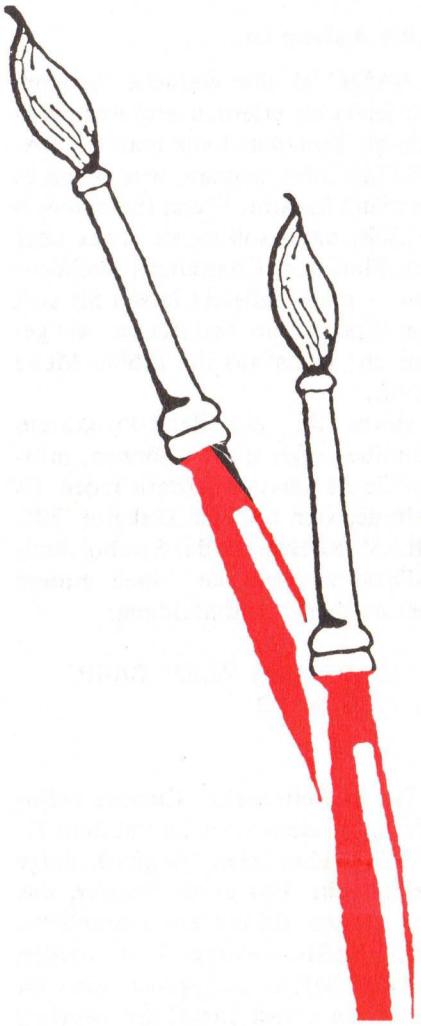
Die Erstellung eines Flächenmusters erfolgt mittels der vier 16-teiligen Felder am unteren Bildschirm. Er kann hiermit ein 4-Pixel hohes Flächenmuster definiert werden. Das löschen und setzen eines Bits erfolgt mit der Maus, wie bei dem Linienmuster beschrieben. Die Fläche erscheint unten rechts am Bildschirm in Originalgröße. Beenden Sie das Programm, indem Sie das ENDE-Feld anklicken.

Wie Sie den PALETTE und PATTERN Befehl benutzen, entnehmen Sie bitte dem Handbuch. Und nun viel Spaß beim Farbenmischen, Linien- und Flächenmuster erstellen.

Herbert Kunz

```
REM Programm: COLOR.ED
REM H. Kunz 1987
REM
REM Amiga 512 K / 80 Zeichen
REM

REM Anfangswerte einstellen

OPTION BASE 1
SCREEN 1,320,200,4,1
WINDOW 2,"COLOR & PATTERN EDITOR  "+CHR$(169)+"  by KICKSTART",,16,1
DIM linienwert(16),musterwert(4,16),flaeche%(4),muster%(4)
DIM farbwert(3,9)
FOR i=1 TO 16:linienwert(i)=1:NEXT
FOR j=1 TO 4
  FOR i=1 TO 16:musterwert(j,i)=1:NEXT
NEXT

FOR i=1 TO 4:muster%(i)=&HFFFF:flaeche%(i)=&HFFFF:NEXT
linie=65535&
y1(1)=21:y2(1)=29
y1(2)=37:y2(2)=45
y1(3)=53:y2(3)=61

PALETTE 4,1,0,0
PALETTE 5,0,1,0
PALETTE 6,0,0,1
FOR i= 1 TO 9:RANDOMIZE TIMER
  farbwert(1,i)=INT(RND (TIMER)*100)/100
  farbwert(2,i)=INT(RND (TIMER)*100)/100
  farbwert(3,i)=INT(RND (TIMER)*100)/100
  PALETTE i+6,farbwert(1,i),farbwert(2,i),farbwert(3,i)
NEXT

REM Bildschirm aufbauen

LINE (19,20)-(121,30),1,b
LINE (19,36)-(121,46),1,b
LINE (19,52)-(121,62),1,b
xpos=21:GOSUB changenummer

nummer=7
FOR i=20 TO 260 STEP 30
  LINE (i,70)-(i+20,90),nummer,bf
  nummer=nummer+1
NEXT
nummer=1

LINE (232,33)-(280,55),1,b
LOCATE 6,31:PRINT "ENDE"

LINE (18,68)-(42,92),1,b
FOR i=20 TO 170 STEP 10
  LINE (i,100)-(i+10,110),1,b
  LINE (i+1,101)-(i+9,109),3,bf
NEXT
LOCATE 14,25:PRINT "&H"HEX$(linie)
LINE (0,115)-(320,115),3


FOR j= 120 TO 165 STEP 15
  FOR i=20 TO 170 STEP 10
    LINE (i,j)-(i+10,j+10),1,b
    LINE (i+1,j+1)-(i+9,j+9),3,bf
  NEXT
NEXT

x=16
FOR i=1 TO 4
  LOCATE x,25:PRINT "&H"HEX$(muster%(i)):x=x+2
NEXT
PATTERN -1,muster%
LINE (245,120)-(315,175),3,bf

REM Mausabfrage

checkmouse:
taste=MOUSE(0):IF taste >=0 THEN checkmouse
xpos=MOUSE(1):ypos=MOUSE(2)

IF xpos >10 AND xpos <130 THEN
    IF ypos >20 AND ypos <30 THEN
      farbe=1:GOSUB changecolor:GOTO checkmouse
    END IF

    IF ypos >36 AND ypos <46 THEN
      farbe=2:GOSUB changecolor:GOTO checkmouse
    END IF

    IF ypos >52 AND ypos <62 THEN
      farbe=3:GOSUB changecolor:GOTO checkmouse
    END IF
END IF

IF xpos >20 AND xpos <180 THEN
    IF ypos >100 AND ypos <110 THEN
      GOSUB changelinie:GOTO checkmouse
    END IF

    IF ypos >120 AND ypos <175 THEN
      GOSUB changemuster:GOTO checkmouse
    END IF
END IF

IF ypos>=70 AND ypos<=90 THEN
    GOSUB changenummer
END IF

IF xpos>=232 AND xpos<=280 THEN
  IF ypos >=33 AND ypos <=55 THEN ende
END IF

GOTO checkmouse

ende:
  CLS:WINDOW CLOSE 2:WINDOW 1:END

changecolor:
  taste=MOUSE(0)
  IF taste >=0 THEN RETURN
  xpos=MOUSE(1)
  IF xpos>120 THEN xpos=120
  IF xpos<20  THEN xpos=20
  IF xpos<>20 THEN
    LINE (20,y1(farbe))-(xpos,y2(farbe)),farbe+3,bf
  END IF
  IF xpos<>120 THEN
    LINE (xpos,y1(farbe))-(120,y2(farbe)),0,bf
  END IF
  farbwert(farbe,nummer)=(xpos-20)/100
  PALETTE nummer+6,farbwert(1,nummer),farbwert
  (2,nummer),farbwert(3,nummer)
  LOCATE farbe*2+2,17:PRINT "    "
  LOCATE farbe*2+2,17:PRINT farbwert(farbe,nummer)
GOTO changecolor

changelinie:
  xpos=INT(xpos/10)*10-20:i=xpos/10+1
  w=2^(16-i):linienwert(i)=1-linienwert(i)
  IF linienwert(i)=1 THEN
    linie=linie+w
    LINE (xpos+21,101)-(xpos+29,109),3,bf
  END IF
  IF linienwert(i)=0 THEN
    linie=linie-w
    LINE (xpos+21,101)-(xpos+29,109),0,bf
  END IF
  LOCATE 14,25:PRINT "        "
  LOCATE 14,25:PRINT "&H"HEX$(linie)
  PATTERN linie
  LINE (0,115)-(320,115),3
  PATTERN 65535&
FOR k= 0 TO 300:NEXT  ' warteschleife
RETURN

changemuster:
  IF ypos>=120 THEN j=1:yy=120
  IF ypos>=133 THEN j=2:yy=135
  IF ypos>=148 THEN j=3:yy=150
  IF ypos>=163 THEN j=4:yy=165
  xpos=INT(xpos/10)*10-20:i=xpos/10+1
  w=2^(16-i):musterwert(j,i)=1-musterwert(j,i)
  IF musterwert(j,i)=1 THEN
    muster%(j)=muster%(j)+VAL("&h"+HEX$(w))
    LINE (xpos+21,yy+1)-(xpos+29,yy+9),3,bf
  END IF
  IF musterwert(j,i)=0 THEN
    muster%(j)=muster%(j)-VAL("&h"+HEX$(w))
    LINE (xpos+21,yy+1)-(xpos+29,yy+9),0,bf
  END IF
  PATTERN -1,muster%
  LINE (245,120)-(315,175),3,bf
  PATTERN -1,flaeche%
  LOCATE 16+(j-1)*2,25:PRINT "      "
  LOCATE 16+(j-1)*2,25:PRINT "$H"HEX$(muster%(j))
  FOR k=0 TO 300:NEXT  ' warteschleife
RETURN

changenummer:
  f1=(nummer-1)*30+20
  f=0:gefunden=0
  FOR i=20 TO 260  STEP 30
    f=f+1
    IF xpos >=i AND xpos <=i+20 THEN
      LINE (f1-2,68)-(f1+22,92),0,b
      LINE (i-2,68)-(i+22,92),1,b
      FOR j=1 TO 3
        LOCATE j*2+2,17:PRINT "    "
        LOCATE j*2+2,17:PRINT farbwert(j,f)
        IF farbwert(j,f)=0 THEN skip1
        LINE (20,y1(j))-(farbwert(j,f)*100+20,y2(j)),j+3,bf
        skip1:IF farbwert(j,f)=1 THEN skip2
        LINE (farbwert(j,f)*100+20,y1(j))-(120,y2(j)),0,bf
      skip2:NEXT
      gefunden =1:i=260
    END IF
  NEXT
  IF gefunden=1 THEN nummer=f
RETURN
```





ANFÄNGER AUFGEPASST:

Der AMIGA-BASIC-KURS beginnt

Aber auch derjenige, der schon einmal mit Basic programmiert hat, kann etwas in diesem Kurs lernen, denn der Schwerpunkt liegt bei den speziellen Befehle von AmigaBasic, die hier ausführlich besprochen werden sollen. Trotzdem werden erst einmal die grundlegenden Befehle von BASIC erklärt, um auch einem echten Einsteiger in das Programmieren eine Chance zu geben, die folgenden, interessanteren Teilen dieses Kurses zu verstehen.

Aller Anfang ist...

BASIC ist eine einfache Sprache, die leicht zu erlernen und anzuwenden ist. Trotzdem kann man mit BASIC fast alles machen, was einem in den Sinn kommt. Wenn Ihnen nichts einfällt, dann sollten sie etwas über den Sinn von Computern nachdenken – oder vielleicht lassen Sie sich von diesem Kurs inspirieren, wir geben uns jedenfalls die größte Mühe damit.

Bevor Sie ein Basic-Programm schreiben oder starten können, müssen Sie erst das AmigaBasic laden. Es befindet sich auf der Diskette 'EXTRAS'. Klicken Sie das Symbol AmigaBasic zweimal an. Nach einiger Zeit erscheint eine Meldung:

```
Commodore Amiga BASIC
Version 1.2
...
```

Die Schreibmarke (Cursor) befindet sich in einem Fenster mit dem Titel LIST. Hier geben Sie gleich einige Befehle ein. Das große Fenster, das den ganzen Bildschirm umschließt, heißt BASIC-Fenster. Hier werden spezielle Befehle eingegeben, die vom Computer direkt ausgeführt werden. Zwischen den beiden Fenstern können Sie leicht und schnell mit der Maus umschalten, indem Sie im betreffenden Fenster einmal die linke Maustaste drücken. Wenn Sie sich jetzt im BASIC-Fenster befinden, dann wird alles, was Sie hier eingeben, sofort erledigt, sofern es sich nicht um eine falsche Eingabe handelt.

Fangen wir also an: Die effektvollste Anweisung in BASIC ist der Befehl PRINT, denn dabei wird direkt auf dem Bildschirm etwas sichtbar. PRINT ist, wie alle Basic-Begriffe, aus dem Englischen abgeleitet und bedeutet drucken bzw. etwas ausgeben. Unser erster Versuch lautet deshalb folgendermaßen:

```
PRINT 1
```

Dabei ist es egal, ob Sie das Wort PRINT mit großen oder kleinen Buchstaben schreiben. Wenn Sie diese Zeile im BASIC-Fenster eingegeben haben und dann die Taste <RETURN> drücken, dann wird in einer neuen Zeile eine '1' ausgegeben. Haben Sie diese Zeile aber im LIST-Fenster eingegeben, dann passiert nicht viel, wenn die <RETURN>-Taste gedrückt wird, außer daß das Wort PRINT auf Großbuchstaben umgestellt wird. Wenn dieses Programm (tatsächlich ist diese eine Zeile schon ein Programm!) gestartet werden soll, dann müssen Sie mit der Maus im Menü 'RUN' das Wort 'Start' auswählen. Augenblicklich wird das BASIC-Fenster gelöscht und in der oberen linken Ecke eine '1' ausgegeben. Nun können Sie natürlich auch andere Zahlen ausgeben lassen. Dazu müssen Sie entweder im Basic-Fenster erneut PRINT und ein Zahl eingeben, oder Sie ändern im List-Fenster die Zahl hinter dem PRINT-Befehl. Sicher ist nun schon der Wunsch aufgekommen, auch einmal ein Wort oder einen Satz ausgeben zu lassen. Nichts leichter als das:

```
PRINT "hello world"
```

Wenn Texte ausgegeben werden sollen, dann müssen sie mit Anführungszeichen gekennzeichnet werden. Wenn Sie nun

```
PRINT "Harald"
```

eingeben, dann müßte klar sein, was auf dem Bildschirm erscheint, dagegen kann die Zeile

```
PRINT Harald
```

schon einige Probleme aufwerfen, denn die Anführungszeichen fehlen hier. Trotzdem erkennt der Computer diese Zeile als richtig an und gibt etwas auf dem Bildschirm aus. In diesem Fall ist es eine '0', was nun nicht heißt, daß Harald eine Null ist, denn eigenständig denken kann der Computer nicht. Es bedeutet vielmehr, daß die Variable Harald einen Wert von Null hat. Was ist eigentlich eine Variable? Eine Variable ist ein Platzhalter für etwas, das hier nicht näher festgelegt ist und deshalb noch '0' ist. Um einer Variablen einen Wert zu geben, gibt es die Anweisung LET, die so aussieht:

```
LET harald = 5
```

Nun passiert nicht viel, erst wenn zusätzlich noch eine PRINT-Anweisung eingegeben wird, erscheint etwas auf dem Bildschirm:

```
LET harald = 5
PRINT harald
```

Dies ist schon ein zweizeiliges Programm, das vom Computer der Reihe nach bearbeitet wird. In der ersten Zeile wird der Variablen 'harald' der Wert 5 zugewiesen, den sie so lange behält, bis er wieder geändert wird. Die PRINT-Anweisung gibt den Wert der Variable 'harald', der momentan 5 ist, auf dem Bildschirm aus. Variablen sind eine sehr praktische Erfindung, auch wenn das nicht jeder gleich verstehen wird. Zu einer Variablen kann leicht ein fester Wert oder eine weitere Variable dazugezählt werden:

```
LET harald = harald + 5
```

Diese Zeile dürfen Sie allerdings nicht als mathematische Gleichung betrachten, denn dann wäre sie falsch. Wenn die Variable 'harald' davor den Wert 5 hatte, dann wird 5 dazugezählt, und 'harald' hat nun den Wert 10. Probieren Sie doch einmal mehrere Beispiele aus.

Natürlich gibt es nicht nur die Variable 'harald'. Variablen können beliebige Namen bzw. Bezeichnungen haben, allerdings gibt es auch ein paar Regeln. So darf z. B. keine Zahl bzw. Ziffer am Anfang des Namens stehen, wohl aber ab der zweiten Stelle. Richtige Variablennamen sind:

```
harald
h1
i
Punkt79A
```

Bis jetzt wurden die Werte der Variablen direkt im Programm festgelegt. Wenn mehrere Rechnungen durchgeführt werden sollen, ist es jedoch umständlich, die Werte jedesmal im Programm zu ändern. Deshalb gibt es einen Befehl, der beim Ablauf des Programms darauf wartet, daß der Bediener etwas eingibt. Dieser Befehl heißt INPUT:

```
LET a = 0
INPUT a
PRINT a
```

Nach dem Starten mit RUN oder <AMIGA> + <R> erscheint im Basic-Fenster ein Fragezeichen. Dies bedeutet, daß der Computer auf eine Eingabe wartet. Wenn nun eine Zahl eingegeben und <RETURN> gedrückt wird, dann erscheint diese Zahl auf dem Bildschirm, eventuell aber in Exponentialdarstellung, wenn die Zahl zu groß war. (Wenn Sie Buchstaben oder andere alphanumerische Zeichen eingeben, dann erscheint die Fehlermeldung '?Redo from start' und das Fragezeichen wartet auf eine bessere Eingabe.)

Anmerkung: Sollte bei einer Eingabe der Bildschirm nur kurz aufflackern und sonst keine Reaktion zeigen, dann ist vermutlich das Basic-Fenster nicht aktiviert. Klicken Sie deshalb einmal kurz mit der Maus in dieses Fenster und versuchen Sie die Eingabe erneut.

Bis jetzt wurden nur Variablen verwendet, die vom Typ REAL sind. Dies bedeutet, daß diesen Variablen sowohl ganze (INTEGER) als auch Zahlen mit Nachkommastellen zugewiesen werden könnnen. Es gibt aber auch Variablen, die nur ganze Zahlen annehmen können. Dazu wird einfach an den Variablennamen ein Prozentzeichen angehängt. Dann ist für den Computer alles klar. Wir wollen diese Neuerung gleich einmal ausprobieren:

```
INPUT a%
PRINT a%
```

Die Variable a% kann nun nur noch ganzzahlige Werte annehmen. Wenn Sie trotzdem eine Gleitkommazahl (REAL) eingeben, dann wird diese gerundet.

Interessant ist auch, daß die Variablen a und a% völlig verschieden sind und somit auch getrennt behandelt werden können. Ein kleiner Test zeigt dies:

```
LET a = 5
INPUT a%
PRINT a%
PRINT a
```

Untereinander werden nun die Werte für a% und a ausgegeben, die je nach Eingabe völlig verschieden sind.

Nachdem nun der Typ INTEGER besprochen wurde, soll nicht verheimlicht werden, daß es auch noch einen anderen, sehr wichtigen Variablentyp gibt, die Zeichenkette (STRING). Bereits ganz am Anfang dieses Kurses wurde ein solcher String verwendet:

```
"hello world"
```

Wenn dieser Text einer Variablen zugeordnet werden soll, dann muß sie vom Typ STRING sein. Dazu wird der Variablenname mit einem Dollarzeichen versehen:

```
text$ = "hello world"
```

Genau wie schon bei den Integer- und Real-Variablen funktioniert jetzt die Ein- und Ausgabe eines Strings.

```
a$ = ""
INPUT a$
PRINT a$
```

Wenn nun das Fragezeichen erscheint, dann kann jedes auf der Tastatur befindliche Zeichen eingegeben werden, ohne daß eine Fehlermeldung erscheint.

Anmerkung: Vielleicht ist es einigen schon aufgefallen, daß vor der Zuweisung in der ersten Zeile hier kein 'LET' steht. Trotzdem wird dieses Beispiel funktionieren. Dies liegt daran, daß das Wort 'LET' zwar im Standard-Basic vorgesehen war, aber bei vielen späteren Basic-Dialekten nicht mehr vorgeschrieben wird. Es gibt aber gerade jetzt wieder ein Basic-Version, die dieses Befehlswort zwingend vorschreibt: TRUE BASIC, das von den Vätern des Ur-Basic entwickelt wurde (siehe Test in KICKSTART 8). Wer in AmigaBasic Programme schreibt, der kann also auf dieses Befehlswort verzichten, er sollte aber im Hinterkopf behalten, daß es so etwas gibt!

Entscheidungen

Im allgemeinen hört man, daß Computer ja so schlau sein sollen. Ehrlich gesagt, ist das eingemachter Blödsinn, zumindest momentan noch. Gewisse Grundkenntnisse hat ein Rechner aber schon, z. B. das Treffen von Entscheidungen. Ein einfaches Beispiel wird dies verdeutlichen. Der Rechner stellt eine Frage und prüft die Antwort. Ist sie korrekt, dann wird der Befragte beglückwünscht.

```
PRINT "Wieviel ist 9+6?"
INPUT A
IF A=15 THEN PRINT
   "Richtig mein Freund"
```

Die Abfrage ist, wie Sie sicher schon erkannt haben, in fast flüssigem Englisch abgefaßt. Übersetzt lautete es ungefähr so:

Wenn A gleich 15 dann drucke: "Richtig mein Freund"

Also gar nicht so schwer, aber trotzdem immer der Reihe nach: Zuerst wird die Frage auf dem Bildschirm ausgegeben. Dann fragt der Rechner nach der Lösung und wartet dabei solange, bis sie eingegeben wurde. Die Eingabe, z. B.:

15 <Returntaste>

veranlaßt den Rechner, zum nächsten Befehl zu laufen; das wäre 'IF'. 'IF' sagt dem Computer, daß nun eine Abfrage kommt. Der Ausdruck, der dahinter folgt, entscheidet, ob der nächste Befehl ausgeführt wird oder nicht. Gibt man nun in diesem Beispiel die Zahl 15 ein, so erscheint auf dem Bildschirm der Satz 'Richtig mein Freund'. Gibt man eine andere Zahl ein, erscheint diese Meldung nicht.

Aber die 'IF-Abfrage' kann noch viel mehr. Nehmen wir ein anderes Beispiel, etwas sehr computertypisches. Man hört viel von Hackern, die in andere Rechnersysteme eindringen und dort ihr Unwesen treiben. Dagegen will man sich natürlich schützen. Eine einfache Möglichkeit, so etwas zu realisieren, zeigt das nächste Programm.

Zuerst wird das Geheimwort festgelegt, denn der Rechner muß ja schließlich wissen, wie es heißt. Danach wird der Benutzer gefragt, wie seiner Meinung nach das Geheimwort lautet. Nach der Eingabe prüft der Rechner nun nach, ob das Wort richtig ist oder nicht. Ist dem so, begrüßt der Rechner den Benutzer freundlich. War das Passwort hingegen falsch, gibt der Rechner eine weniger freundliche Meldung auf den Bildschirm.

```
LET Geheimwort$="KICKSTART"
INPUT A$
IF A$=Geheimwort$ THEN
   PRINT "Korrektes Passwort"
   PRINT "Ich darf Sie höflichst
   willkommen heißen"
ELSE
   PRINT "Falsche Eingabe – Ich
   lasse mich nicht täuschen"
   PRINT "Ich lasse mich nicht täuschen"
   PRINT "Und nun nehmen Sie die
   Finger von meiner Tastatur"
ENDIF
```

Der Aufbau ist dem ersten Beispiel ähnlich. Ist die Abfrage richtig, werden alle Befehle von 'THEN' bis 'ELSE' ausgeführt, also zwei Zeilen Begrüßungstext ausgegeben. War die Abfrage hingegen negativ, so werden die Befehle von 'ELSE' bis 'ENDIF' ausgeführt, in unserem Beispiel drei Zeilen Ablehnungstext ausgegeben. 'ENDIF' stellt also die Endmarke der ELSE-Anweisung dar.

Schleifen

Nun folgt ein wichtiger Abschnitt der Basic-Programmierung: Die Schleifen, die sehr viel Arbeit abnehmen. Stellen Sie sich nur einmal vor, Sie wollten den Satz „Schwarzbraun ist die Haselnuß" zehnmal hintereinander auf den Bildschirm schreiben. Ein Möglichkeit wäre folgendes Programm:

```
Print "Schwarzbraun ist die Haselnuß"
Print "Schwarzbraun ist die Haselnuß"
Print "Schwarzbraun ist die Haselnuß"
Print "Schwarzbraun ist die Haselnuß"
Print "Schwarzbraun ist die Haselnuß"
Print "Schwarzbraun ist die Haselnuß"
Print "Schwarzbraun ist die Haselnuß"
Print "Schwarzbraun ist die Haselnuß"
Print "Schwarzbraun ist die Haselnuß"
Print "Schwarzbraun ist die Haselnuß"
```

Zugegeben, das Programm funktioniert, aber elegant ist es nicht gerade gelöst. Eine Schleife hilft hierbei weiter.

```
FOR I=1 TO 10
   PRINT "Schwarzbraun ist die
   Haselnuß"
NEXT I
```

Der dazu benutzte FOR-Befehl stellt den Anfang der Schleife dar. Die dahinter angegebene Variable benötigt man, um die Anzahl der Durchläufe anzugeben. In unserem Fall läuft die Schleife genau zehn mal und druckt den wohlbekannten Satz zehn mal auf den Bildschirm. Nach diesem prachtvollen Beispiel, das gleichzeitig auch das heimatliche Liedgut ein wenig wiederbelebt, etwas anderes:

```
FOR I=1 TO 10
   PRINT I
NEXT I
```

Der Variablen I wird zuerst der Wert 1 zugewiesen. In der nächsten Zeile erscheint der Inhalt der Variablen I auf dem Bildschirm und beweist, daß I zu diesem Zeitpunkt wirklich gleich 1 ist. Die Anweisung NEXT kennzeichnet das Ende der Schleife. Jetzt wird der Programmablauf mit der FOR-Anweisung fortgesetzt. Die Variable I wird um 1 erhöht und hat nun den Wert 2. Die Zahl wird wieder ausgegeben und die Schleife somit erneut durchlaufen. Dies geschieht solange, bis die Variable I den Wert 10 erreicht. Die Schleife wird dann verlassen, das Programm hat seinen Dienst getan.

Um sich eine kleine Vorstellung der Leistungsfähigkeit dieser Schleife zu verdeutlichen, noch ein kleines Programm:

```
FOR A=10 TO 15
   PRINT
   PRINT A;"  2 = ";A 2
   PRINT "-----------"
NEXT A
```

Wie man sieht, können innerhalb der Schleife auch mehrere Befehle stehen, und die Schleifensteuerungsvariable (hier 'A') kann auch noch zum Rechnen benutzt werden.

Das war der erste Teil des Kurses, in dem schon die wichtigsten Befehle von BASIC besprochen wurden – wohlgemerkt die wichtigsten und nicht die meisten, denn ein Basic-Dialekt wie z. B. AmigaBasic hat einen Umfang von rund 190 Befehlen. Sicher können wir im Rahmen dieses Kurses nicht annähernd alle besprechen, aber die interessantesten werden wir schon ausführlich vorstellen. Bis zum nächsten Mal wünschen viel Erfolg beim Üben der angesprochenen Befehle.

Harald & Markus.

AMIGA-WARE

067 Aegis Draw Plus A	485,–	095 Pro MIDI Studio A	345,–
003a Aegis Animator m. Images A	265,–	129 Publisher 1000 A	390,–
108 Alien Fires S	95,–	127 Roadwar 2000 S	95,–
114 Aztec C (Developers) A	620,–	072 Silent Service S	90,–
044 Bards Tale S	95,–	059 Sinbad S	95,–
088 Bureaucracy S	90,–	096 Sound Sampler A	225,–
111 Butcher A	75,–	126 Space Battle S	29,90
086 CLI-Mate A	65,–	057 Space Quest S	115,–
117 Cruncher Factory S	29,90	091 Starglider S	75,–
112 dBMan A	295,–	085 The Surgeon S	149,–
074 Deluxe Art Parts Vol II A	70,–	054 True Basic S	295,–
041 Deluxe Paint II A	265,–	125 TV Text A	225,–
013 Deluxe Video 2 A	265,–	076 Uninvited S	110,–
118 Demolition S	29,90	VIP Professional A	320,–
119 Digiview A	410,–		
076 Dynamic CAD A	980,–		
109 Feary Tale Adventure S	90,–		
036 Flight Simulator II S	95,–		
104 Flight Sim. Scenery Disc (East USA) S	55,–		
120 Flip Flop S	29,90		
063 Gridiron S	135,–		
123 Guild of Thieves S	85,–		
113 Hollywood Poker S	49,90		
105 Kampfgruppe S	140,–		
055 Kings Quest I-III S	95,–		
082 Marauder II A	80,–		
106 Metacompco Assembler A	185,–		
107 Metacompco Pascal A	185,–		
097 MIDI Interface (Mimetics) A	115,–		
098 Modula 2 (Regular Version) A	195,–		
099 Modula 2 (Developers Version) A	325,–		
100 Modula 2 (Commercial Vers.) A	630,–		
131 Mouse Hide (Leder)	29,90		
101 Mouse Pad	14,50		
130 Phalany S	29,90		
079 Portal S	95,–		
092 Prism A	135,–		

©ma

UND WEITERE
44 PROGRAMME

Alle Preise in DM

Porto bei Vorkasse 4,– DM
bei Nachnahme 6,– DM

PDC

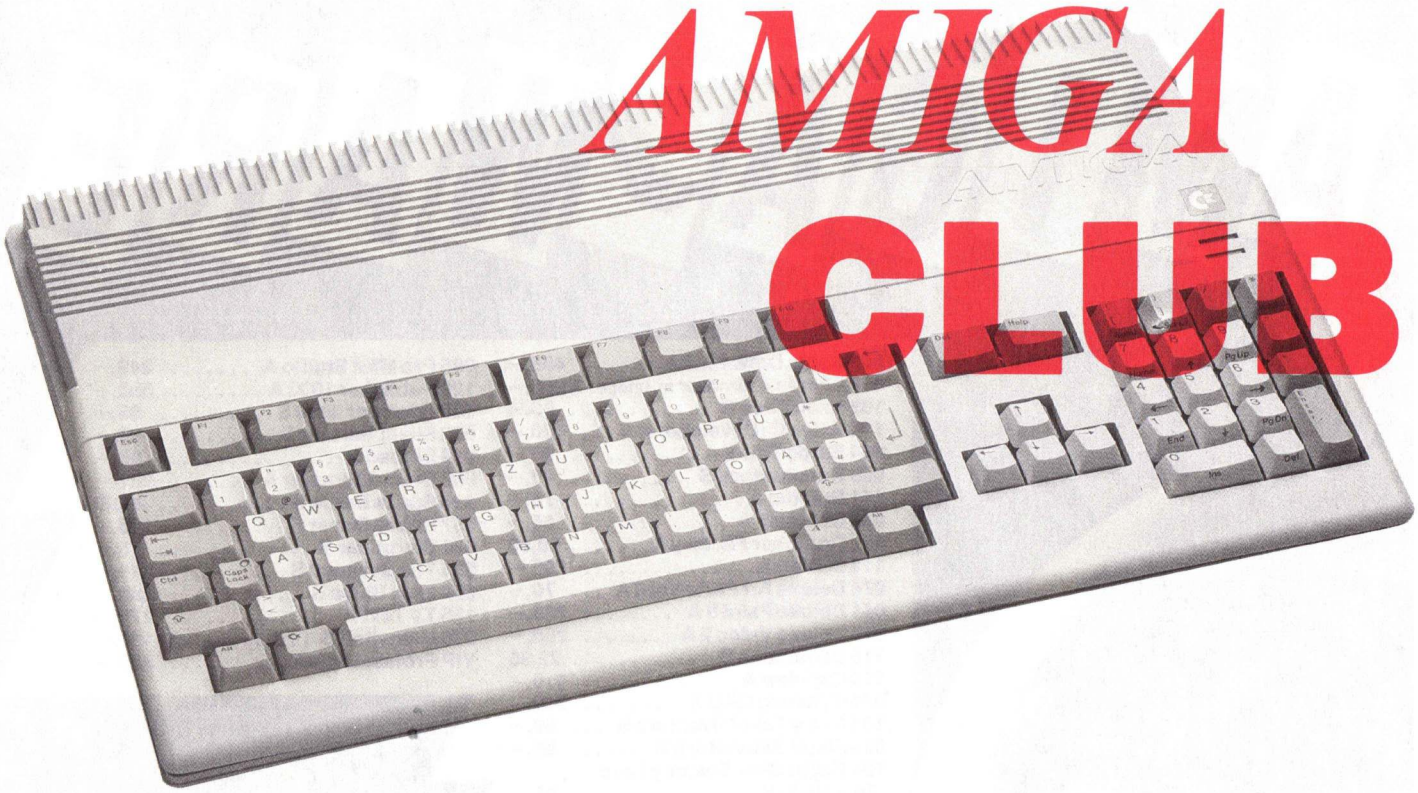
Bestellservice: 9 - 18.30 · 06172/24748
18.30 - 22 · 06171/53863

Auslandsbestellungen nur gegen Vorkasse + 6,– DM Porto

S - Spiel
A - Anwender

PDC GmbH · Louisenstr. 115 · Passage Alter Bahnhof · 6380 Bad Homburg · Tel. 06172/24748

AMIGA CLUB



In der vergangenen KICKSTART-Ausgabe (Juli/August) hatten wir alle AMIGA-Clubs dazu aufgerufen, uns ihre Kontaktadresse und die Aktivitäten, Schwerpunkte, Clubbeiträge usw. mitzuteilen. Zahlreiche Clubs haben uns daraufhin geschrieben.

Wir geben die Informationen an Sie weiter: An erster Stelle wird die Kontaktadresse genannt. Dann folgen allgemeine Informationen über den Club: Sie erfahren, ob eine Clubzeitschrift erscheint und ob ein Beitrag zu entrichten ist.

Interessenten, die vielleicht an einem AMIGA-Kreis mitwirken möchten, können sich aus unserer Liste die Anschrift des betreffenden Clubs heraussuchen. Ihren Briefen sollten Sie unbedingt Rückporto beilegen, um die jeweilige Clubkasse zu schonen!

LABBA SOFT
Der AMIGA Club
Postfach 22
5282 Ranshofen
Allgemeine Clubinfo:
Gedankenaustausch, gemeinsamer Einkauf von Hard- und Software, Programmieren (BASIC), kein Siezen.
Clubzeitung: „AMIGARIUS"
Beitrag: 3 DM pro Monat.

YING-YANG Club
Nick Dollhausen
Lenaustr. 38
6000 Frankfurt 1
Tel. 069/554037
Allgemeine Clubinfo:
Alles über und auf dem AMIGA.
Clubzeitung erscheint monatlich.
Beitrag: 2,50 DM pro Monat.

Rüsselsheimer Computer Verein 1986 e.V.
Helmut u. Angelika Koch
Holbeinstr.3
6090 Rüsselsheim
Tel. 06142/562449
Allgemeine Clubinfo:
Programmieren in BASIC und C, Erfahrungsaustausch, regelmäßiges Treffen, Wettbewerbe.
Clubzeitung vorhanden.
Beitrag: 5 DM für Erwachsene, 3 DM für Jugendliche.

A.U.G.E. 4000
c/o U. Tempelmann
Lochnerstr. 24
4030 Ratingen
Tel. 02102/23371
Allgemeine Clubinfo:
Messebesuche, Public Domain Pool, Hardwarebasteleien, Programmprojekte, Erfolgsfeiern.
Clubzeitung: Momentan keine.
Beitrag: Nicht bekannt.

ACD
Nachtigallenweg 50
2070 Ahrensburg
Allgemine Clubinfo:
Erfahrungsaustausch, Hilfe für Ein- und Umsteiger, Zeitschriftenbibliothek, verbilligter Hard- und Software Einkauf durch Sammelbestellungen.
Clubzeitung: Erscheint vierteljährlich.
Beitrag: 40 DM jährlich.

German Amiga User Club
z. Hd. Herrn Krumrey
Hansestr. 124
2400 Lübeck 1
Allgemeine Clubinfo:
Alles über und auf dem Amiga.
Clubzeitung: „Artistline", erscheint alle zwei Monate.
Kein Clubbeitrag.

C.U.S.
Chiffre Nr. 057462 C
Hauptpostlagernd
Saarbrücken
Mailbox Nebosa 0681/64624
Allgemeine Clubinfo:
Problemlösungen zu Hard- und Software, regelmäßiges Treffen, Softwaretausch.
Clubzeitung: Erscheint vierteljährlich, für Mitglieder kostenlos, ansonsten 2 DM.
Beitrag: Nicht bekannt.

GFC
Computer Clubs
Innsbruckerstr. 12
A-6250 Kundl
Tel. 05332/3763 Kurt
Tel. 05338/309 Ernst
Tel. 05332/6255 Martin
Allgemeine Clubinfo:
Auch andere Rechner außer Amiga (PCs, Atari ST, Macintosh).
Clubzeitung: Momentan noch nicht.
Beitrag: Nicht bekannt.

Public Projekt
Dechant-Röper Straße 32
5750 Menden 1
Tel. 02373/10225
(Samstag von 16 – 18 Uhr)
Allgemeine Clubinfo:
Software und Erfahrungsaustausch, gemeinsamer Vertrieb selbsterstellter Software, alles für Commodore Amiga und Schneider CPC.
Clubzeitung vorhanden.

AMIGA BUCHHITS

Was leisten die neuen Amigas? Hier finden Sie die Antwort. Unabhängig davon, ob Sie den Amiga schon haben oder den Kauf planen. Dieses Buch bietet Ihnen Entscheidungshilfen, technische Details und jede Menge von dem, was man mit Amiga 500 & 2000 so alles anstellen kann. Eben Informationen, die man braucht, wenn man sich für die neuen Amigas interessiert. Aufbereitet nach einem völlig neuartigen didaktischen Konzept, in einer Sprache, die zum Amiga paßt.

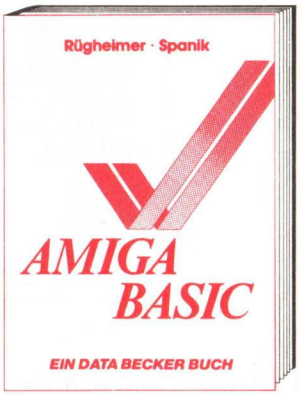
Das können Amiga 500 & 2000
190 Seiten, DM 29,–

Wählen Sie gleich den richtigen Einstieg zu Ihrem Amiga 500. Denn das Handbuch läßt Sie dabei völlig allein. Versuchen Sie es lieber gleich mit Amiga 500 für Einsteiger. Hier heißt es: Anschließen und loslegen. Verständlich für jedermann zeigt Ihnen dieses Buch: Workbench, AmigaBASIC, CLI und AmigaDOS. Locker aufbereitet bietet es Ihnen alles Wissenswerte. Bis hin zu den beim Amiga 500 mitgelieferten Zusatzprogrammen.

Amiga 500 für Einsteiger
343 Seiten, DM 39,–

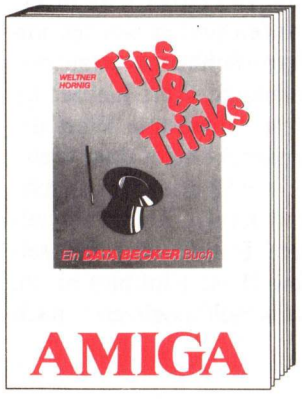
Läßt das Handbuch Sie auch in so manchen Dingen allein, das große Amiga-2000-Buch nicht. Hier finden Sie eine umfassende Einführung in die Arbeit mit Ihrem neuen Rechner – und mehr als das. Sind Sie erst einmal mit dem Amiga 2000 „per Du", zeigen Ihnen die Autoren, was einen Amiga-Profi ausmacht: Laufwerk-Einbau in den Amiga 2000, Speicher-Erweiterung, Arbeit mit der PC-Karte, Einbau und Einrichtung der Harddisk, Karten für Amiga- oder PC-Seite, Kickstart im RAM, und, und, und.

Das große Amiga-2000-Buch
Hardcover, ca. 600 Seiten
DM 59,–
erscheint ca. 8/87



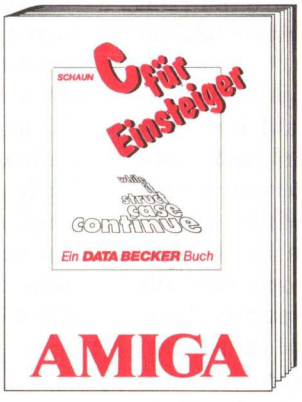
Das erfolgreiche Buch zu Amiga-BASIC – jetzt in Neuauflage! Erweitert um Kickstart 1.2, neuer Workbench und Amiga 500 & 2000. Mit allem, was BASIC-Programmierern Spaß macht: Grafik und Sound, Laden und Speichern von Graficraft-Bildern in BASIC-Programme, sequentielle und relative Dateien, Business-Grafik, Computeranimation, Windows, Umgang mit IFF-Bildern, Sprachausgabe und, und, und. Das Buch für Einsteiger, Aufsteiger und Profis.

AmigaBASIC
Hardcover, 774 Seiten, DM 59,–



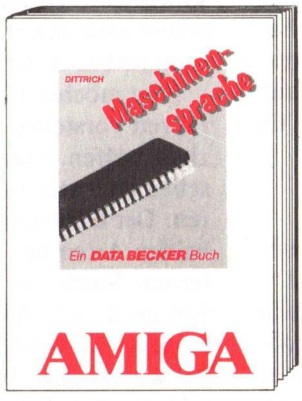
Amiga Tips & Tricks. Ein Buch, das voller Überraschungen steckt: 64 Farben gleichzeitig auf dem Amiga. Von BASIC aus Zugriff auf die Libraries. Benutzung verschiedener Zeichensätze in BASIC. Sinnvoller Einsatz von Windows, Screens und Menüs. Tips zu einzelnen Grafikbefehlen, Programm- und AmigaDOS-Routinen! Greifen Sie in die Trickkiste, und schon sind Dinge möglich, die man gar nicht gedacht hätte.

Amiga Tips & Tricks
Hardcover, 364 Seiten, DM 49,–



C an einem Wochenende? Durchaus möglich! Mit C für Einsteiger. Ein Einführungskurs, der Ihnen schnell und einfach die wichtigsten Grundlagen dieser Sprache vermittelt. Vom ersten Programm bis hin zu den Routinen in den Bibliotheken. Mit dem gesamten Sprachumfang und den besonderen Features von C. Zahlreiche Tips & Tricks zur Programmierung und eine Beschreibung der beiden Compiler Lattice C und Aztek runden das Ganze ab.

Amiga C für Einsteiger
254 Seiten, DM 39,–



Schreiben Sie Ihre Programme in Maschinensprache – und Sie werden sehen, wie schnell ein Amiga sein kann. Das nötige Know-how liefert Ihnen dieses Buch: Grundlagen des 68000, das Amiga-Betriebssystem, Druckeransteuerung, Diskettenoperationen, Sprachausgabe, Windows, Screens, Register, Pull-Down-Menüs ... Und damit Sie auch gleich praktisch arbeiten können, werden die wichtigsten Assembler vorgestellt.

Amiga Maschinensprache
Hardcover, 282 Seiten
DM 49,–

BESTELL-COUPON

Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir.

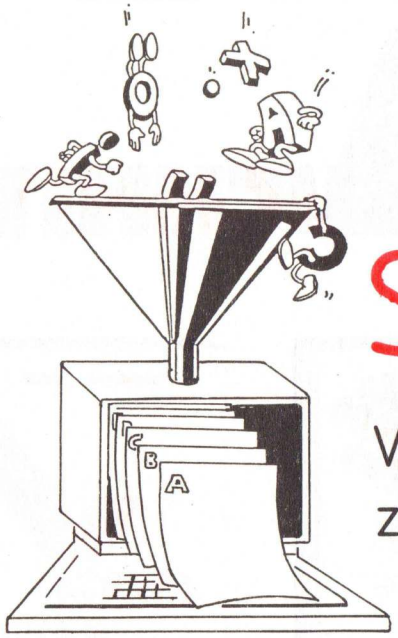
zzgl. DM 5,– Versandkosten
unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

Name

Straße

Ort

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010



Sortierprobleme

Vom einfachen BUBBLESORT zum QUICKSORT-

Fast jeder, der sich mit der Programmierung eines Computers beschäftigt hat, wird irgendwann auf das Problem des Sortierens von Datenbeständen gestoßen sein. Viele Lehrbücher für Programmiersprachen geben Algorithmen und Programme zur Lösung dieses Problems an. Unter dem Begriff Sortieren versteht man das Anordnen von Datensätzen in eine bestimmte Reihenfolge. Der Computer stellt folgende Vergleichsoperatoren zur Verfügung:
<,>,=,<>,>= und <=.

Diese Operatoren gelten sowohl für numerische als auch für alphanumerische Variablen. Bei alphanumerischen Vergleichen ist jedoch auf eine Besonderheit hinzuweisen: Es gilt

A < B < ... < Y < Z <
a < b < ... < y < z;

d. h. ein kleines „a" kommt beim Sortieren nach einem großen „Z". Hierin liegt auch der Grund, weshalb viele Computerprogramme nur Großbuchstaben bei Namen akzeptieren. Datensätze sind z. B. Buchtitel, Namen, u.s.w. Neben diesen einfachen Datensätzen, bei denen klar ist, wonach sortiert werden soll, gibt es häufig kompliziertere Datensätze, wie z. B. Adressen u. ä. Solch ein Adressensatz hat beispielsweise die folgende Form:

Vorname
Nachname
Straße
Hausnummer
Plz
Ort
Zustellpostamt

Bei der zuletzt genannten Art von Datensätzen ist es möglich, da daß Sortiermerkmal wechselt, d. h. einmal soll eine alphabetische Liste aller gespeicherten Personen erstellt werden, und zum anderen möchte man wissen, wer alles in Frankfurt wohnt (also Plz 6000), und diese Personen sollen auch in Form einer alphabetischen Liste ausgegeben werden. Das Teilfeld, nach dem ein Datenbestand sortiert werden soll, bezeichnet man als Schlüsselfeld (Key). Wenn nach mehreren Teilfeldern sortiert werden soll, bieten die meisten Programmiersprachen Stringfunktionen an, mit denen man sich das Schlüsselfeld zusammenbauen kann.

Nun möchte ich einige Sortierverfahren vorstellen und auf ihre Effizienz prüfen. Man unterscheidet zwischen internem und externem Sortieren. Der Unterschied zwischen diesen beiden Arten besteht darin: Beim internen Sortieren befinden sich alle Schlüssel im Arbeitsspeicher (was beim AMIGA schon eine ganze Menge sein kann). Im Gegensatz dazu gibt es den Fall, da der benötigte Platz nicht physikalisch vorhanden ist und man auf langsamere Speichermedien wie die Platte oder das Band ausweichen mu. Um solch große Datenmengen zu sortieren, gibt es Mischverfahren, mit denen ich mich hier nicht weiter beschäftigen möchte. Der interessierte Leser sei auf die Bibliographie am Ende des Artikels hingewiesen. Im folgenden gehe ich davon aus, da alle zu sortierenden Schlüssel in einem Array, dem Schlüsselarray, vorhanden sind. Also liegen die Daten in folgender Struktur vor: Key[1],Key[2],...,Key[n – 1],Key[n].

Der Einfachheit halber beschränke ich mich in den Beispielprogrammen auf das Sortieren von Integer-Zahlen. Für den Programmierer dürfte es kein Problem sein, diese Skelettprogramme an seine Anwendung anzupassen.

Der erste Algorithmus, den ich besprechen möchte, ist der Bubble-Sort. Er wird in vielen Lehrbüchern behandelt und ist um einiges schlechter als seine Verbreitung. Das Verfahren basiert auf dem Austausch von benachbarten Elementen in dem Schlüsselarray. Der Array wird mehrmals durchlaufen und es werden immer benachbarte Schlüssel vertauscht, wenn der vorherige Wert kleiner als der momentan betrachtete Wert ist. Formal ausgedrückt heißt das: Wenn Key[j – 1] > Key[j] dann vertausche Key[j – 1] und Key[j]. Dadurch steigen bei jedem Durchlauf Schlüsselwerte wie eine Blase (Bubble) bis zu ihrem Gewicht(Schlüsselwert) nach oben.

Bubblesort

Nun zur Analyse: Der Bubble-Sort benötigt

$$(n-1)+(n-2)+(n-3)+\ldots+(n-n) = \sum_{i=0}^{n-1} i$$

$$= \frac{n \star (n-1)}{2} = \frac{n-n}{2} \text{ Vergleiche.}$$

Die Anzahl der Zuweisungen ist minimal gleich 0 und maximal gleich

```
                Pass1 Pass2 Pass3 Pass4 Pass5 Pass6 Pass7 Pass8 Pass9
                 i=2   i=3   i=4   i=5   i=6   i=7   i=8   i=9   i=10

   Key[1] =10  --1     1     1     1     1     1     1     1     1
   Key[2] = 8 ¦ 10   --2     2     2     2     2     2     2     2
   Key[3] = 7 ¦  8   ¦ 10   --3     3     3     3     3     3     3
   Key[4] = 3 ¦  7   ¦  8   ¦ 10   --5     5     5     5     5     5
   Key[5] = 2 ¦  3   ¦  7   ¦  8   ¦ 10   --7     7     7     7     7
   Key[6] = 9 ¦  2--   3--   7   ¦  8   ¦ 10   --8     8     8     8
   Key[7] =15 ¦  9   --5     5--   7--   8--   10  ---9     9     9
   Key[8] =17 ¦ 15   ¦  9     9     9     9     9-   10    10    10
   Key[9] = 5 ¦ 17   ¦ 15    15    15    15    15    15    15    15
   Key[10]= 1-   5--  17    17    17    17    17    17    17    17

   Die Punkte sollen das Aufsteigen der einzelnen Werte verdeut


   Programm

   PROGRAM BubbleSort;
     CONST n=10;        (* Anzahl der Arrayelemente *)
     VAR i,j,Hilfe:Integer;

   Key:Array [1..n] of Integer;
BEGIN
  FOR i:=1 TO n DO READLN(Key[i]);  (* Werte in das Schlüsselfeld lesen)

                                    (* normalerweise von einer Datei)

  FOR i:=2 TO n DO
    BEGIN
      FOR j:=n DOWNTO i DO
        BEGIN
          IF Key[j-1] > Key[j] THEN
            BEGIN
              Hilfe:=Key[j];
              Key[j]:=Key[j-1];
              Key[j-1]:=Hilfe;
            END;
        END;
    END;
  FOR  i:=1  TO  n  DO  WRITELN(Key[i]);(* Sortiertes Array ausgeben *)

END.
```

Zahlenbeispiel Bubblesort

$\frac{3\star(n\ -n)}{2}$ Die mittlere Anzahl liegt demnach bei $\frac{3\star(n\ -n)}{4}$ Zuweisungen, wenn man davon ausgeht, daß die Schlüsselwerte gleich verteilt sind.

Wer sich das Zahlenbeispiel genauer ansieht, wird feststellen, da der Array schon nach Pass 7 sortiert ist. Eine Verbesserung besteht darin, eine Hilfsvariable einzuführen, die pro Pass auf TRUE gesetzt wird, wenn eine Vertauschung stattgefunden hat, ansonsten hat die Variable den Wert FALSE. Im Zahlenbeispiel würde der Algorithmus dann nach Pass 8 aufhören. Noch weiter läßt sich das Verfahren verbessern, indem man sich nicht nur merkt, ob eine Vertauschung stattgefunden hat, sondern zusätzlich noch behält, an welcher Stelle die Vertauschung war. Denn es ist klar, da alle Schlüsselwerte vor dieser Position schon in der gewünschten Reihenfolge sind, und nachfolgende Durchläufe können dann bei dieser Position abbrechen. Diese Eigenschaft folgt aus der Asymmetrie des Bubble-Sorts: Ein kleiner Schlüsselwert am unteren Ende des Arrays (großer Index) wird in einem Pass an die richtige Stelle gebracht. Beispiel:

Der Array 1 2 5 7 8 9 10 15 17 3 wird in einem Durchlauf sortiert. Ist dagegen ein großer Schlüsselwert am unteren Ende des Arrays plaziert, ist es nicht möglich, den Array in einem Pass zu sortieren. Beispiel: Um den Array 17 1 2 3 5 7 8 9 10 15 zu sortieren, braucht man 9 Durchläufe, obwohl auch nur ein Wert am falschen Platz ist. Hier bietet sich eine weitere Verbesserung an, indem man die Sortierrichtung nach jedem Pass wechselt, d. h. bei Pass 1 werden benachbarte Elemente vertauscht wenn Key [j − 1] > Key[j]. Die Schleife läuft abwärts, kleine Schlüsselwerte steigen auf. Bei Pass 2 werden benachbarte Elemente vertauscht, wenn Key [j − 1] < Key[j]. Die Schleife läuft aufwärts, große Schlüsselwerte sinken ab. Zusätzlich merkt man sich auch hier, an welcher Stelle die letzte Vertauschung stattfand, dies führt zu einer unteren Schranke.

Bidirektionaler Bubblesort

Eine genaue Analyse des verbesserten Bubble-Sort ist ziemlich kompliziert. Klar dürfte jedoch sein, da die

```
Obere  Schranke          2          3          4          5
Untere Schranke         10          9          5          4
Key[1] =                10-       + 1          1          1
Key[2] =                 8 -     +  8 -        2          2
Key[3] =                 7  -  +    7  -       7----++++++3
Key[4] =                 3  - +     3    -     3++++--    5
Key[5] =                 2  - +     2     -+++ 5       ----7
Key[6] =                 9  - +     9    +- - 8          8
Key[7] =                15   +- - 10  ++      9          9
Key[8] =                17  +      15 +       10         10
Key[9] =                 5 +        5+        15         15
Key[10]=                 1+        17         17         17

+ Symbolisiert aufsteigende Werte
- Symbolisiert sinkende Werte

Programmbeispiel für den Bidirektionalen Bubble-Sort mit 2
Schranken

PROGRAM Bubble1;
  CONST n=10;        (* Anzahl der Arrayelemente *)
  VAR k,i,j,Hilfe,UntereSchranke,
      ObereSchranke,Vertauschung:Integer;
      Key:Array [1..n] of Integer;
  BEGIN


FOR i:=1 TO n DO READLN(Key[i]);
     (* Werte in das Schlüsselfeld lesen *)
     (* Normalerweise von einer Datei    *)
ObereSchranke:=n;
```

Bidirektionaler Bubblesort

Anzahl der Vertauschungen nicht abnimmt, sondern nur die Anzahl der unnötigen Vergleiche reduziert wird. Da im allgemeinen jedoch eine Vertauschung von Variablen länger als ein Vergleich dauert, leisten die Verbesserungen doch nicht so viel, wie man sich erhofft hat. Die Laufzeit befindet sich auch bei diesem Algorithmus in der Größenordnung von n^2 Schritten.

Nun möchte ich noch eine höhere Sortiermethode vorstellen. Es handelt sich um den von C.A.R. Horae 1962 entwickelten Partition-Exchange-Sort oder auch Quicksort. Es ist eine der schnellsten internen Sortiermethoden, die bis heute bekannt sind.

Das Verfahren basiert, wie auch der Bubble-Sort, auf dem Austausch von Elementen. Beim Quicksort werden die Elemente über groe Distanzen vertauscht und nicht wie beim Bubble-Sort die benachbarten.

```
        Vertauschung:=n;

        REPEAT
            FOR j:=ObereSchranke DOWNTO UntereSchranke DO
              BEGIN
                IF Key[j-1] > Key[j] THEN
                  BEGIN
                    Hilfe:=Key[j];
                    Key[j]:=Key[j-1];
                    Key[j-1]:=Hilfe;
                    Vertauschung:=j;
                  END;
              END;
            UntereSchranke:=Vertauschung+1;
            FOR j:=UntereSchranke TO ObereSchranke DO
              BEGIN
                IF Key[j-1] > Key[j] THEN
                  BEGIN
                    Hilfe:=Key[j];
                    Key[j]:=Key[j-1];
                    Key[j-1]:=Hilfe;
                    Vertauschung:=j
                  END;
              END;
            ObereSchranke:=Vertauschung-1;
          UNTIL UntereSchranke>ObereSchranke;
          FOR i:=1 TO n DO WRITELN(Key[i]);
        END.
```

Bidirektionaler Bubblesort

Algorithmus:

1. Wähle einen mittelgroßen Wert aus dem zu sortierenden Array aus.
2. Durchsuche den Array und vertausche Elemente, so da der Array nachher in zwei Teile geteilt ist. Im linken Teil befinden sich nur noch Schlüsselwerte, die kleiner als der ausgesuchte Wert sind, und im rechten Teil stehen alle Elemente, die größer sind.
3. Diese Schritte werden für jeden so entstandenen Teilbereich wiederholt, bis der Array sortiert ist.

```
Zahlenbeispiel

   10  8  2  3  7  9  15  17  5  1        i=5  Vergleichswert=7
    :                          :
    1  8  2  3  7  9  15  17  5  10       10,1 werden vertauscht
       :                       :
    1  5  2  3  7  9  15  17  8  10       8,5  werden vertauscht
```

```
Program quicks;
var Key:array[1..10] of integer;
    i:Integer;
  Procedure sort(l,r:Integer);
    var
      i,j:Integer;
      Mitte,w:Integer;
    begin
      i:=l; j:=r;
      Mitte:=Key[(l+r) div 2];
      repeat
        while Key[i] < Mitte do i:=i+1;
        while Mitte < Key[j] do j:=j-1;
        if i<=j then
        begin
          w:=Key[i]; Key[i]:=Key[j]; Key[j]:=w;
          i:=i+1; j:=j-1;
        end;
      until i > j;
      if l<j then sort(l,j);
      if i<r then sort(i,r);
  end;{Procedure sort}
begin
  for i:=1 to 10 do readln(Key[i]);
  sort(1,10);
  for i:=1 to 10 do writeln(Key[i]);
end.
```

Programmbeispiel für einen Quicksort

Quicksort

Nun befinden sich alle Elemente, die kleiner als 7 sind, auf der linken Arrayseite, und alle Elemente größer 7 auf der rechten Seite. Jetzt macht man dasselbe mit dem linken und dem rechten Teil.

Programmbeispiel für einen Quickssort

Als Vergleichswert wählt das Programm immer den Wert, der in der Mitte des Arrays steht. Die beiden While-Schleifen suchen von unten bzw. von oben ein Element, das größer bzw. kleiner als der Vergleichswert ist. Wenn zwei solcher Werte gefunden sind, werden sie vertauscht, aber nur dann, wenn der kleinere Wert rechts und der größere Wert links im Array vom Vergleichswert steht. Das ganze wird solange wiederholt, bis al-

le Werte, die kleiner als der Vergleichswert sind, auf der linken Seite und alle, die größer sind, auf der rechten Seite im Array stehen. Dann wird die Procedure sort rekursiv einmal mit der linken Seite und dann mit der rechten Seite aufgerufen u.s.w., bis der Array sortiert ist.

Die Analyse des Quicksort ist mathematisch schwer. Ich möchte sie an dieser Stelle auslassen und den Leser auf das Buch von Knuth (siehe Literaturangaben), Seite 119, verweisen. Die Laufzeit liegt in der Größenordnung von n★ld(n) Schritten (ld ist der Logarithmus zur Basis 2). Dies gilt aber nur wenn als Vergleichswert immer mittelgroße Werte gewählt werden. Der schlechteste Fall ist der, wenn immer der größte Wert als Vergleichswert gewählt wird. Dann liegt die Laufzeit in der Größenordnung von n Schritten. Die Wahl eines geeigneten Vergleichswertes ist also außerordentlich wichtig. Der Quicksort ist für große Datenbestände sehr schnell (n>10). Bei sehr kleinen Mengen wirkt sich die Verwaltung des Stacks für die Rekursion negativ auf die Laufzeit aus. Verbesserungen liegen darin, daß man nicht den gesamten Bestand mit dem Quicksort sortiert, sondern nur solange rekursiv vorgeht, bis der Array in Teile der Länge n < = 10 vorsortiert ist. Dann nimmt man eines der oben angegebeben Verfahren (Bubble-Sort), um den Bestand in die gewünschte Ordnung zu bringen.

Roland Foerster

Bibliographie

Donald E. Knuth:
The Art of Computer programming
Vol.3 / Sorting and Searching
Addison-Wesley Publishing Company
1973

Dr. Dr. h.c. Niklaus Wirth:
Algorithmen und Datenstrukturen
B. G. Teubner, Stuttgart 1983

FONT as FONT can!

Sie sind kurz vor der Vollendung Ihres neuesten Programms, doch der letzte Schliff fehlt noch? Oder haben Sie ein Projekt von vornherein aufgegeben, weil Ihnen spezielle Zeichen fehlten? Dies muß nicht sein – lesen Sie selbst.

Das Betriebssystem des Amiga erlaubt es dem Anwender, eine beliebige Anzahl von Zeichensätzen auf einem Massenspeicher bereitzuhalten. Sie können bei Bedarf geladen und in das System eingebunden werden. Dies ermöglicht die individuelle Gestaltung eigener Programme. Denken Sie nur an einige mathematische Sonderzeichen oder vielleicht an ein kleines Textverarbeitungsprogramm mit kyrillischem oder „nur" altdeutschem Alphabet.

Beschäftigen wir uns zunächst mit einer wichtigen Eigenart des Amiga: Er kennt keinen speziellen Textausgabemodus wie etwa die Rechner, die mit einem MDA (Monochrom Display Adapter) zusammenarbeiten, bei dem jeweils eine Speicherstelle von einem Byte einem Buchstaben aus dem integrierten Zeichensatzeprom zugeordnet ist. Die Textdarstellung des Amiga erfolgt stattdessen durch Übertragung der Fontdaten in den sichtbaren Teil des Videorams, d. h. die Ausgabe des Textes erfolgt durch Zeichnen der Buchstaben als grafisches Objekt auf dem Bildschirm. Diese Technik erlaubt eine einfache Vermischung von Text und Grafik auf demselben Bildschirm; als Bildschirm ist hier nicht der physikalische Schirm des Monitors, sondern ein virtueller 'Screen' zu verstehen.

Die Ausgabe des Textes erfolgt über den RastPort des Screens. Die Positionierung des Cursors kann über die Funktion Move(&RastPort,x,y) vorgenommen werden. Die Werte für die Variablen x und y beziehen sich auf die Baseline des benutzten Fonts. Für die Ausgabe der obersten Textzeile bedeutet dies einen y-Wert, der um den Wert 'Baseline' größer als Null sein muß, da die Text-Funktion sonst in einen Bereich schreiben würde, der außerhalb der RastPort-Grenzen liegt. Abbildung 1 zeigt ein kleines Programmbeispiel und erklärt die verwendeten Variablen der einzelnen Funktionen.

Laden

Um eine Textausgabe mit einem bestimmten Font zu erreichen, muß dieser zuerst geöffnet werden. Dabei stehen dem Programmierer zwei Funktionen zur Verfügung. Die Handhabung dieser Routinen sehen Sie auf Bild 2.

Einbinden

Nachdem der gewünschte Font für der Zugriff geöffnet wurde, kann er mit Hilfe der Funktion SetFont(font, rp) für den angegebenen RastPort installiert werden. Die Zuweisung eines neuen Fonts bleibt bis zu einem erneuten Aufruf von SetFont mit anderen Parametern erhalten. Die verschiedenen Möglichkeiten, eine Textausgabe zu beeinflussen, sind in Abbildung 3 zusammengefaßt.

Ein eigener Font

Alle dem System verfügbaren externen Zeichensätze sind unter dem Zugriffspfad für FONTS: gespeichert. Die Starteinstellung ist auf den Pfadnamen sys:fonts gesetzt. Sie kann jedoch vom Benutzer mit Hilfe des Assign-Befehls umgeleitet werden (siehe AmigaDos Handbuch). Wenn Sie sich einen Eintrag in dieses Directory anschauen, könnte er wie folgt aussehen:

fontxyz.font
fontxyz(dir)

Die Datei fontxyz.font enthält den sogenannten FontContentsHeader. In diesem Headerfile sind die Eigenschaften der im gleichnamigen Directory abgelegten Zeichensätze beschrieben. Der Aufbau eines solchen Files ist in Abbildung 4 dargestellt. Die Definition des Zeichensatzes ist in den Dateien des zugehörigen Fontdirectory enthalten. Die dazu nötige Datenstruktur wird durch 'struct TextFont' in der Includedatei graphics/text.h festgelegt. Die verwendeten Variablen sind in Abbildung 5 erläutert.

Los geht's!

Im abgedruckten Beispielfont können Sie die Umsetzung der dargestellten Strukturen und Variablen erkennen und im Zweifelsfall noch einmal nachvollziehen. Der Konstruktion eigener Zeichensätze steht nun nichts mehr im Weg. Sollten Sie einen besonders gelungenen Font erstellen, können Sie ihn gerne der Redaktion zusenden. Bei ausreichender Beteiligung werden wir eine Public-Domain-Diskette mit den eingegangen Fonts erstellen und über unseren PD-Service vertreiben.

(GC)

Die hier vorgestellten Routinen können im Zusammenhang mit den Fonts verwendet werden.

AddFont(textFont)

Die Funktion AddFont() macht einen beliebigen Font für das System verfügbar.
textFont muß die Adresse einer gültigen Textfont Struktur gemäß der Definition im Includefile 'graphics/text' enthalten.

C :

```
AddFont(Eigener_textFont);
```

Assembler :

```
AddFont(textFont) , gfxbase
          A1          A6
```

AskFont(rp,textAttr)

Die Funktion AskFont() ermittelt die Eigenschaften des dem Rastport rp momentan zugewiesenem Font. Die ermittelten Werte werden in der angegebenen textAttr Struktur (gemäß graphics/text) gespeichert.

C :

```
AskFont(rp,&Meine_textAttr);
```

Assembler :

```
AskFont( rp , textAttr ) , gfxbase
          A1      A0          A6
```

AskSoftStyle(rp)

Die Funktion AskSoftStyle() gibt den softwaremäßig generierten Stil des monentan gesetzten Fonts des Rastports rp aus (siehe hierzu z.B. Style MenÜ im Notepad). Der Stil kann mit der Funktion SetSoftStyle() verändert werden.

C :

```
stil = AskSoftStyle(rp);
```

Assembler :

```
stil = AskSoftStyle( rp ) , gfxbase
 D0                  A1          A6
```

AvailFonts(Speicher,Speichergröße,Typ)

Die Funktion AvailFonts() gibt dem aufrufendem Programm im angegebenen Speicher die dem System zur Verfügung stehenden Fonts bekannt. Der Speicherplatz kann mit Hilfe der Variablen Speichergröße festgelegt werden. Die letzte Variable bestimmt die Art der abgefragten Fonts (Typ = AFF_MEMORY sucht im Arbeitsspeicher und Typ = AFF_DISK auf der Diskette). Der Rückgabewert Fehler gibt die Zahl der Bytes an, die der Funktion fehlten, um alle Informationen zurückzugeben. Die Daten entsprechen dem Typ AvailFontsHeader in 'graphics/text'.

C :

```
Fehler = AvailFonts(Speicher,Größe,AFF_xxxxx);
```

Assembler :

```
Fehler = AvailFonts( Speicher , Größe , Typ ) , gfxbase
  D0                  A0          D0     D1         A6
```

CloseFont(font)

Die Funktion CloseFont() gibt den mit OpenFont() geöffneten Font an das System zurück. font ist der Wert der von OpenFont() geliefert wurde.

C :

```
CloseFont(font);
```

Assembler :

```
CloseFont( font ) , gfxbase
            A1        A6
```

OpenDiskFont(textAttr)

Die Funktion OpenDiskFont() öffnet den durch textAttr definierten Font. Die Struktur textAttr ist in der Includedatei 'graphics/text' enthalten.

C :

```
font = (struct TextFont *)OpenDiskFont(&Meine_textAttr);
```

Assembler :

```
font = OpenDiskFont( textAttr ) , diskfontbase
 D0                    A0              A6
```

OpenFont(textAttr)

Die Funktion ist ähnlich der vorherigen. Sie arbeitet jedoch mit den Fonts die im Moment schon im System integriert sind (z.B. Romfonts bzw. Fonts die mit AddFont installiert wurden).

C :

```
font = (struct TextFont *)OpenFont(&Meine_textAttr);
```

Assembler :

```
font = OpenFont( textAttr ) , gfxbase
 D0                A0          A6
```

RemFont(textFont)

Die Funktion RemFont() entfernt einen mit AddFont in das System eingebundenen Font.

C :

```
Fehler = RemFont(textFont);
```

Assembler :

```
Fehler = RemFont( textFont ) , gfxbase
  D0                 A1          A6
```

SetAPen(rp,pen)

Die Funktion SetAPen() stezt die Vordergrundfarbe auf den Wert pen. Eine Zahl zwischen 0 und 255 ist hierbei möglich.

C :

```
SetAPen(rp,pen);
```

Assembler :

```
SetAPen( rp , pen ) , gfxbase
          A1    D0        A6
```

SetBPen(rp,pen)

SetBpen() ist im Gegensatz zu der letzten Funktion für die Hintergrundfarbe zuständig.

```
SetBPen(rp,pen);
```

Assembler :

```
SetBPen( rp , pen ) , gfxbase
          A1    D0        A6
```

SetFont(rp,font)

Die Funktion SetFont() weist dem Rastport rp den Font font als standart Ausgabefont zu. (Alle jetzt folgenden Textausgaben benutzen den neuen Font)

C :

```
Fehler = SetFont(rp,font);
```

Assembler :

```
Fehler = SetFont( rp , font ) , gfxbase
  D0                A1    A0        A6
```

SetOPen(rp,pen)

Diese Funktion setzt die Fullfarbe des Rastports rp auf die Farbe pen.

C :

```
SetOPen(rp,pen);
```

Assembler :

```
SetOPen( rp , pen ) , gfxbase
          A1    D0        A6
```

SetSoftStyle(rp,stil,freigabe)

Diese Funktion erlaubt es dem Programmierer dem Font bestimmte Eigenschaften zuzuweisen. (siehe auch Stylebits im Fontheader). stil beschreibt den gewünschten Stil (alle Attribute mit oder verknüpft) und freigabe ist die Bitmaske der zu änderenden Stilbits.

C :

```
Stil = SetSoftStyle(rp,stil,freigabe);
```

Assembler :

```
Stil = SetSoftStyle( rp , stil , freigabe ) , gfxbase
 D0                    A1    D0        D1          A6
```

Text(rp,text,zähler)

Die Funktion Text() hat drei Parameter in der Übergabezeile. Die erste Variable rp enthält die Adresse des zur Textausgabe gehörigen Rastport. Die Handhabung dieser Variablen kann im Programmbeispiel ersehen werden. (Beachten Sie den unterschiedlichen Syntax bei Fenstern/Bildschirmen .)
text enthält einen Zeiger auf die Zeichenkette, die über den Rastport ausgegeben werden soll. Die Länge dieser Zeichenkette (= Anzahl aller Buchstaben incl. aller Leerzeichen) muß in der Variablen zähler übergeben werden.

C :

```
Fehler = Text(rp,"Dies ist der Text",17);
```

Assembler :

```
Fehler = Text( rp , text , zähler ) , gfxbase
 D0             A1    A0    D0 0-16      A6
```

```
TextLength(rp,text,zähler)

    Die Funktion TextLength() bestimmt die Anzahl der Pixel in
    horizontaler Richtung ( x Achse ) die der String text bei der
    Ausgabe in den durch rp bestimmten Rastport benötigen würde.
    Die Variable zähler bestimmt wiederum die Zeichenanzahl.

    C :

    länge = TextLength(rp,"von diesem Text",15);
      D0                   A1     A0     D0 0-16
```

Der FontContentsHeader enthält Informationen über die Fontgruppe die unter dem entsprechenden Directory abgespeichert ist.

Ein solcher Header ist durch folgende Struktur definiert :

```
struct FontContentsHeader {
  UWORD  fch_FileID;
  UWORD  fch_NumEntries;
  struct FontContents;
        ...
  };
```

Variablen :

fch_FileID - FontHeader ID

Diese Variable enthält den Hexwert 0x0f00 als Kennzeichnung des Dateitypes als Fontheader.

fch_NumEntries - Fontheader Font Eintrage

Hier ist die Anzahl der Fonts abgelegt die durch die FontContents Struktur beschrieben sind. Der Wert entspricht der Zahl der im Header enthaltenen FontContents Strukturen.

FontContents Stuktur :

```
struct FontContents {
  char   fc_FileName[MAXFONTPATH];
  UWORD  fc_YSize;
  UBYTE  fc_Style;
  UBYTE  fc_Flags;
  };
```

FileName[MAXFONTPATH] - Name des Fontfiles

Dieser String enthalt den Zugriffspfad auf die Fontdefinition. Beispiel :

Unter dem Pfad sys:fonts/fontxyz/12 ist der Font fontxyz der Höhe 12 abgespeichert. Die Variable Fontname hat dann den Wert "fontxyz/12".
Die Stringlänge ist auf MAXFONTPATH (z.Z. 256) Zeichen mit NULL (0x00) zu erweitern.

fc_YSize - Fonthöhe
Diese Wortvariable beschreibt die Anzahl der Pixel in Y-Richtung die der Font benötigt.

fc_Style - Fontstil
Kennzeichnet bestimmte Eigenschafften eines Fonts.

```
0x00    der Font hat keine Besonderheiten
0x01    der Font ist unterstrichen
0x02    der Font ist fett
0x04    der Font ist 'italic'
0x08    der Font ist gestreckt ( x-Richtung )
```

fc_Flags - Fontkennzeichnung

Der Wert für die Flagvariable errechnet sich aus der Summe der gesetzten Bits. Die Bedeutung der einzelnen Bits ist folgende :

```
0x01    der Font ist im Rom
0x02    der Font ist diskettenorientiert
0x04    der Font ist 'rückwerts' definiert
        z.B. Hebräisch
0x08    der Font ist für einen 640x200 Bildschirm
        definiert
0x10    der Font ist für einen 320x400 Bildschirm
        definiert
0x20    der Font ist für proportional Ausgabe
        definiert ( variable Breite der Buchstaben )
0x40    der Font ist 'designed'
0x80    der Font ist zur Zeit nicht resistent im
        Speicher

Beispiel :  Proportional + Designed = 0x20+0x40
            Flags = 0x60
```

Zusammenfassend für einen Beispielfont 'fontxyz' den es nur in der Höhe 14 Pixel gibt :

```
FONTS:
      \
       fontxyz.font
       fontxyz(dir)
             \
              14
```

fontxyz.font enthält dann :

```
0f00   0001  fontxyz/14 0x00 ... 0x00   0x000e  0x00  0x60
```

Mit Hilfe des folgenden kleinen Abasic Programm kann ein solcher FontContentsHeader leicht erstellt werden :

```
MAXFONTPATH = 256
FontName$ = "xyzfont.font"
OPEN "FONTS:xyzfont.font" FOR OUTPUT AS #1
PRINT #1,CHR$(15);                          : REM 0x0f
PRINT #1,CHR$(0);                           : REM 0x00
PRINT #1,CHR$(0);                           : REM 0x00
PRINT #1,CHR$(1);                           : REM 0x01

PRINT #1,FontName$;
FOR i = 1 to MAXFONTPATH - LEN(FontName$)   : REM Mit NULL
  PRINT #1,CHR$(0);                         : REM auffüllen
NEXT i
PRINT #1,CHR$(0);                           : REM 0x00
PRINT #1,CHR$(14);                          : REM 0x0e
PRINT #1,CHR$(0);                           : REM 0x00
PRINT #1,CHR$(96);                          : REM 0x60
CLOSE #1
END
```

Der eigentliche Font wird durch die TextFont Struktur definiert

```
struct TextFont {
   struct Message tf_Message;
   UWORD  tf_YSize;
   UBYTE  tf_Style;
   UBYTE  tf_Flags;
   UWORD  tf_XSize;
   UWORD  tf_Baseline;
   UWORD  tf_BoldSmear;
   UWORD  tf_Accessors;
   UBYTE  tf_LoChar;
   UBYTE  tf_HiChar;
   APTR   tf_CharData;
   UWORD  tf_Modulo;
   APTR   tf_CharLoc;
   APTR   tf_CharSpace;
   APTR   tf_CharKern;
};
```

Die Variablen haben folgende Bedeutungen :

tf_Message - Eine Messagestruktur die dem Betriebssystem die Länge der Fontstruktur mitteilt.

tf_YSize - Enthält die Höhe des Fonts in Pixel.

tf_Flags - Diese Variable beschreibt die Eigenschaften des Fonts. Die Bedeutung ist die selbe wie in der FontContentsHeader.

tf_XSize - Gibt die Breite der definierten Buchstaben an.

tf_Baseline - Beschreibt die Lage der Basiszeile im Font. Unterhalb dieser Zeile wird vom Betriebssystem der Unterstrich im Unterstreichungsmodus eingeblendet.
Der Wert darf mit größer als tf_YSize-1 gesetzt werden.

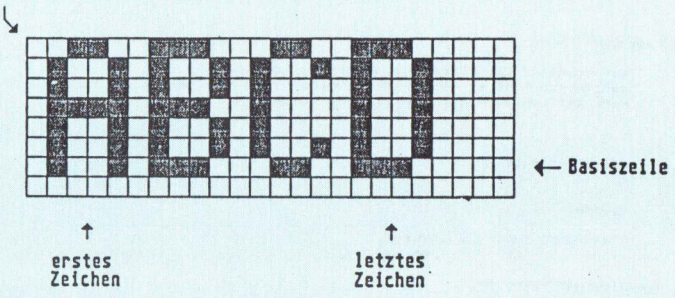
tf_BoldSmear - Diese Variable kontrolliert die 'Anschlagsstärke' der Textausgabe. Die Zeichen werden im Fettdruckmodus nach rechts 'verschmiert'. Der Wert in tf_BoldSmear legt die Anzahl der Pixel fest mit denen das Zeichen verbreitert wird.

tf_Accessors - Diese Variable enthält einen Zugriffszähler. Der Wert dieses Worts wird bei einem Zugriff mit OpenFont um eins erhöht und bei einem Aufruf von CloseFont um eins erniedrigt. Der Startwert ist 0x00.

tf_LoChar - Dieses Byte enthält den Wert des niedrigsten durch den Zeichensatz definierten ASCII Zeichen.
Beispiel : erstes definiertes Zeichen = 'A'
tf_LoChar = 65
= 0x41

tf_HiChar - Gibt das letzte definierte Zeichen im Zeichensatz an.

tf_CharData - Enthält einen 32 Bit Zeiger auf die Zeichendefinition.



dieses Beispiel würde zu folgenden Werten führen :

```
tf_LoChar = 0x41
tf_HiChar = 0x44

CharData :      ( tf_CharData zeigt hierher )

0x0338c , 0x0e000
0x04552 , 0x09000
0x04550 , 0x09000
0x07690 , 0x09000
0x04550 , 0x09000
0x04552 , 0x09000
0x0468c , 0x0e000
```

Definiert man einen Font auf diese Weise, ist nach dem durch HiChar bestimmten Zeichen noch ein weiteres zu definieren. Dieser Charakter wird vom Betriebssystem immer dann ausgegeben wenn ein Zeichen gedruckt werden soll das nicht in den durch Lo- und HiChar vorgegebenen Grenzen liegt. (siehe Beispielfont).

tf_Modulo - Gibt die Anzahl von Bytes an die im Zeichendefinitionsarray addiert werden müssen um die Werte der nächsten Zeile zuerreichen.

Im obrigen Beispiel wäre der Wert für tf_Modulo gleich 8.

tf_CharLoc - Dieser Zeiger deutet auf ein Feld von 32 Bit Werten. Diese wiederum setzen sich aus zwei 16 Bit Worten zusammen. Pro definiertem Zeichen existiert ein Langwort. Das erste Word der 32 Bitzahl beschreibt den Bitoffset des entsprechenden Zeichen ,das zweite die Breite dieses Zeichens.

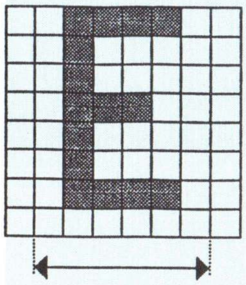
CharLoc: (tf_CharLoc zeigt hierher)

0x00000005 , 0x0005000a , 0x000a000f , 0x000f0014

tf_CharSpace - Enthält einen Zeiger auf ein Wortarray in dem die Breite der einzelnen Buchstaben abgelegt ist. Dieser Zeiger kann NULL sein, wenn alle definierten Zeichen die Breite tf_XSize haben.

tf_CharSpace = 0x00000000 (alle Zeichen gleich breit

oder

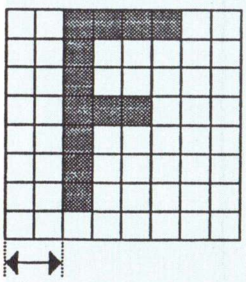


wenn die Zeichendefinition den durch die Pfeile angedeuteten Bereich umfaßt ist der Wert im tf_CharSpace Array für das Zeichen 'E' gleich 6 .

tf_CharKern - Ein Zeiger auf ein Array von Worten . Dieses Array gibt die Zahl der Nullbits an die von der durch CharLoc beschriebenen Position abgezählt werden können bis das erste Datenbit erreicht wird.

tf_CharKern = 0x00000000 (alle Zeichen linksbündig definiert)

oder



dieser Pfeil zeigt den Wert für das entsprechende Datenwort im tf_CharKern Array des Buchstaben 'F' CharKern[F] = 0x0002

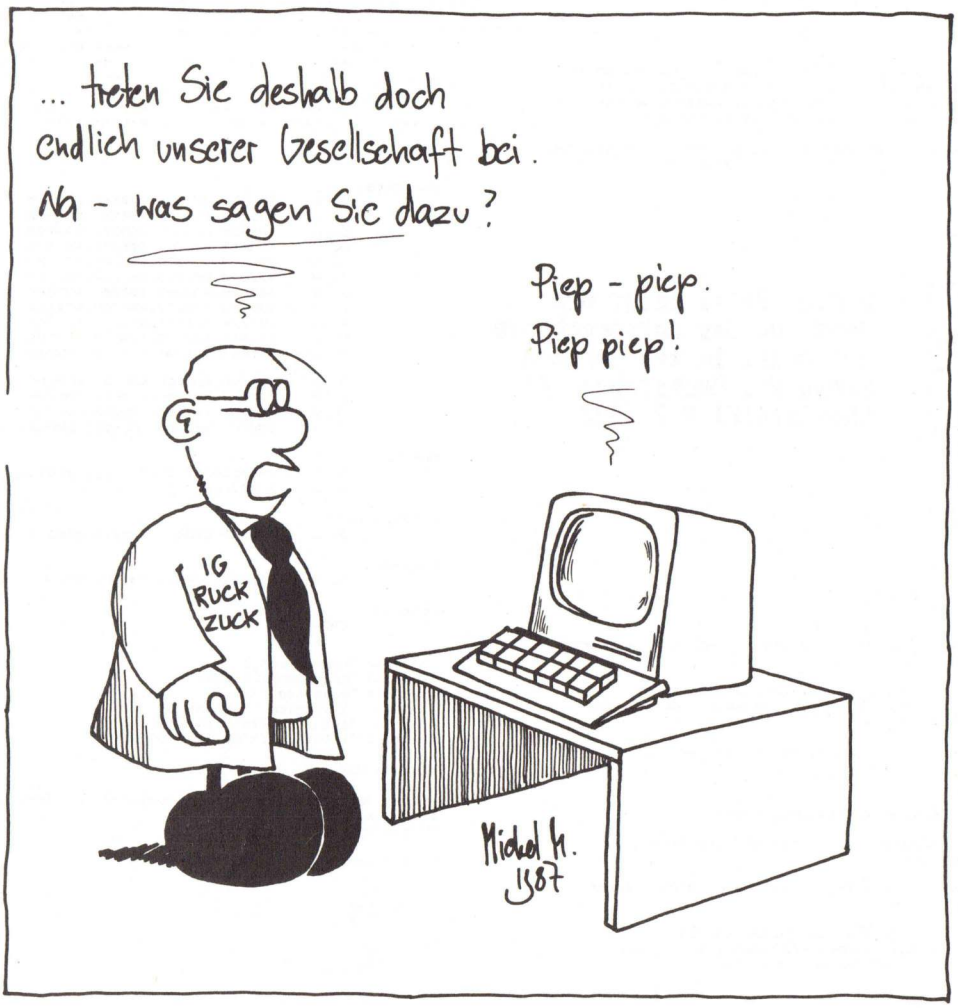
Um den hier abgedruckten Beispielfont in das System zu integrieren sind folgende Schritte nötig :

1. Erstellen Sie eine Datei Namens xyzfont.font im Directory FONTS: . Sie können hierzu das Basicprogramm aus Abbildung 4 direkt übernehmen.
2. Legen Sie ein Direktory mit dem Namen xyzfont an :
   1> CD FONTS:
   1> MAKEDIR xyzfont
3. Assemblieren Sie die unten abgebildete Datei.
4. Linken Sie die entstandene Objektdatei mit dem Befehl
   1> ALINK FROM 14.OBJ TO 14
5. Kopieren Sie die entstandene Datei '14' nach FONTS:xyzfont
   1> COPY 14 TO FONTS:xyzfont
6. Wenn bis jetzt alles fehlerfrei verlaufen, ist können Sie nun das Notepad im Utility Ordner aufrufen und über die Menüzeile den neuen Font einladen und testen.

```
;       Datei 14.asm
;       enthält den Beispielfont zu xyzfont

            INCLUDE   "exec/types.i"
            INCLUDE   "exec/nodes.i"
            INCLUDE   "libraries/diskfont.i"

            MOVEQ     #0,D0           ; Die Fontdatei ist als normales
            RTS                       ; Programmodul angelegt um dem Betriebs-
                                      ; system die Möglichkeit zu eröffnen
                                      ; die Funktionen LoadSegment() und
                                      ; UnloadSegment() zur Verwaltung des
                                      ; Speichers zu verwendem.
                                      ; Mit der RTS Anweisung wird die Datei
                                      ; bei einem versehendlichen Start
                                      ; sofort wieder verlassen.

            DC.L      0               ; ln_Succ
            DC.L      0               ; ln_Pred
            DC.B      NT_FONT         ; ln_Type
            DC.B      0               ; ln_Pri
            DC.L      FontName        ; ln_Name
            DC.W      DFH_ID          ; Dateikennzeichner
            DC.W      1               ; Revisionsnummer
            DC.L      0               ; Segment
FontName:
            DS.B      MAXFONTNAME     ; Name
Font:
            DC.L      0               ; ln_Succ
            DC.L      0               ; ln_Pred
            DC.B      NT_FONT         ; ln_Type
            DC.B      0               ; ln_Pri
            DC.L      FontName        ; ln_Name
            DC.L      0               ; mn_ReplyPort
            DC.W      FontEnd-Font    ; mn_Length
            DC.W      14              ; tf_Ysize
            DC.B      0               ; tf_Style
            DC.B      FPF_DESIGNED+FPF_PROPORTIONAL    ; tf_Flags
            DC.W      19              ; tf_XSize
            DC.W      12              ; tf_Baseline
            DC.W      1               ; tf_BoldSmear
            DC.W      0               ; tf_Accessors
            DC.B      88              ; tf_LoChar  'X'
            DC.B      90              ; tf_HiChar  'Z'
            DC.L      FontData        ; tf_CharData
            DC.W      8               ; tf_Modulo
            DC.L      FontLoc         ; tf_CharLoc
            DC.L      FontSpace       ; tf_CharSpace
            DC.L      FontKern        ; tf_CharKern

 ; Hier sehen Sie die Fontdefinition
 ; Die einzelnen Zeichen sind direkt aneinander gehängt. Um
 ; sie wieder zu trennen sind die Informationen in FontLoc,
 ; FontSpace und FontKern nötig.
 ;
 ; ***        *****          ***************** *   **   *****
 ;   **      **  **          ****         ** ** *  *  *     *
 ;    **    **   **          **          **  * **  *  *     *
 ;     **  **     **        **          **   *  *   **      *
 ;      ** **      ***      **         **
 ;       ***        *******         *******    *   * *  *   ****
 ;      ** **          **          **         **    ***     *
 ;     **   **        **         ***         **      *    *
 ;    **     **  **    **       **           *  *  **    ****
 ;   **      **  **   **        **        **
 ; ***        ***  ****          ***************
 ;
 ;

FontData:
            DC.W      $0e00f,$0803f,$0ffe4,$063e0
            DC.W      $03019,$0803c,$001b4,$09080
            DC.W      $01821,$08030,$0032c,$09080
            DC.W      $00c60,$0c060,$00624,$06080
            DC.W      $006c0,$03060,$00c00,$00000
            DC.W      $00380,$01fc0,$0ff12,$093c0
            DC.W      $006c0,$000c0,$0300c,$07080
            DC.W      $00c60,$00180,$0e00c,$01100
            DC.W      $01830,$0c301,$08012,$063c0
            DC.W      $03018,$0c603,$000c0,$00000
            DC.W      $0e00e,$07c0f,$0ffc0,$00000

            DC.W      $00000,$00000,$00000,$00000
            DC.W      $00000,$00000,$00000,$00000
            DC.W      $00000,$00000,$00000,$00000
            DC.W      $00000,$00000,$00000,$00000

FontLoc:
            DC.L      $0000000f,$0000f000d,$0001c000e
            DC.L      $0002a0011

FontSpace:
            DC.W      000017,000015,000016,000019

FontKern:
            DC.W      000002,000002,000002,000002

FontEnd:
            END
```

```
#include "exec/types.h"
#include "graphics/gfxbase.h"
#include "graphics/text.h"
#include "libraries/diskfont.h"
#include "intuition/intuitionbase.h"
#include "intuition/intuition.h"

#define REV     29L

extern void *OpenLibrary(),*OpenScreen(),*OpenDiskFont();
extern long Text();
extern void Move();

struct NewScreen NeuerBildschirm = {
        0,
        0,
        640,
        200,
        4,
        0,1,
        HIRES,
```

```
        CUSTOMSCREEN,
        NULL,
        (UBYTE *)"Kleines Beispiel",
        NULL,
        NULL
        };

struct Screen *Bildschirm;
struct RastPort *rp;
struct TextFont *tf;
struct TextAttr TextAttribut;
struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct DiskfontBase *DiskfontBase;

main()
{
        if(!(DiskfontBase = OpenLibrary("diskfont.library",REV)))
            {
              printf("Kann Diskfont Library nicht finden ! Schade\n");
              exit(FALSE);
            };
        if(!(IntuitionBase = OpenLibrary("intuition.library",REV)))
            {
              printf("Kann Intuition Library nicht finden ! Ende\n");
              CloseLibrary(DiskfontBase);
              exit(FALSE);
            };
        if(!(GfxBase = OpenLibrary("graphics.library",REV)))
            {
              printf("Kann Graphics Library nicht finden ! ????\n");
              CloseLibrary(DiskfontBase);
              CloseLibrary(IntuitionBase);
              exit(FALSE);
            };

        Bildschirm=(struct Screen *)OpenScreen(&NeuerBildschirm);

        TextAttribut.ta_Name=(UBYTE *)"sapphire.font";
        TextAttribut.ta_YSize=14;
        TextAttribut.ta_Style=0;
        TextAttribut.ta_Flags=0x60;

        if(!(tf = (struct TextFont *)OpenDiskFont(&TextAttribut)))
            {
              printf("Sapphire Font weg !!\n");
              CloseScreen(Bildschirm);
              Ende();
            };

         rp=&Bildschirm->RastPort;

        SetFont(rp,tf);
        Move(rp,50L,50L);
        Text(rp,(UBYTE *)"Hallo Sapphire Font\n",19L);

        Delay(600L);

        CloseFont(tf);
        CloseScreen(Bildschirm);

        Ende();
}

Ende()
{
        CloseLibrary(GfxBase);
        CloseLibrary(IntuitionBase);
        CloseLibrary(DiskfontBase);
        exit(TRUE);
}
```



GO AMIGA!

TEXTVERARBEITUNG + PUBLISH
* GoAmiga Text 174
Prowrite 242
Prowrite deutsch . 273
Talker 135
* Vizawrite 290
* Dynamic Word 385
* Word Perfect 763
Pagesetter 290
Laserscript 87
Publisher 386
Zuma Fonts 58
Fast Fonts 78
DATENBANKEN
GoAmiga Datei 174
Organize 193
Acquisition 579
dBMan 290
TABELLENKALKULATION
Analyze 2.0 290
Maxiplan plus 385
INTEGRIERTE SOFTWARE
Logistix 288
VIP Professional . 290
DFUEBERTRAGUNG
Diga 155
Online 135
Bulletin Board Sys 193
GRAFIK SOFTWARE
Images 78
Art Pak I 78
Impact 174
Brushworks I 58
Brushworks 2 58
Butcher 72
Deluxe Paint 2 ... 251
Deluxe Data I 58
Deluxe Data 2 58
Deluxe Data 3 58
Prism 159
* Digi Paint 116
GRAFIK HARDWARE
Digi View 386
Easel 963
VIDEO
Animator 271
Deluxe Video 1.2 . 271
EF/X 570
* Videoscape 3D 386
TV Text 193
CAD
Draw plus 502
Logicworks 193
Dynamic CAD 965
MUSIK SOFTWARE
Sonix 155
Deluxe Musik 2.0 . 193
Musik Data I 58
Pro Studio 288
MUSIK HARDWARE
Future Sound 338
ECE Midi Interface 116
GOL Midi Interface 153
MIM Midi Interface 95
MIM Sound Sampler 192
SUN Sound Sampler 174

SPIELE
Accolade
Mean 18 Golf 59
Activision
Hacker I 49
Portal 78
Aegis Development
Arazoks Tomb 97
Baudville
Video Vegas 68
Classic Image
Diablo 49
Dark Horse
Chessmate 58
Electronic Arts
Chessmaster 87
Space Quest 78
Adventure Construc 78
Kings Quest 78
Ultima 3 97
Ogre 58
* Autoduel 97
Finally Software
Dr. Xes 97
Firebird
Guild of Thieves . 87
* Starglider 87
Infinity Software
Galileo 193
ISM
Surgeon 97
Jagware
Alien Fires 78
Meridian Software
* Games Gallery 58
Micro Illusions
Fairy Tale Advent 97
* Land of Legens ... 97
* Fire Power 49
* Turbo 49
* Galactic Invasion 49
Microprose
* Gunship 78
Miles Computing
Quintettes 87
Mindscape
Shadowgate 97
Ininvited 97
Balance of Power 97
Deja Vue 97
Sinbad & Falcon . 78
Brataccas 49
Halley Project .. 78
Polarware
Crimson Crown ... 78
Psygnosis
Arena 78
SSI
* Roadwar 2000 78
Sublogic
Flight 2 deutsch 128
Interkabel zu FS2 10
Scenery Disk 7 ... 49
Scenery Disk II .. 49
LERNEN
Master Type 78

BASIC
AC Basic 377
True Basic 290
True Runtime 290
True Developers .. 97
True 3D Graphics . 97
* Sams Basic .....
Basic Professional 386
MODULA 2
Modula 2 - Reg ... 193
Modula 2 - Dev ... 290
Modula 2 - Com ... 579
Modula 2 - Kermit 58
Modula 2 - Examples 49
Modula 2 - Editor 97
Modula 2 - Grid 97
PASCAL
Pascal Professional 386
MCC Pascal 193
C
Lattice C + TxUt . 450
Manx C - Dev 578
Manx C - Com 963
FORTRAN
AC Fortran 570
ASSEMBLER
MCC Assembler 193
MCC Toolkit 97
KOPIERPROGRAMME
Mirror 97
Marauder 2 78
WORKBENCH + CLI
Enhancer 1.2 29
DOS Express 58
Zing 189
CLI Mate 78
MCC Shell 135
DIENSTPROGRAMME
Floppy Accelerator 68
Transfer Atari - AM 107
Transfer IBM - AM 135
Transfer C 64 - AM 97
Hacker Pack 97
C Montior V 2.0 .. 179
Explorer 97
Drive Alignment Kit 578
Zing Keys 109
TXED Texteditor .. 78
Disk Pro plus 58
Custom Screens ... 135
Power Windows 174
Diskettenverwalter 99
Harddisk Backup .. 139
Encore 76
Wow 76
Key Genie 97
PRODUKTIVITAET
Gizmos 2.0 135
Flipside 116
* Order 97
Flow 193
Outline 97
Maxidesk 134
IFF GRABBER
Grabbit 58
* DBuddy 155

SPEICHERWEITERUNGEN
Alegra 0,5 M RAM 732
Alegra Expansion 58
aMega 1 M RAM 999
Insider 1 M RAM 714
Alegra 2 M RAM 1446
Cage 2 M RAM 1349
* Megaboard 2 M RAM 1156
Cage 4 M RAM 2505
HARDDISK
C Ltd. 22 M 1930
Supra 20 M 1921
Supra 30 M 2307
Supra 60 M 3851
HARDWARE ALLGEMEIN
Timesave Uhr+Macro 155
Kickstart Eliminat 251
Diskmappe fuer 3,5" 19
MousePad klein ... 10
MousePad gross ... 20
Flickermaster 35
Schablone DOS 20
Schablone BASIC .. 20
Kabel parallel ... 35
Kabel RS 232 m/w 29
Kabel RS 232 m/m 29
Kabel Midi 25
BUECHER ENGLISCH
Amiga DOS Manual . 49
User Guide Graphics 37
The Amiga System . 31
Microsoft Amiga .. 39
BUECHER DEUTSCH
Handbuch Flight 2 . 31
Handbuch Prowrite . 31
ZEITSCHRIFTEN ENGLISCH
Amazing Computing . 7
Robo City News 4
AMIGA MAGAZINE AUF DISK
Jumpdisk 19
PUBLIC DOMAIN
Gesamtliste deutsch 0
FAUG Hot Mix 10er . 130
FRED FISH 10er 130
AMICUS 10er 130
DEMODISKETTEN
Demo TV Text 15
Demo Zing 15
Demo GoAmiga Datei 15
Demo Perfect Sound 15
Demo Zuma Fonts ... 15
Demo Animator 15
Demo Draw 15
Demo True Basic ... 15
Demo Pagesetter ... 15

INFO

Diese Preisliste ist nur ein kleiner Auszug aus unserem lieferbaren Gesamtsortiment. Bitte fragen Sie telephonisch an.
Unsere Gesamtpreisliste erhalten Sie mit vielen attraktiven Angeboten kostenlos.

Bestellservice:

BRD: 0041-1-3115959
CH: 01-3115959

Geschäftszeiten:
10.00–12.30, 13.30–18.30 Uhr, außer montags,
Sa.: 10.00–16.00 Uhr.

Versand ins Ausland nur Vorkasse (Scheck, bar, Visa Card, Master Card) zzgl. DM 7,– Porto.

softwareland
Franklinstraße 27
CH-8050 Zürich (Schweiz)

* = Lieferbar nach Verfügbarkeit.

Lieferungen ins Ausland sind zu deklarieren.

Preisänderungen vorbehalten.

PUBLIC DOMAIN SERVICE

Für den AMIGA gibt es schon eine Unmenge von Public-Domain-Programmen; manche Anbieter haben über 100 Disketten in Ihrem Programm. Die verschiedenen Sammlungen sind jedoch zum Teil nicht sortiert und in sich sehr unübersichtlich. Die Disketten wurden meist in den USA zusammengestellt (z.B. die 'Fish-Disks'), andere stammen aus den unterschiedlichsten Quellen. Dies hat dazu geführt, daß sich einige Programme auf mehreren Disketten wiederfinden. Hinzu kommt eine Vielzahl von Updates, also verbesserten Programmversionen, die die alten überflüssig machen. Ein weiterer Kritikpunkt an den meisten bestehenden Sammlungen ist die fehlende oder unzureichende Dokumentation, mit denen die Programmsammlungen angeboten werden. In den Listen steht oft nur eine Sammlung von Namen, deren Bedeutung jedoch nur selten deutlich wird. Fast immer fehlen Angaben über die Programmiersprache, den verwendeten Compiler und die notwendige Kickstartversion.

Aus diesem Dilemma soll Ihnen der Public-Domain-Service der KICKSTART helfen. Die Disketten enthalten ausschließlich auf ihre Funktionstüchtigkeit getestete Programme. Die einzelnen Disketten werden nach festen Kriterien zusammengestellt. Jede Diskette hat also einen Schwerpunkt (z.B. Lehrgänge, Bilder-Show, C-Programme, Spiele, u. ä.). Außerdem werden Angaben über die Programmiersprache, den verwendeten Interpreter oder Compiler usw. gemacht.

Diese Aktion, die uns einige schlaflose Nächt gekostet hat, soll Ordnung in die bestehende Public-Domain-Software bringen, damit Sie den größtmöglichen Nutzen daraus ziehen können. Wir hoffen, daß Sie mit der Einteilung und dem Ergebnis zufrieden sind und würden uns über Ihre Anregungen zu diesem Thema sehr freuen. Schreiben Sie uns, wenn Sie einen Verbesserungsvorschlag machen wollen! Wir haben dafür ein offenes Ohr und werden sicher einige dieser Vorschläge umsetzen.

Die Programme der Disketten 1 – 20 laufen auf allen AMIGA-Computern mit Kickstart/Workbench 1.2. Allerdings sollten 512k Speicher vorhanden sein. Sollten dennoch irgendwelche Einschränkungen gelten, werden wir dies bei den betreffenden Programmen angeben.

Das aktuelle Angebot

PUBLIC DOMAIN SERVICE

Diskette 1: C SOURCE

Eine Programmsammlung, die besonders dem Anfänger zeigt, wie man Intuition programmiert. Die Beispiele liegen als C-Quellcode und auch als fertige Programme vor, die sofort gestartet werden können.

Diskette 2: Spiele

- YachtC
  (Würfelspiel für 4 Personen)
- Puzzle
- Missile
  (verteidigen Sie Ihre Stadt, starker Sound)
- TriClops (sehr schönes 3D-Spiel)
- Breakout (3D-Effekt mit Brille)
- Trek73
  (bekannte Star Trek Variante)

Diskette 3: Spiele

HACK: Das bekannte Textadventure, das ursprünglich auf UNIX-Rechnern erstellt wurde, liegt hier als spezielle Grafikversion für den Amiga vor.

Diskette 4: Terminal-Programme

KERMIT: Bekanntes, luxuriöses Terminalprogramm (drei verschiedenen Versionen, mit Source-Code in C).

Diskette 5: Terminal-Programme

- WOMBAT (VT102/52 Emulator, XModem, Autodial)
- VT100 (grafikfähig, Source in C)
- TermPlus (XModem, Source in C)
- DG210
  (Data General D-210 Emulator)
- Ahost (XModem, Kermit)
- TeK4010 (XModem, VT100)

Diskette 6: Terminal-Programme

- Speech Term
  (spricht den empfangenen Text, XModem)
- StarTerm
  (mit Phone, Duplex, XModem)
- Argo Term
- PD Term (Source in C)
- AmigaDisplay
- Kermit

Diskette 7: UTILITIES

- QuickCopy
  (gutes Kopierprogramm)
- DirUtil (File-Copy)
- FileZap (File-Monitor)
- DiskZap (Disk-Monitor)
- DiskSalv (Diskettenretter)
- System-Monitor
- CSH (UNIX-ähnlicher Shell)

Diskette 8: Spiele

Monopoly: Sehr schöne Grafik, einfache Mausbedienung (Source in A-BasiC).

Diskette 9: Grafik-Show

- Grafik-Show mit bekannten Cartoons

Diskette 10: Grafik-Show

JUGGLER DEMO: Ein bewegliches Männchen jongliert mit drei verspiegelten Kugeln, sehr schöne Demo.

Diskette 11: Grafik-Show

RAY TRACERS: Wunderschöne räumliche Bilder, die auf einer VAX berechnet und auf den AMIGA übertragen wurden.

Diskette 12: Grafik

- Digitalisierte Bilder mit erstaunlicher Qualität (IFF-Format)

Diskette 13: Grafik

- Sehr schöne Bilder-Show (IFF-Format)

Diskette 14: EDITOR

Bekannter Texteditor MICROEMACS Version 30. Viele Features: Search/Replace/Copy.

Diskette 15: Grafik

verschiedene mit dem AEGIS-ANIMATOR erstellte Filme incl. PLAYER zum Abspielen der Filme (einige Filme benötigen auf einem AMIGA 1000 mehr als 512k Speicher!).

Diskette 16: Sprachen

XLISP 1.7 (neueste Version) mit ausführlicher Anleitung (über 50k)

Diskette 17: Sprachen

MODULA-2: Pre-Release eines Modula-Compilers mit verschiedenen kleineren Beispielprogrammen, die als Source-Code vorliegen.

Diskette 18: Grafik

MANDELBROT

Diskette 19: Grafik-Show

- Sehr schöne digitalisierte H.A.M.-Bilder

Diskette 20: Grafik-Show

'Fred the Baker und Rose's Flower Shop': COMIC-Film, der die Multitasking-Fähigkeiten des AMIGA erklärt.

Diskette 21: AMIGA-Tutor

Einführung in die Bedienung des AMIGA 500. Ein farbenfroher Lehrgang, der ganz am Anfang beginnt und mit vielen Bildern und Grafiken die Grundbegriffe des AMIGA erklärt (für Anfänger, komplett in Deutsch).

Diskette 22: Sprachen

MVP-FORTH und C-FORTH
C-Forth ist ein recht leistungsfähiger FORTH-Interpreter, der auch als Quelltext vorliegt.

Diskette 23: Grafik-Show

Viele abwechslungsreiche Motive in verschiedenen Auflösungen, verpackt in einer Grafik-Show (startet automatisch!)

Diskette 24: Grafik-Show

Sehr schöne digitalisierte Frauengesichter (startet automatisch!)

Diskette 25: UTILITIES

CLOCK, Portar, Checkmodem, PopCLI, MACView, Kickbench, Disassembler, Tracker und vieles mehr

Diskette 26/27: Grafik-Show

Auf zwei randvollen Disketten erleben Sie eine einmalige Dia-Show mit hervorragend digitalisierten futuristischen Bildern in voller PAL-Auflösung. Dazu gibt es stimmungsvolle, spärische Musik (startet automatisch!)

Diskette 28: Editoren

Auf dieser Diskette befinden sich einige schöne Editoren (UEDIT, MED, BLITZ) mit dazugehörigen Zeichensatz-Utilities (UeTurbo, Blitzfonts, Pearlfonts).

Diskette 29: UTILITIES

PrtDrvGen: erstellt Drucker-Treiber
DropShadow: jedes Fenster bekommt einen Schatten
MemClear: löscht den Speicher
ScreenSave: speichert den Bildschirm auf Diskette
Compress: komprimiert Programme

Diskette 30: Sound-Demos

Changing Minds, Joan Lui, Miami Vice II, Respectable, Holiday

Versandbedingungen:

Um einen schnellen und problemlosen Versand zu gewährleisten, beachten Sie bitte folgende Punkte:

- Legen Sie pro bestellter Diskette DM 10,– als Scheck bei.
- Fügen Sie dem Betrag folgende Versandkosten (für Porto und Verpackung) bei: Inland DM 5,–, Ausland DM 10,–.
- Zu dem Kostenbeitrag müssen wir leider pro Scheck eine Scheckgebühr von DM 0,50 berechnen. Verwenden Sie deshalb nur einen Scheck, auf dem die Gesamtsumme steht (z. B. für zwei Disketten DM 25,50).
- Für ausländische Besteller sei erwähnt, daß Sie mit Euroschecks auch in DM bezahlen können.
- Legen Sie unbedingt einen Aufkleber mit Ihrer vollständigen Adresse bei.

KICKSTART-Redaktion
Industriestr. 26
6236 Eschborn
Tel. 06196/41245

Wichtig!

Die Software wird nicht auf 'No-name'-Disketten geliefert, sondern auf Qualitätsdisketten der Marke FUJI FILM, mit denen wir sehr gute Erfahrungen gemacht haben.

C-KURS TEIL 1

Der in dieser Ausgabe beginnende C-Kurs soll Ihnen, liebe Leserin (hoffentlich ist auch einmal eine Frau unter den Lesern eines Computermagazins!) und lieber Leser, die Scheu vor einer Programmiersprache nehmen, die keineswegs so schrecklich kompliziert und unübersichtlich ist, wie es ein Blick auf das Listing eines C-Enthusiasten glauben lät. Natürlich kann auch dieser Kurs keine Garantie dafür leisten, da Sie sich am Ende sofort zu den 'Helden der C-Programmierung' (was auch immer das sein mag) zählen können, doch er soll einen Versuch darstellen, Ihnen den Anfang des Weges so weit wie möglich zu erleichtern.

Die Lektüre dieser Seiten kann Ihnen freilich nicht den Lerneffekt bieten, den man durch praktische Handhabung erzielt. Arbeiten Sie also zuhause mit eigenen Programmen weiter und experimentieren Sie mit speziellen Eigenschafften Ihres Compilers, auf die in diesem Kurs nicht eingegangen werden konnte. Sie gewinnen dadurch Sicherheit und Routine mit den Möglichkeiten, die Ihnen C bietet.

Fangen wir also am besten gleich mit der Praxis an und wenden uns dem ersten Projekt zu. Für die ganz neuen unter Ihnen sind die verwendeten Systembefehle und die Compilerbedienung noch einmal separat in den unterlegten Bereichen erklärt.

Nun geht's aber los!

Das erste Programm, mit dem wir beginnen, soll einen Text auf den Bildschirm ausgeben. Geben Sie dazu bitte das folgende Listing mit einem Editor in Ihren Rechner ein (z. B. 1 > ed programm1.c).

```
main()
{
printf(„ Zwischen diesen Anführungszeichen kann jeder Text stehen n“);
}
```

Übersetzen Sie nun dieses Programm mit dem Compiler und starten Sie es durch die Eingabe des Programmnamens (1 > programm1). Das Programm hat nun den Text zwischen den beiden Anführungszeichen auf den Bildschirm ausgegeben (zu dem ' n' später mehr).

Die erste Zeile unseres kleinen Programms besteht aus main(). Diese Anweisung sagt dem Compiler, daß hier die Funktion 'main' beginnt. Innerhalb einer Funktion können beliebige Befehle oder weitere Funktionen aufgerufen werden. Die beiden geschweiften Klammern { } fassen dabei die Anweisungen zusammen, aus denen die Funktion besteht. In Pascal wären dies z. B. die beiden Wörter 'begin' und 'end'. Hinter dem Funktionsnamen folgen die beiden runden Klammern; dies bedeutet, da an die Funktion keine Parameter übergeben werden. Diese Klammern gehören zu der Sprachdefinition von C und können auch dann nicht einfach weggelassen werden, wenn wie in unserem Fall gar keine Variablen übergeben werden sollen.

Mit der Funktion 'main' hat es eine weitere Besonderheit auf sich. Sie muß in jedem Programm vorhanden sein, da das fertig übersetzte Programm immer mit der ersten Anweisung der Funktion 'main' beginnt. Die erst Anweisung in unserem Programm ist demnach die Zeile mit dem 'printf'-Aufruf. Das heißt, die Funktion 'main' ruft ihrerseits gleich eine weitere Funktion auf. Diese Funktion wird jedoch nicht parameterlos aufgerufen (die Klammern sind ja keineswegs leer), sondern es wird eine Zeichenkette an die Funktion 'printf' übergeben. Die Funktion 'printf' hat in C die Aufgabe, Zeichenketten und Variablen in einer Form an das Ausgabegerät zu senden, in der sie durch den Benutzer erfaßt werden können (dies sind im Gegensatz zu der rechnerinternen Darstellung 'echte' Buchstaben und Ziffern).

Die Funktion 'printf' bietet einige Besonderheiten gegenüber den Druckanweisungen anderer Programmiersprachen wie z. B. Basic (print) oder Pascal (write bzw. writeln). Das erste Merkmal fällt gleich in unserem Beispiel auf. Es ist die Zeichenkombination ' n'. Sie steht für ein einzelnes, nicht sichtbares Zeichen. Solche Zeichen nennt man auch Steuercodes. Die Einleitung für die Darstellung eines solchen Zeichens ist der Schrägstrich ' '. Der Programmierer kann mit seiner Hilfe sogenannte Steuercodes in die Zeichenkette einfügen. Unter Steuercodes versteht man Zeichen, die vom Rechner als Anweisung verstanden werden. Solche Steuerzeichen sind z. B. ein Tabulatorsprung (Tab), das Zeichen für neue Zeile (Line Feed) oder der Wagenrücklaufbefehl (Return). In Ihrem Compilerhandbuch sind die Möglichkeiten, die Ihnen von Ihrem Compi-

ler geboten werden, aufgelistet. Experimentieren Sie doch einmal mit den verschiedenen Steuerzeichen (was passiert z. B. bei ' "' oder ' ' ?).

Wir lernen Rechnen

Natürlich sollen nicht Sie das Rechnen lernen, sondern Ihr Computer; und zwar in C. Zum Rechnen braucht man Variablen. In ihnen sollen das Ergebnis gespeichert oder auch nur Zwischenergebnisse gemerkt werden. Wir wollen nun damit beginnen, zwei Zahlen zu addieren. Auf dem Papier hätte eine solche Rechnung folgendes Aussehen: 1 + 2 = x (x ist hier der Variablenname für das Ergebnis unserer Rechnung). In der Spache Basic würde ein solches Programm so aussehen:

```
10 x=1+2
20 end
```

Dasselbe Prográmmchen in C hat dieses Format:

```
main()
{
   int x;
   x=1+2;
}
```

Die Bedeutung von 'main' und den beiden geschweiften Klammern { } kennen Sie ja bereits. Auch die vorletzte Zeile mit der Anweisung x= 1+2; ist auf den ersten Blick verständlich. Das einzig neue und für C typische in diesem kleinen Programm ist die dritte Zeile mit dem Inhalt 'int x;'. Diese Anweisung teilt dem Compiler mit, daß Sie eine Variable benutzen wollen. Im Gegensatz zu Basic, in dem Sie an jeder beliebigen Stelle im Programm eine neue Variable einführen können, müssen in der Sprache C alle verwendeten Variablen vor der Benutzung deklariert werden. Eine solche Deklaration hat einen bestimmten Aufbau. Am Anfang der Zeile steht das Schlüsselwort für den gewünschten Datentyp. In unserem Fall 'int' für Integer (Ganzzahl). Danach folgt eine Aufzählung aller Variablennamen, die diesem Typ entsprechen sollen. In unserem Beispiel enthält diese Liste nur einen Variablennamen. Es könnten jedoch auch beliebig mehr sein, jeweils durch ein Komma getrennt. Die Zeile 'int x,y, z;' würde demnach drei Variablen des Typs Integer deklarieren. Da eine Arithmetik, beschränkt auf ganze Zahlen, für eine Programmiersprache recht unbefriedigend wäre, kennt C noch andere Zahlentypen (allgemein Datentypen). Diese kennzeichnen sich durch entsprechende Schlüsselworte. Die folgende Aufstellung gibt Ihnen einen Überblick über die Wertebereiche der einzelnen Datentypen.

char

Dieser Typ ist in der Lage, ein Zeichen (z. B. 'A') zu speichern.

int

Der Typ 'int' ist Ihnen ja schon begegnet. Er kann ganzzahlige Werte aufnehmen. Viele C-Compiler unterscheiden den Datentyp 'int' noch einmal in 'short' und 'long'. Diese Unterscheidung betrifft die maximal darstellbare Zahl, die noch unter dem Datentyp abgebildet werden kann (Tabelle 1 enthält die Wertebereiche der wohl am meisten verbreiteten Compiler auf dem Amiga). Eine weitere Möglichkeit, den Gültigkeitsbereich zu beeinflussen, ist die Verwendung des Schlüsselwortes 'unsigned'. Dies erlaubt es dem Programmierer, die Zahl 'unsigned', also vorzeichenlos zu verwenden. Der Vorteil dieser Darstellung liegt in der um den Faktor 2 größeren, maximal möglichen, positiven Zahl; der Nachteil ist, daß keine negativen Zahlen mehr darstellbar sind.

float

Eine Variable des Typs 'float' kann einen Gleitkommawert einfacher Genauigkeit aufnehmen. Die Darstellung dieses Typs im Speicher des Rechners ist im Gegensatz zu den bisher genannten Datentypen stark von der Implementierung des verwendeten Compilers abhängig. Dies hat jedoch keinen Einfluß auf die Programmierung in C. Ich werde an geeigneter Stelle in diesem Kurs auf eventuell mögliche Probleme eingehen.

double

Dieser Typ ist der groe Bruder (oder Schwester?) des Typs float. Er kann doppelt genaue Gleitkommazahlen aufnehmen. Der darstellbare Zahlenbereich dieses Datentyps sowie die Anzahl der gültigen Stellen ist wiederum vom verwendeten Compiler abhängig. Sie sollten daher in Zweifelsfällen immer Ihr Compilerhandbuch zu Rate ziehen.

pointer

In einer Variablen des Typs 'pointer' kann der Programmierer einen Zeiger ablegen. Dieser Zeiger enthält dann die Adresse des Objektes, unter der Ihr Rechner dieses Objekt adressieren (ansprechen) kann. Die Art und Fähigkeit, mit diesem Datentyp umzugehen, macht C zu einer sehr leistungsfähigen Sprache.

Kommen wir an dieser Stelle noch einmal auf die Funktion 'printf' zurück. Im ersten Beispiel haben wir die einfachste Form kennengelernt, diese Funktion aufzurufen. Wie sieht es nun mit der Ausgabe von Variablen durch die Funktion 'printf' aus? Es ist ja wenig sinnvoll, mit einem Programm bestimmte Werte zu berechnen und diese dem Benutzer dann nicht mitteilen zu können. Die Möglichkeiten von 'printf' erstrecken sich auch auf dieses Gebiet. Das dritte Beispielprogramm zeigt Ihnen, wie man mit Hilfe dieser Funktion die einzelnen Datentypen in einer dem Benutzer verständlichen Form auf den Bildschirm bringen kann.

```
main()
{
   char Buchstabe;
   short Kleine_Zahl;
   int Zahl;
   long Große_Zahl;
   float Bruch_Zahl1;
   double Bruch_Zahl2;

/* Zuweisung der einzelnen Werte
*/
   Buchstabe = 'e';
   Kleine_Zahl = 12;
   Zahl = 23679;
   Große_Zahl = 123456789;
   Bruch_Zahl1 = 1.5
   Bruch_Zahl2 = 1.4142135;

/* Ausgabe der zugewiesenen Werte
*/
   printf("Der Inhalt von Buchstabe
   ist : %c n",Buchstabe);
   printf("Kleine_Zahl enthält den
   Wert : %4d n",Kleine_Zahl);
   printf("Die Variable Zahl ent-
```

```
    spricht : %d n",Zahl);
    printf(„Große_Zahl ist %l groß
    ! n",Große_Zahl);
    printf(„In der nächsten Zeile fol-
    gen die Brüche : n");
    printf(„%1.1f und %f n",
    Bruch_Zahl1,Bruch_Zahl2);

/★ Programmende ★/
}
```

In diesem Programm sind auch gleich ein paar Neuigkeiten für Sie untergebracht. Auffällig sind auf den ersten Blick sicher die Zeilen, die mit der Zeichenfolge '/★' beginnen und mit '★/' enden. In so gekennzeichneten Zeilen kann der Programmierer Kommentare zu seinen Programmtexten schreiben. Dies entspricht dem in Basic gebräuchlichen 'Rem'-Befehl. Alle Zeichen zwischen diesen beiden Zeichenfolgen werden bei der Übersetzung durch den Compiler überlesen und nicht mitübersetzt. Sie sollten sich angewöhnen, Ihre Programme schon während der Entwicklung zu kommentieren, da man sich diese Mühe nach der Fertigstellung kaum noch machen will. Doch auch der beste Programmierer hat nach einem halben Jahr den Überblick über die speziellen Eigenschaften einer bestimmten Funktion verloren. Auch erleichtert er sich oder anderen durch sinnvolle Kommentierung eine spätere Erweiterung des Programms.

In Zeile 21 begegnet uns zum ersten Mal eine Konstruktion, mit deren Hilfe es möglich ist, Variablen innerhalb einer Zeichenkette einzufügen. Die Funktion 'printf' benutzt hierzu eine Zeichenfolge, die mit dem Zeichen '%' beginnt. Das Prozentzeichen und die danach folgenden Formatierungsbefehle erlauben es, die Stelle und das Aussehen der Variable im umgebenden Text zu bestimmen. Ein kleiner Vergleich mit einer Basiczeile kann dies leicht verdeutlichen:

```
10 a = 10
20 print„Jetzt bist du „,a," Jahre alt"
```

Diese beiden Basiczeilen führen zu der Ausgabe des Textes:

```
Jetzt bist du 10 Jahre alt
```

Wie Sie sehen, wird die Variable direkt an der Stelle im Ausgabetext eingefügt, an der sie später gedruckt werden soll. Die Funktion 'printf' hingegen erwartet an der gewünschten Stelle lediglich einen Platzhalter. Eine entsprechende Zeile in C sieht somit wie folgt aus:

```
a = 10;
printf(„Jetzt bist du %d Jahre
alt n",a);
```

Die Bezeichnung der Variablen, die dann für den Platzhalter ausgegeben wird, steht in einer Aufzählung, die dem Ausgabestring folgt. Die Reihenfolge ist dabei die gleiche, in der auch die Platzhalter auftreten. Der erste Platzhalter steht also für die erste Variable in der Liste, der zweite für die zweite u.s.w. Die Kennzeichnung, welche Art von Variable in den Ausgabetext eingefügt werden soll, erfolgt durch das Zeichen nach '%'. Je nach Version des vorliegenden Compilers werden folgende Möglichkeiten unterstützt:

%d

Die entsprechende Zahl (vom Typ int) der Variablenliste wird in dezimaler Schreibweise ausgegeben.

%o

Die int Zahl wird zur Basis 8 ausgegeben (octal Zahlen).

%x

Die int Zahl wird in hexadezimaler Schreibweise ausgegeben.

%c

Die Variable wird als Buchstabe des Rechnerzeichensatzes ausgegeben. Der Zahlenwert der auszugebenden Variable entspricht dabei der Ordinalzahl des Buchstabens (z. B. Variableninhalt = 65, ausgegebener Buchstabe = 'A'). Die Variable ist vom Typ 'char'.

%s

Die Variable enthält einen Zeiger auf eine Zeichenkette. Der Inhalt der Zeichenkette wird ausgegeben, bis das Zeichen NULL erreicht wird. Der Compiler terminiert Zeichenketten, die mit einem " eingeschlossen sind, automatisch mit dem NULL-Zeichen.

%f

Eine Variable vom Typ 'float' oder 'double' wird in dezimaler Schreibweise ausgegeben.

%%

Das Zeichen % wird selbst ausgegeben.

Eine zusätzliche Möglichkeit ist die Integration von Formatierungszeichen zwischen das '%' Zeichen und die oben aufgeführten Möglichkeiten. Folgende Formatanweisungen sind möglich (in ihrer Reihenfolge nach dem % Zeichen geordnet):

-

Das Minuszeichen erzeugt eine linksbündige Ausgabe der Variable.

88

Die '88' steht hier für eine beliebige einzusetzende Zahl. Diese Zahl bestimmt die Größe des Ausgabefeldes in Zeichen. Ist die darzustellende Variable kleiner als die verlangte Zahl von Zeichen, werden die fehlenden Stellen aufgefüllt.

.88

Wird die nächste Möglichkeit verwendet, muß sie von der vorherigen mit einem '.' getrennt werden. Die Zahl nach dem Punkt gibt die Anzahl von auszugebenden Stellen nach dem Dezimalpunkt bzw. die Zahl der zu druckenden Zeichen bei einem String an.

l

Das Zeichen 'l' teilt der Funktion mit, daß eine Zahl nicht vom Typ 'int', sondern vom Typ 'long' ist.

Wie Sie sehen, ist die Funktion 'printf' ein vielseitiges Element in der Programmierung mit C, obwohl sie nicht im Umfang der eigentlichen Sprachdefinition enthalten ist. Sie bietet jedoch eine gute Möglichkeit, sich mit Ihrem C-System (Editor, Compiler, Linker und Libraries) bekanntzumachen. Nebenstehend sind zwei Stapeldateien (Batchfiles) abgedruckt, die Ihnen den Aufruf des Compilers und Linkers 'von Hand' abnehmen. Die beiden Versionen sind auf die Compiler von Manx (Aztec C) und Lattice (Lattice C) abgestimmt und können mit dem System-Editor ED eingebeben werden.

Am Ende dieser ersten Folge soll-

ten Sie über die Funktion 'main' sowie die Variablendeklaration von C Bescheid wissen, oder Sie sollten zumindest eine Anregung zum Studium weiterführender Literatur erhalten haben. Die nächste Folge beginnt dann mit der Programmierung von Schleifen und führt Sie auf diese Weise gleich ein Stück in die Besonderheiten von C ein. Bis dahin wünsche ich Ihnen viel Erfolg und Spaß mit dem bisher gelernten Stoff. Experimentieren Sie also, gewissermaßen als Hausaufgabe, mit 'printf'. Versuchen Sie doch einmal, die Zeile >Hallo „Fritz" wieviel % sind noch im Tank? Reicht es noch bis Frankfurt/Main?< auszudrucken!

(GC)

Für Aztec-Besitzer:

Außer den C-typischen Includedateien und Libraries sind drei Programme für den Einstieg in C wichtig. Es sind dies die Dateien mit den Namen cc, as und ln.

cc

Hinter dem Kürzel cc verbirgt sich der leistungsfähige Compiler der Firma Manx. Von ihm existieren bereits Implementierungen auf anderen Rechnersystemen (z.B. UNIX). Der Compiler bietet einige Optionen, die beim Compilieren vor den Quellname gestellt werden können:

cc [-OPTION] Quelltext.c

Unsere Kurzübersicht gibt Ihnen einen Überblick über die Möglichkeiten des Compilers (näheres erfahren Sie in Ihrem Handbuch):

-A

Der Compiler ruft normalerweise nach dem Compilieren den Assembler auf. Die Option -A unterbindet dies.

-DSymbol

Nach der Option -D kann eine Preprozessor-Definition stehen.
Beispiel:
#define Buffer 512
entspricht
-DBuffer=512

-IPfad

Die Option -I setzt den Suchpfad für Includedateien auf den durch 'Pfad' angegebenen Weg.

-O Datei

Diese Option setzt die Ausgabedatei auf 'Datei'.

-S

Diese Option unterdrückt die Ausgabe von Fehler- und Warnmeldungen.

-T

Diese Option arbeitet eng mit der Option -A zusammen. Sie bindet den C-Quelltext als Kommentar in das mit Hilfe der Option -A erzeugte Assemblerlisting ein.

-B

Der Compiler stoppt normalerweise nach jeder fünften Fehlermeldung und fragt, ob er weiternachen soll. Die Option -B verhindert dieses Vorgehen.

-EAnzahl

Die Tabelle, die zur Berechnung von Ausdrücken während der Compilation dient, ist 80 Einträge groß (=1120 Bytes). Die Anzahl der Einträge kann mit der Option -E jedoch auch von Hand eingestellt werden.

-LAnzahl

Diese Option erzeugt eine lokale Symboltabelle mit 'Anzahl' Einträgen.

-YAnzahl

Die Tabelle zur Verwaltung von 'case'-Abfragen erhält 'Anzahl' Einträge.

-ZAnzahl

Der Compiler verwaltet eine spezielle Tabelle zum Ablegen von Zeichenketten aus dem Quelltext. Diese Tabelle ist auf 2000 Zeichen voreingestellt. 'Anzahl' verändert die Größe dieser Tabelle.

+B

Verwende kein 'public.begin' im erzeugten Assemblercode.

+C

Verwende das 'large code'-Modell.

+D

Verwende das 'large data'-Modell.

+HDatei

Schreibe Symboltabelle nach 'Datei'.

+IDatei

Lese Symboltabelle von 'Datei'.

+L

Alle verwendeten int-Variablen und Konstanten sind 32 Bit groß. Voreingestellt ist 16 Bit (schnellere Programme).

+Q

Alle Zeichenketten-Konstanten werden im Datensegment gespeichert. Voreingestellt ist das Codesegment.

as

as ist der Assembler des Manx Systems. Er wird vom Compiler automatisch aufgerufen und ist in diesem Zusammenhang nicht von größerer Bedeutung für den 'reinen' C-Programmierer.

ln

Der Linker (Binder) dient zum Verknüpfen des erzeugten Objektcodes mit den Funktionen aus den Libraries (Büchereien). Folgende Optionen sind dabei verfügbar:

-O Datei

Schreibe das fertige Programm nach 'Datei'.

-LName

Suche in der Library 'Name.lib' nach verwendeten Modulen.

-F Datei

Lese Kommandozeile von 'Datei' ein.

-T

Erzeuge eine Symboltabelle mit ASCII-Zeichen. Diese Tabelle dient zum symbolischen Debuggen und kann mit einem Editor betrachtet werden.

-W

Erzeuge eine Symboltabelle, die vom 'Wack'-Debuger gelesen werden kann.

-V

Erzeuge ausführliche Meldungen.

+C

Mache Programm im Chipmemory lauffähig.

+F

Mache Programm im Fastmemory lauffähig.

Folgendes Batchfile vereinfacht den Aufruf des Aztec Compilers. Es erwartet folgende Eingabe:

1> execute mache quelle

Quelle ist dabei das zu compilierende Programm. Beachten Sie bitte, daß die Extention .c automatisch angehängt wird. Tippen Sie die Stapeldatei mit einem Editor ein und speichern Sie sie unter dem Namen 'mache' in Ihrem Arbeitsverzeichnis (auf der Original-Diskette ist dies SYS2:Examples).

```
.key quelle
if exists <quelle>.c
  cc <quelle>.c
 else
  echo „Quelldatei <quelle>.c
  nicht vorhanden."
  skip ende
endif
if exists <quelle>.o
  ln <quelle>.o -lm -lc
 else
  echo „Qbjektdatei <quelle>.o
  nicht vorhanden."
endif
lab ende
```

Für Lattice-Besitzer:

Die drei Hauptprogramme des Lattice-Paketes verbergen sich hinter den Namen lc, asm und blink.

lc

lc ist der für den Benutzer sichtbare Teil des Compilers. Er ruft die Programme lc1 und lc2 auf, die den eigentlichen Compiler enthalten.

lc1

lc1 ist der erste Teil des Compilers. Er erzeugt ein sogenanntes quad-File, das dann von lc2 weiterverarbeitet werden kann. Der Aufruf erfolgt folgendermaßen:

lc1 [–OPTION] Quelltext

Als Option kann hierbei einer oder auch mehrere der folgenden Buchstaben genannt werden.

–b

Die Adressierung von Daten erfolgt relativ zu einem Basisregister. Diese Adressierungsart spart Zeit.

–c

Diese Option setzt den Compilerstandard von ANSI auf eigene Wünsche um. Nach –c muß dazu eine Auswahl der folgenden Buchstaben folgen:

c
erlaubt geschachtelte Kommentare.

d
erlaubt das '$'-Zeichen innerhalb von Variablennamen.

m
erlaubt das Erzeugen von Zeichenkonstanten der Form 'ab'. (Nicht zu verwechseln mit „ab".)

s
läßt den Compiler gleichartige Konstanten auf einen Speicherplatz legen.

t
läßt den Compiler Fehlermeldungen für undefinierte Strukturelemente erzeugen.

u
wandelt alle 'char'-Deklarationen in 'unsigned char' um.

w
verhindert Warnmeldungen bei Funktionen, die entgegen ihrer 'int'-Definition keinen Wert über return() zurückgeben. Der ANSI-Standard verlangt in diesem Fall eine 'void'-Definition.

–d

Die Option –d aktiviert den Debugmodus. Es werden zusätzliche Informationen im Objektcode abgelegt (symbolisches Debuggen). Als zweite Möglichkeit kann mit dieser Option eine Preprozessor-Konstante definiert werden (siehe auch: 'Für Aztec-Benutzer').

–e

Die Option –e aktiviert spezielle Behandlung von asiatischen Zeichen (manche asiatische Zeichen werden durch zwei Bytes dargestellt). Diese Option ist hier nicht von Interesse (siehe evtl. Handbuch).

–f

Der Compiler soll das Motorola Fast Floating Point Format verwenden.

–iPfad

Der Suchpfad für Includedateien wird auf 'Pfad' gesetzt.

–I

Diese Option weist den Compiler an, alle Daten auf Langwortgrenze zu legen.

–n

Diese Option setzt die Anzahl der Zeichen, mit denen eine Variable erkannt wird, auf acht Zeichen; d. h. Rastport1 und Rastport2 werden nicht mehr unterschieden.

–oDatei

Die Ausgabe des erzeugten quadfiles soll nach 'Datei' erfolgen.

–p

Bei der Angabe dieser Option wird nur der Preprozessor angestartet. Die Ausgabe erfolgt in eine Datei, die denselben Namen wie die Quelldatei hat. Als Extention wird jedoch p. anstatt von .c angehängt.

–u

Diese Option macht Preprozessor-Definitionen ungültig.

Der zweite Teil des Compilers ist in der Datei lc2 enthalten. lc2 erwartet folgende Komandozeile:

lc2 [OPTION] quelle

Die Quelldatei muß dabei die Extention .q haben (wird von lc1 erzeugt). Folgende Optinen sind zugelassen:

–c

Das fertige Programm läuft im Chipmemory.

–h

Das fertige Programm läuft im Fastmemory.

–oDatei

Schreibe den Objektcode nach 'Datei'.

–r

Benutze relative Sprünge bei Funktionsaufrufen (schneller).

–s

Spezifiziere Segment-Name.

–v

Der Compiler überwacht normalerweise die Stackabläufe. Die Option –v schaltet diese Überwachung ab.

–y

Diese Option erzwingt das unbedingte Laden des Baseregisters.

Die Zusammenfassung von lc1 und lc2 durch lc erwartet meist andere Optionen. Konsultieren Sie im Zweifelsfall Ihr Handbuch.

asm

asm ist der Assembler des Lattice-Systems. Er ist zum jetzigen Zeitpunkt noch nicht von Bedeutung.

blink

Hinter blink verbirgt sich ein zu dem originalen Amiga-Linker alink kompatibler Linker. Die Geschwindigkeit wurde jedoch erhöht.

Die Zusammenfassung von lc1, lc2 und blink mit lc macht ein Batchfile für kleinere Anwendungen überflüssig. Der Aufruf des Compilers erfolgt durch die Kommandozeile:

1> lc –L quelle

(Beachten Sie das groß geschriebene 'L'!)

AMIGA ART MACHINE

Designer Construction Set

Amiga Art Machine nennt sich ein Team von Grafikern, Technikern und Trickfilmspezialisten. Sie haben sich zur Aufgabe gemacht, professionelle Videoclips und Hilfen für Einsteiger auf dem Amiga zu erstellen. Grafik, Musik und Animationsprogramme, die im IFF-Standart arbeiten, werden hierfür benutzt.

„Apokalypse" und *„Starbird"* sind zwei Videoclips, die zum Preis von je 98,– DM zu kaufen sind. Der Lieferumfang umfaßt neben der Clip Diskette mit dem entsprechenden Video noch eine zweite Diskette mit Grafiken im IFF-Format.

Weiterhin enthalten ist ein Anleitungsbuch, das das eigentlich Interessante ist. In diesem Kleinen, aber umfangreichen Buch werden Tips und Tricks für DVideo erklärt. Hier kann man hinter die Karten der Profis schauen und das Beste aus DVideo herausholen. Spot, Rotation und 3 D-Effekte werden neben vielem anderem beschrieben und erklärt. Um zu sehen, ob es sich lohnt, dieses Kapitel durchzuarbeiten, braucht man sich nur das Video anzuschauen. Jedes Ein- oder Ausblenden, jedes Rotieren oder Animieren, das im Videoclip zu bewundern ist, wird im Buch ausführlich bearbeitet.

Die Bilder, Frames und Musikstücke auf der zweiten Diskette, die übrigens in eigenen Clips verwendet werden dürfen, zeigen, was der Amiga im Bereich Grafik und Musik leisten kann. Bilder wie Fantasyschach, Fantasyback oder Weltallmond begeistern auch verwöhnte Amigafreaks. Alle Bilder und Musikstücke sind im IFF-Standard verfaßt und können mit allen IFF-kompatiblen Programmen bearbeitet werden.

Der Preis ist zwar etwas überteuert, aber man kann sich viel Zeit und Arbeit mit dieser Ansammlung von Insidertricks sparen.

Weiterhin gibt es von Amiga Art Machine Team eine Vielzahl von Produkten, die dem Laien diese neue Art der Kunst etwas näher bringen. Interessant erscheint hier auch die Amiga Art Machine Computermalschule, die anhand von Beispielen zeigt, wie Bilder mit Graphicraft, Aegis Images und DPaint entstehen. Für die Computermalschule sind drei Disketten mit den Titeln Fantasy, Landschaften und Animation erschienen. Jedes Programmpaket ist mit einem ausführlichen Handbuch ausgestattet und für den Preis von je 49,– DM erhältlich.

Zum professionellen Arbeiten mit D-Paint II gibt es von Walter Friedhuber ein gleichnamiges Buch zum Preis von 58,– DM. Die Diskette zum Buch kostet nochmals 25,– DM.

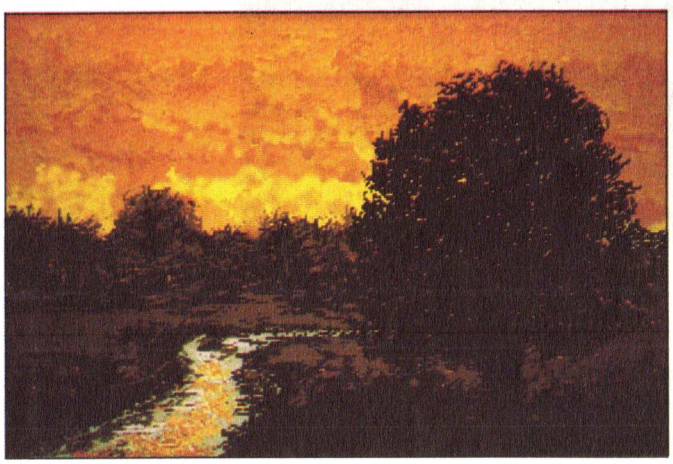
Zwei Sound-Disketten mit einer Vielzahl digitalisierter Geräusche im IFF-Format sind zum Preis von je 25,– DM ebenfalls erhältlich.

„The Best of" nennt sich eine Diskette die die besten Bilder des Amiga Art Machine Teams zeigt. Als Leckerbissen ist auf dieser Diskette noch ein Editor vorhanden, der es ermöglicht, Text, Sound und Grafik zu verbinden. Dieses File kann nun ganz normal vom CLI oder von der Workbench geladen werden. Der Preis für diese Diskette beträgt 48,– DM.

Bestellen können Sie all diese Softwarepakete bei folgender Adresse:

Firma G. Lechner
Roseggstraße 1
8000 München 60

Thomas Schlereth



BARBARIAN

In Thelston lebten einst die Zwillingsbrüder Thoron und Necron. Ihr Lehrer war ein alter Druide, der sehr schnell die unterschiedlichen Neigungen der Brüder erkannte. Thoron war tapfer und gütig, Necron hingegen skrupellos und gerissen. Beide entwickelten große magische Fähigkeiten. Am Ende ihrer Ausbildung erhielt Thoron ein glänzendes Schwert, Necron einen wundervollen Bogen. Der Druide ermahnte Necron, sich niemals gegen seinen Bruder zu wenden.

Die Brüder wurden erwachsen und gingen ihrer Wege. Thoron wurde ein bekannnter Jäger und durchstreifte das Land nach Abenteuern. Necron verschwand. Jahre später trafen sich die Brüder in einer Stadt, die von Necron ausgebeutet und tyrannisiert wurde. Necron fiel im Kampf, aber die Mächte des Bösen retteten seine dunkle Seele. So wurde er der Herrscher der Unterwelt von Durgan und begann, mit seinen finsteren Gestalten das Land zu geißeln.

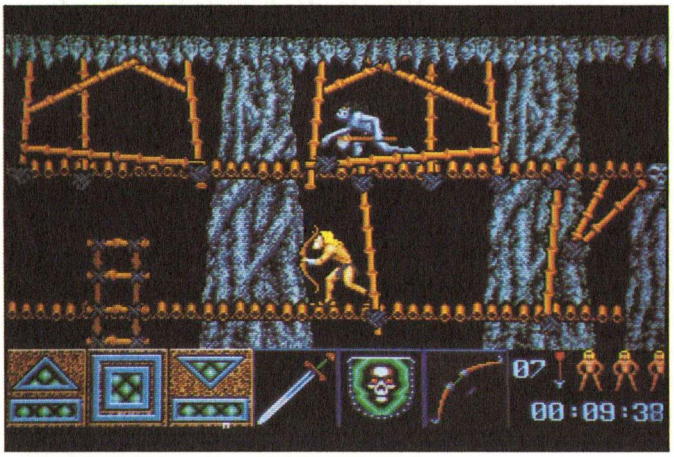
Indessen wurde Thoron ein Sohn geboren: Hegor. Bereits in jungen Jahren wurde er von seinem Vater im Kampf mit Schwert und Bogen ausgebildet. Tapfer, klug und abenteuerlustig war der junge Hegor, als er auszog, um der berühmteste Drachentöter aller Zeiten zu werden. Er durchstreifte das Land und war bald überall als tapferer Kämpfer bekannt. Necrons dunkle Horden hingegen brachten den König immer mehr in Bedrängnis. Der Drache Vulcuran tötete Thoron; schließlich wußte der Herrscher keinen anderen Ausweg mehr, als demjenigen, der der Zerstörung Einhalt böte und die bösen Mächte bezwänge, die Krone des Reiches und all seine Reichtümer zu versprechen. Hegor machte sich auf, den Tod seines Vater zu rächen. Ein weites Sumpfgebiet schützt den Eingang der Unterwelt von Durgan. Hegor zieht sein Schwert und schreitet voran...

Story, Szenario und Handlung des neuen Spiels von Psygnosis sind faszinierend. Wer einmal angefangen hat, sich mit Hegor durch die Unterwelt zu kämpfen, wird sehr schnell vom „Barbarian-Fieber" gepackt. Nicht immer ist es ratsam, einen Gegner anzugreifen. An manchen kann man besser vorbei- oder drüberspringen. So ist der Ritter im 16. Screen unbesiegbar, der hinter im liegende Bogen nur eine Atrappe. Selbst am Drachen kann man mit dem richtigen Timing vorbeispringen.

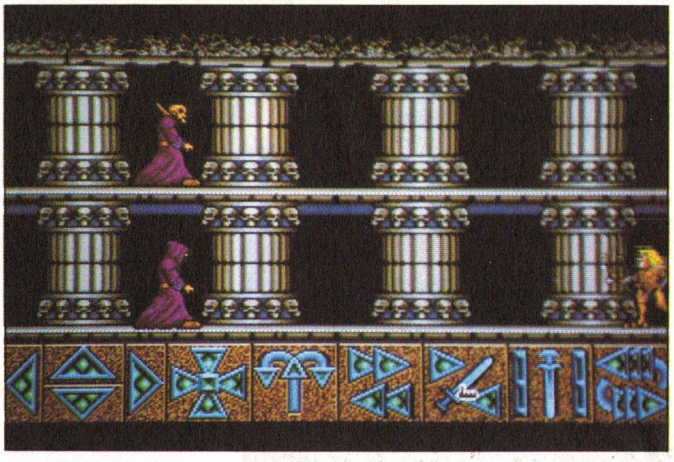
In der Endphase läuft die Zeit rückwärts, die Uhr zeigt dann an, wann der Vulkan ausbricht und die Spitze des Berges explodiert.

Wenn Sie sich jetzt von Durgangs Unterwelt genauso angezogen fühlen wie Hegor, schreiben Sie uns doch einfach eine Postkarte. Wir verlosen unter den Einsendern fünf Originale von „BARBARIAN" für den Amiga. Schicken Sie uns die Postkarte bis zum 30. August 1987 (Datum des Poststempels).

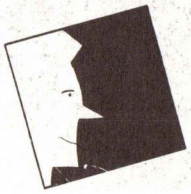
Es gilt das Datum des Poststempels. Der Rechtsweg ist ausgeschlossen.



Seltsame Wesen kreuzen den Weg des Helden



Eine der schwierigsten Etappen: die Mönche und ihre Feuerkugeln



IDEE
SOUND
GRAFIK

GARRISON

Digital Dungeons

Länger als sechs Monate hat Andreas Hommel von Digital Dreams an der Umsetzung des Spielhallenhits „Gauntlet“ für den Amiga gearbeitet. Das Spiel heißt „Garrison“ und unterscheidet sich mit Absicht deutlich von der Automatenversion. Alle Spielebenen sind vollkommen neu aufgebaut. Zu den vom Automaten bekannten Gegnern sind noch weitere hinzugekommen. Der Spieler kann unter fünf verschiedenen Spielfiguren auswählen: Merlin the Wizard, Agor the Warrior, Golwyn the Elf, Valeria the Valkyrie und Thorin the Dwarf. Bis zu fünf Spieler können an der Schlacht gegen steinwerfende Zwerge, blutsaugende Zauberer und keulenschwingende Monster teilnehmen. Dabei können jedoch maximal zwei Spieler gleichzeitig im jeweiligen Level spielen. Stirbt ein Spieler, kann mit den Funktionstasten ein weiterer in dasselbe Level dazugeholt werden. Die Sprites der Spielfiguren sind sehr schön dargestellt, die Liebe zum Detail ist bestechend. Sound und Animation nutzen die Möglichkeiten, die die der Amiga bietet. Die Amiga-Besitzer ohne Speichererweiterung oder die Besitzer eines Amiga 1000 kommen nicht in den Genuß der verschiedenen Sprites mit denen Merlin, Agor, Golwyn, Valeria und Thorin dargestellt werden. Der Speicherplatz reicht nur aus, um für alle Spieler gleich aussehende Sprites zu erzeugen. Alles andere ist aber in der „abgespeckten“ Version gleich geblieben. Die Level eins bis vier sind immer die gleichen, ab dem fünften Level entscheidet der Zufall, in welchem der 128 Irrgärten sich die Spieler wiederfinden. Auf diese Weise wird auch der etwas routiniertere Kämpfer immer wieder vor neue Probleme gestellt. Graphik und Animation sind sehr gut gelungen. Der digitalisierte Sound ist wirklich hörenswert. Die Amiga-Besitzer können sich ab sofort den Gang in die Spielhallen wieder sparen – das Problem der Gestaltung so manches verregneten Spätsommernachmittages ist ab sofort gelöst.

(cpl)

Bild 1: Gespenster und Monster sollte man sich möglichst vom Halse halten.

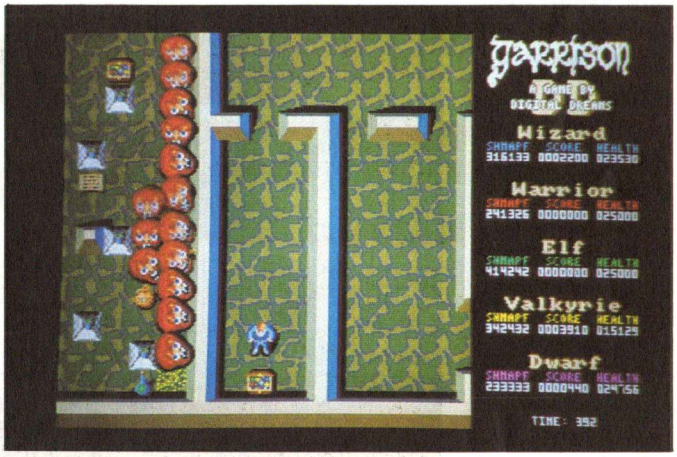
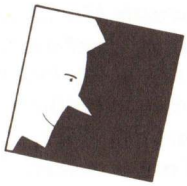


Bild 2: Das Einsammeln von den verschiedenen Gegenständen bringt Überraschungen.

IDEE
SOUND
GRAFIK



THE GUILD OF THIEVES

„Eigentum ist Diebstahl"

„Eigentum ist Diebstahl": Nach diesem Motto ist die Gilde der Diebe – korrekt ausgedrückt: The Worshipful And Partially Hnourable Guild of Professional Nocturnal and Surreptitious Entry and Removal Operatives Of Kerovnia – zu einer der mächtigsten Organisationen im Lande Kerovnia aufgestiegen. Mit zwiespältigen Gefühlen sieht die Gilde allerdings die in allen Teilen der Bevölkerung wachsende Anerkennung ihrer Grundsätze. Wie sollen die Meister dieses Faches ihren Lebensunterhalt sichern, wenn sich selbst der Durchschnittsbürger mehr und mehr zum Spitzendieb entwickelt? Wie sollen die Mitglieder der Gilde die in Kerovnia üblichen Bestechungsgelder zusammenklauben, wenn zwar die Mehrheit nach dem Prinzip der Gilde lebt, aber keinen Beitrag an die Gilde entrichtet? Wahrlich schwere Zeiten, denen diese ehrenwerte Gilde entgegensieht. Deshalb sucht sie nach einem geeigneten Mann, der der Gilde zu ihrer alten Macht zurückführt, die sie in den glorreichen Tagen ihrer Gründung durch den Lord-Oberrichter Trushwhacker innehatte. Der hatte nämlich seinerzeit die Gilde in einer demokratischen Sitzung – one man, one vote – gegründet. Daß bei der Abstimmung die einzig zuglassene Stimme seine eigene war, berührt das demokratische Prinzip dieser historischen Tat kaum.

Die Gründe, die zur Einrichtung der Gilde führten, waren ebenso logisch wie (besonders in den Augen des Lord-Oberrichters) verständlich, wurde er doch schließlich wegen jedes einfachen überfalls oder unbedeutenden Meuchelmordes bei so wichtigen Geschäften wie Golf, Roulette oder „very important one-to-one business situations with my personal assistant, FiFi La Touche" gestört. Fortan waren alle Aktivitäten der

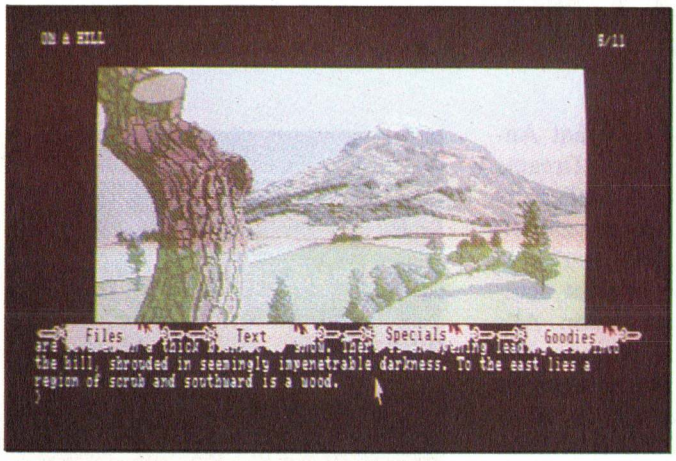


Bild 1:
Die einmalige Einschaltgrafik von the Guild of Thieves.

chen Lebens mehr und mehr verloren.

Eine ganz neue Form der Prüfung Mitglieder der Gilde gesellschaftlich anerkannt und gesetzlich geschützt. Nach dem Tode des ehrenwerten Trushwhacker gewannen die Regeln der Gilde auch in der gemeinen Bevölkerung immer mehr Freunde. Dabei ging der Führungsanspruch der Gilde in diesen Punkten des öffentlisoll unter den Kandidaten denjenigen herausfinden, der geeignet erscheint, die Gilde zu ihrer alten Größe zurückzuführen. Springen Sie aus dem Boot der Gilde mitten hinein in das Abenteuer Ihres Lebens – eines Lebens, das mitunter recht kurz sein kann, wenn Sie auch nur einen falschen Schritt tun.

(cpl)

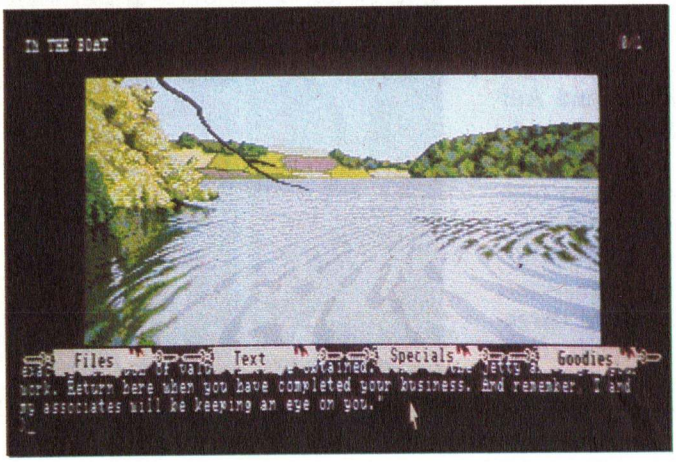
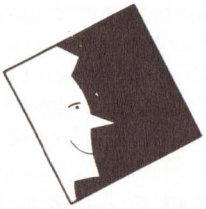
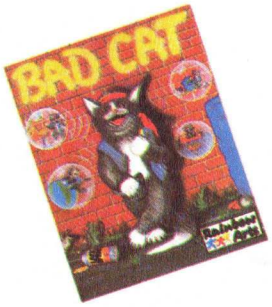


Bild 2:
Phantastische Landschaften begleiten einem auf dem weiten Weg



IDEE
SOUND
GRAFIK



BAD CAT

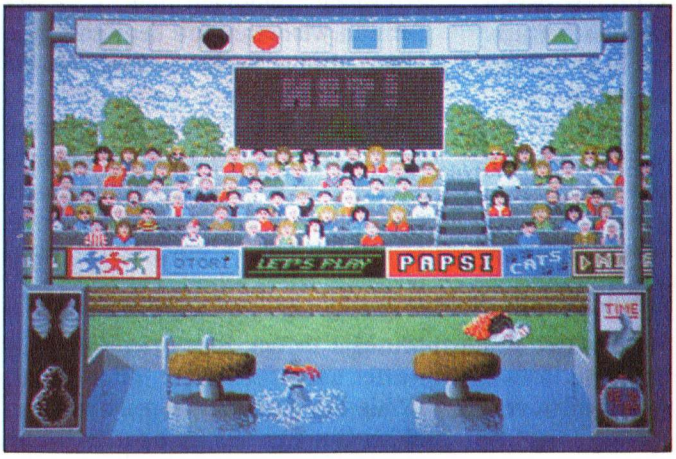
Katzenolympiade

Los Angeles 1984: Die Stadt rüstet sich für die Olympischen Sommerspiele. Große Leute halten ebensogroße Reden. Alles ist bereit für die Besuchermassen. Am Rande dieses Happenings finden sich auch die streunenden Großstadtkatzen ein, um ihre eigene Olympiade zu veranstalten. Sie treffen sich überall in der Stadt, um die ausgefallensten Wettbewerbe auszutragen. Der Favorit ist BAD CAT – wird er sich seines Namens würdig zeigen? Für Katzen ist schon das Erreichen der Schauplätze ein nicht zu unterschätzendes Problem. In rasanten Verfolgungsjagden, stets auf der Flucht vor Hunden, muß zunächst der große Park erreicht werden. Weiter geht es dann in der übelsten Spelunke der Stadt. Beim Kegeln trifft BAD CAT auf die dicke Bulldoge aus dem Westteil der Stadt. Wird er von einer Kugel getroffen, muß er ein großes Bier austrinken. In mehr oder minder ramponiertem Zustand geht es danach ins Kanalsystem von Los Angeles. Geschicklichkeit und Mut sind hier besonders gefragt. Viele angsteinflößende und gefährliche Tiere führen im Kanal ein finsteres Dasein. Endlich im Stadion angelangt, ist sportliche Akrobatik Trumpf: Absprung, Salto und butterweiche Landung – oder ein Sturz ins kalte Wasser, das ist die Frage. Graphisch hebt sich BAD CAT angenehm vom Durchschnitt ab, auch die Titelmusik ist sehr ansprechend. Besonders gut haben uns die vielen, ganz unterschiedlichen Spielsituationen gefallen, in die BAD CAT gerät. Zu der auch den Amiga schon überschwemmenden Welle von Ballerspielen setzt BAD CAT einen angenehmen Kontrapunkt – Geschicklichkeit ist Trumpf.

Wir erhielten BAD CAT erst nach Redaktionsschluß. Deshalb mußten wir ein wenig zaubern, um den Bericht noch in dieser Ausgabe unterzubringen. Wir konnten das Spiel also nicht einem intensiven Test unterziehen, wie wir es normalerweise getan hätten. Unser Bericht hat deshalb eher beschreibenden Charakter und will Ihnen eine Produkt-Information liefern. In der nächsten Ausgabe werden wir zwanzig von Rainbow Arts gestiftete Originale von BAD CAT für den Amiga verlosen. Als Sonderpreis gibt es außerdem einen einwöchigen Besuch bei Rainbow Arts.

cpl

Joystickakrobaten sind gefragt.



Überall befindet sich das für Katzen abscheuliche Naß.

IDEE
SOUND
GRAFIK

THE FAERY TALE

Wir bieten die Lösung

Schneller als erwartet ist im Lande der Feen und Zauberer wieder Frieden eingekehrt. Wie wir in unserer letzten Ausgabe berichteten, hatte sich Julian von Tambry aufgemacht, den Talisman zu retten und die Hand der Prinzessin zu gewinnen. Es ist ihm gelungen. All denen, die noch tapfer mit den Mächten der Finsternis ringen, ist sein kurzer Bericht gewidmet:

Nach mehreren Anläufen fand ich im Südosten von Tambry eine Truhe mit einem blauen Stein. Er war sehr wichtig, denn die Höhle des Drachen ist vom östlichen Steinkreis aus nicht direkt zu erreichen. Der Steinkreis im Osten von Tambry ist das erste Ziel jedes Abenteurers. Wer ihn zuerst erreicht, findet dort 50 Goldstücke, einen goldenen Ring und einen blauen Stein. Der erste Teleportsprung (nach Westen) führt zur Zauberinsel. Dort, im blauen Schloss der guten Fee, erhält der Reisende eine goldene Statue – und bis zu 65 Luck-Punkte, wenn er die Fee lange genug danach fragt. Mit dem zweiten blauen Stein geht es weiter – nach Nordwesten zur Drachenhöhle. Verborgen im ewigen Eis hütet dort der Drache die stärkste Waffe der Zauberwelt – das Magic Wand. Der Rückweg nach Tambry wird dann entweder mit einem weiteren blauen Stein (Norden) oder zu Fuss angetreten. Wer nicht gerne marschiert, findet am Westufer des Lake of Ocean ein Floß, mit dessen Hilfe sich die Reisezeit erheblich verkürzen läßt.

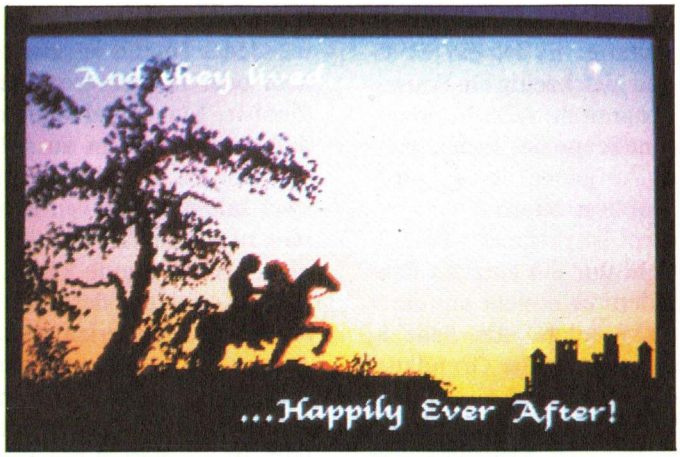
Ausgeschlafen, satt und mit einer Anzahl von Bird-Totems ausgerüstet, beginnt der zweite Abschnitt des Abenteuers. Diesmal ist das Ziel der Watch Tower. In seinem Innern liegt die magische Muschel, mit deren Hilfe die Schildkröte herbeigerufen werden kann. Auf dem Weg dahin habe ich Seine Majestät in Marheim besucht. Er klagte darüber, daß ein böser Zauberer seine schöne Tochter in ein unzugängliches Verlies in die Berge südlich der Hauptstadt verschleppt habe. Ich beschloß, sie zu retten. Das Verlies in den Bergen kann man jedoch nur mit dem goldenen Schwan erreichen. Bändigen läßt sich der Schwan wiederum mit dem goldenen Lasso, das im Besitz der Hexe ist. Sie lebt inmitten eines Labyrinths aus Sträuchern und Höhlen im Zentrum des Grimwoods. In diesem Labyrinth ist auch eine Feuerstelle mit einem Goldschatz und einer Statue verborgen. Doch halt – ein bißchen komplizierter ist es schon.

Gehen wir also nach Erwerb der Muschel zurück zum Friedhof. Dort wartet um Mitternacht der ruhelose König der Träume. Ihm ist von den üblen Gesellen aus Hembath's Tomb ein Knochen gestohlen worden. Er kann deshalb nicht mehr schlafen. Für die Wiederbeschaffung des Knochens verspricht er Hilfe beim Erwerb des Sonnensteines, der in einer Gruft ganz in der Nähe des Verlieses der Prinzessin liegt. In Hembaths unterirdischer Höhle findet sich neben dem Knochen auch eine weitere Statue. So habe ich also dem Lord der Träume seine innere Ruhe wiedergegeben, den Sonnenstein geholt und schließlich der Hexe das goldene Lasso abgenommen. Die Reise von dort zur Schwaneninsel kann man dank zweier blauer Steine erheblich abkürzen. Der Steinkreis nördlich von Tambry befördert jeden mittels eines Doppelsprunges (zuerst nach Westen, dann nach Norden) in die Nähe der Insel. Die freundliche Schildkröte stellt gerne ihren Rücken für den Rest des Weges zur Verfügung. Auf dem Rücken des goldenen Vogels ist man nicht nur sicher vor Angriffen, man reist auch mit der Geschwindigkeit des Windes. So war es nur noch ein kurzer Weg bis zum Verlies der Prinzessin und zu ihrer Befreiung.

Die Dankbarkeit des Königs bringt nicht nur die letzte der goldenen Statuen, sondern auch das Gold, um genügend Bird-Totems zu erwerben und das Abenteuer zu vollenden. Gut gerüstet erreichte ich in der Nacht AZAL, die Stadt in der Wüste. In einem ihrer Häuser ist die Rose verborgen, die Lava versteinern läßt. Der

letzte Weg führt in die Plain of Grief. Ganau gegenüber den Ash Mountains liegt das Castle of Doom – der Eingang zur Unterwelt. Dort habe ich auch den Talisman gewonnen und bin anschliessend zurückgekehrt – „and they lived happily ever after". Dies ist der Bericht meines Abenteuers im Lande Holm. Wie Ihr sicher verstehen könnt, haben meine Gemahlin und ich uns für die Flitterwochen ein wenig zurückgezogen. Alle Fragen, die Ihr noch habt, kann Euch mein treuer Weggefährte beantworten, der auch meine Geschichte für Euch niedergeschrieben hat.

Julian von Tambry
(cpl)



Schreiben Sie uns einfach eine Postkarte. Wir verlosen unter den Einsendern fünf Originale von „BARBARIAN" für den Amiga. Schicken Sie uns die Postkarte bis zum 30. August 1987 (Datum des Poststempels).

Es gilt das Datum des Poststempels. Der Rechtsweg ist ausgeschlossen.

Anpassung des Fractal-Listings an den Aztec-Compiler, Version 3.2a

FRACTALS

In der vergangenen KICKSTART-Ausgabe war ein Listing abgedruckt, das fraktale Landschaften erzeugt. Da in unserer Compiler-Version (3.4a) keine Fehler auftraten, glaubten wir, ein einwandfreies Listing zu veröffentlichen. Viele Amiga-Anwender besitzen jedoch nur den Aztec-C-Compiler mit der Versionsnummer 3.2a, der im Assemblerteil einige Fehlermeldungen produziert. Nun reichen wir eine Version des Listings nach, die mit dem Compiler 3.2a ohne Störungen arbeitet. Im Assemblerteil müssen einige kleine Änderungen vorgenommen werden, und das Programm wird ohne Schwierigkeiten compiliert.

Diese Änderungen betreffen folgende Zeilen des Listings:

Zeile 363 entfällt
Zeile 364 entfällt
Zeile 365 anstatt xdef __rnd
jetzt public __rnd
Zeile 390 entfällt
Zeile 392 entfällt

Viel Spaß mit dem nun hoffentlich einwandfrei funktionierenden Programm!

Im Kurztest: SOUND SCAPE, ein integriertes Musikprogramm

Soundscape ist anders als andere Musikprogramme. Es ist ein Midisequencer und gleichzeitig ein Sound-Sampling-Programm: Alles in einem. Wie gut SoundScape mit seinen weitgefächerten Aufgaben fertig wird, soll unser Kurztest zeigen.

Das Konzept ist raffiniert: SoundScape ist nicht nur ein einziges Programm, sondern es besteht aus einer Reihe von Modulen, die beliebig kombinierbar sind. In der Grundausstattung besteht das Paket aus dem Midi-Sequencer sowie einem Modul zum Bearbeiten und Abspielen von Samples, also digitalisierten Klängen. Wer mehr über dieses Verfahren wissen möchte, sei auf die vorangegangene Ausgabe dieser Zeitschrift verwiesen, in der sich ein ausführlicher Grundlagenartikel über dieses Thema findet. Das Sample-Bearbeitungs-Modul entspricht der Software des SoundScape Sound-Samplers, der ebenfalls im vergangenen Heft getestet wurde. Lediglich die Hardware, die benötigt wird, wenn man selbst Klänge digitalisieren möchte, fehlt. Es ist jedoch möglich, dieses Modul nachträglich in SoundScape zu integrieren. Überhaupt ist dies die hervorstechende Eigenschaft der Modulkonzeption: Jeder Anwender kann, ob Hardware oder Software, sich genau diejenigen Erweiterungen des Grundprogrammes kaufen, die er für seine Anwendung benötigt. Im Augenblick sind allerdings außer dem Sampler keine weiteren Module verfügbar.

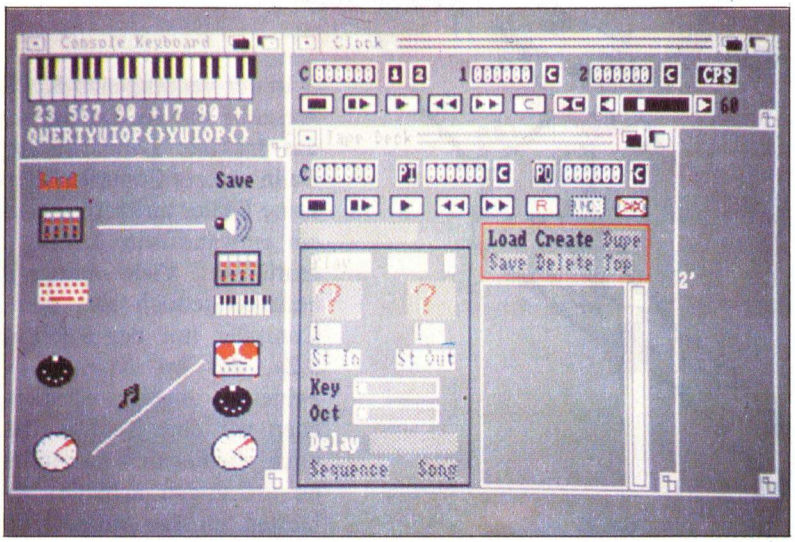
Wichtigstes Merkmal von SoundScape ist eine ungeheure Flut von Fenstern. Da das Paket aus Einzelprogrammen besteht, wurde auf eine Steuerung über Menüleisten verzichtet. Jedes Modul ist komplett in einem oder in mehreren Fenstern bedienbar. Manchmal führt dies dazu, daß der Bildschirm aussieht wie ein Schreibtisch, der die Papierstapel von zwei Jahren intensiven Programmierens zu tragen hat, was der Übersicht nicht gerade zugute kommt.

Die Koppelung der Einzelteile wird in einem Fenster erledigt, das sich, in Anlehnung an die professionelle Studiotechnik, 'Patch Panel' nennt. Ein 'Patch Panel', zu deutsch etwa Steckfeld, ist ein Kasten mit vielen Steckbuchsen, die man über kurze Kabel verbinden kann. Es wird verwendet, um verschiedene Geräte miteinander zu koppeln. Bei SoundScape hat das Panel genau die gleiche Aufgabe; die Geräte sind hier eben Software-Module. Und statt Verbindungskabel in Buchsen zu stecken, muß man beim Software-Panel die gewünschten Teile, ganz einfach, nur mit der Maus verbinden.

Es gibt Module für Ein- und Ausgabe. Die Eingabemodule erhalten Eingaben von außen und senden diese Daten an die Ausgabe- oder Empfängermodule, an die sie mit dem Patch Panel angeschlossen sind. Im Augenblick gibt es für die Eingabe folgende fünf Module:

1. Das Diskettenmodul 'Load'. Es dient dazu, ganze Stücke, Patches, Sounds, also den gesamten Status des Programms, von Diskette zu laden. Selbstverständlich gibt es auch einen Speicherbefehl, der diese so genannten 'Environments' (dt. Umgebungen, im Sinne von Arbeitsumgebung) auf Diskette zu speichern.

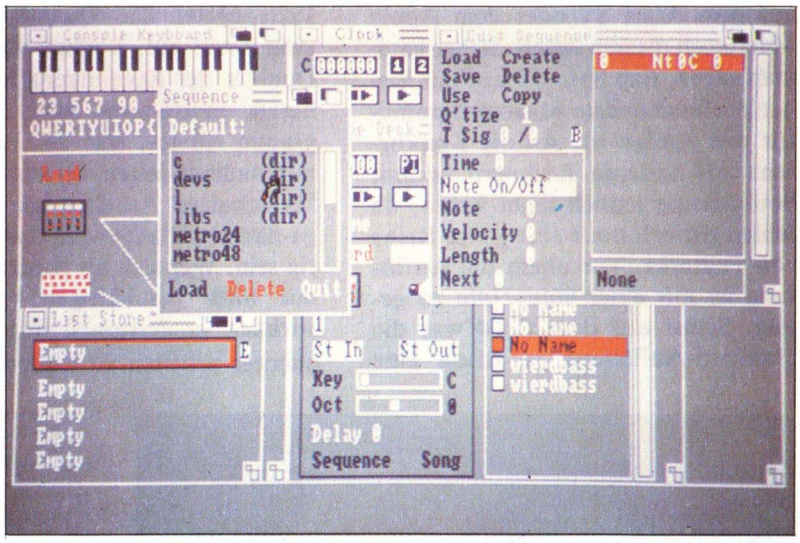
2. 'Console Keyboard' nennt sich ein Modul, das bestimmte Tasten auf der Amiga-Tastatur in Midi-Daten umsetzt. Auf den oberen zwei Reihen der Tastatur wird eine Zweieinhalb-Oktaven-Klaviatur simuliert.



3. Das Midi-Input-Modul empfängt Daten von externen Midi-Instrumenten.
4. Ein Uhren-Symbol steht für das Modul, das allen angeschlossenen Modulen Timing-Daten liefert. Diese Timing-Daten benötigt z. B. ein angeschloßener Midi-Sequencer für die Tempoeinstellung.
5. Schließlich steht noch ein Midi-Mixer zur Verfügung, der Informationen, die auf gleichen Midi-Kanälen ankommen, zusammenmischen kann.

Für die Ausgabe stehen im Moment 7 Module zur Verfügung:

1. Ein 'Save'-Kommando, das, wie bereits erwähnt, das Gegenstück zum Load-Befehl auf der Eingabeseite darstellt.
2. Das Lautsprecher-Icon meint genau das, was es zeigt: Es setzt ankommende Midi-Daten, die zum Beispiel vom Console-Keyboard oder dem Midi-In-Modul kommen können, in hörbare Klänge um, die über den Amiga-Stereoausgang ausgegeben werden. Dazu läßt sich jedem der 16 Midi-Kanäle ein eigener digitalisierter Klang zuordnen – vorausgesetzt, Ihr Amiga ist ein Speicherriese. Leider brauchen diese Samples ungeheuer viel Speicherplatz. Natürlich dürfen Sie nie versuchen, mehr als vier Töne gleichzeitig über den Lautsprecher auszugeben; auch SoundScape kann dem Amiga nicht mehr Tongeneratoren verschaffen, als er besitzt.
3. Das Midi-Out-Modul gibt ankommende Midi-Daten, vom Midi-In-Modul oder dem Midi-Mixer zum Beispiel, einfach kommentarlos an die Midi-Out-Buchse Ihres Midi-Interfaces aus.
4. Das Tape-Recorder-Modul. Dies ist der in SoundScape integrierte Midi-Sequencer, das eigentliche Herzstück von SoundScape also. Dieser Recorder funktioniert wie ein richtiges Tonbandgerät: Alle ankommenden Midi-Daten werden im Speicher des Rechners (statt auf Magnetband wie bei einem richtigen Recorder) aufgezeichnet und können jederzeit wieder ausgegeben werden. Zusätzlich besitzt das Modul, wie es sich für einen Sequencer gehört, noch einige Möglichkeiten zur Bearbeitung des Aufgenommenen. Doch dazu später mehr.
5. Der sogenannte 'Piano Player' ist ein Fenster, in das eine Klaviatur gezeichnet ist. Dort leuchten nun alle die Tasten auf, die man auf einem Midi-Keyboard spielt. Das ist mehr ein Gag.
6. Der Midi-Mixer ist auch auf der Empfängerseite vertreten; schließlich müssen die Informationen, die aus dem Mixer-Modul der Sender-Seite herauskommen, ja auch irgendwie dort hineingelangen...
7. Auch die Uhr ist doppelt vertreten, weil sie extern steuerbar ist. Man kann sie z. B. mit dem Midi-Eingang verbinden, um die Geschwindigkeit des Tape Recorders von einem extern angeschlossenen Gerät steuern zu lassen.



Damit wären die wichtigsten Möglichkeiten von SoundScape schon erklärt: Um irgendwelche Musik zu machen, wählt man irgendein Eingangsmodul und verbindet es mit dem gewünschten Ausgabegerät.

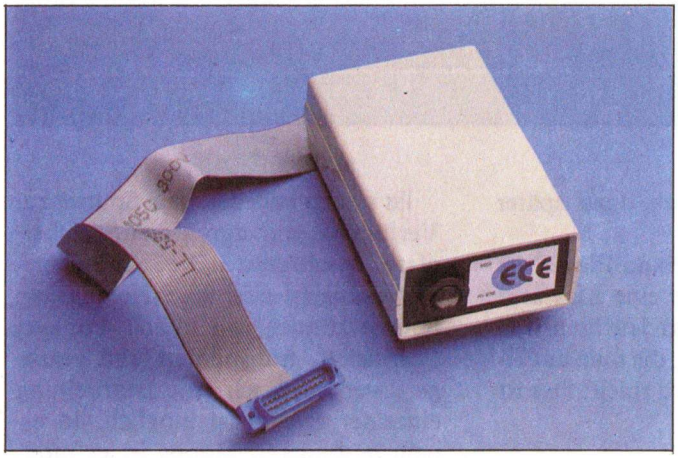
Ein paar Beispiele zur Erläuterung: Sie möchten die Samples des Amiga von einer richtigen Klaviatur aus spielen. Also: Midi-Eingang mit dem Lautsprecher-Symbol verbinden, die richtigen Klänge laden und spielen. Oder Sie möchten etwas aufnehmen, haben aber im Augenblick leider keine midifähige Tastatur. Console Keyboard mit dem Tape Recorder verbinden, der Tape Recorder ist seinerseits automatisch mit dem Lautsprecher und dem Midi-Ausgang verbunden.

Es stehen beliebig viele Spuren zur Verfügung, nur durch den Speicherplatz beschränkt. Die Aufnahme-Auflösung ist eine 96tel-Note. Diese Spuren können, wie üblich, kopiert, quantisiert, transponiert und verzögert werden, auch eine Bearbeitung einzelner Events ist möglich. In einem besonderen Modus ist es möglich, Sequencen zu bilden und diese zu einem Song zu verketten. Einzelne Sequencen können selbstverständlich ebenfalls abgespeichert werden. Für die Aufnahme gibt es zwei Autolocator-Punkte, wodurch automatisches Punch In/Out möglich wird, ebenso wie Step Time Recording. Die Bedienung ist für meinen Geschmack nicht übermäßig komfortabel. Die Fenstervielfalt macht es recht schwierig, den Überblick über eine laufende Produktion zu behalten. Was wirklich hübsch ist, ist die Möglichkeit, die Tonerzeugung des Amiga in Musikstücke einzubeziehen. Die Qualität ist zwar nicht gerade professionell, aber man kann recht gut arbeiten und vor allem schnell einmal etwas ausprobieren, ohne viele Vorbereitungen treffen zu müssen. Allerdings muß denen, die SoundScape richtig nutzen wollen, dringend ein möglichst großer Speicher empfohlen werden. Auf einem 512KByte-Amiga kann man mit Samples praktisch nicht arbeiten.

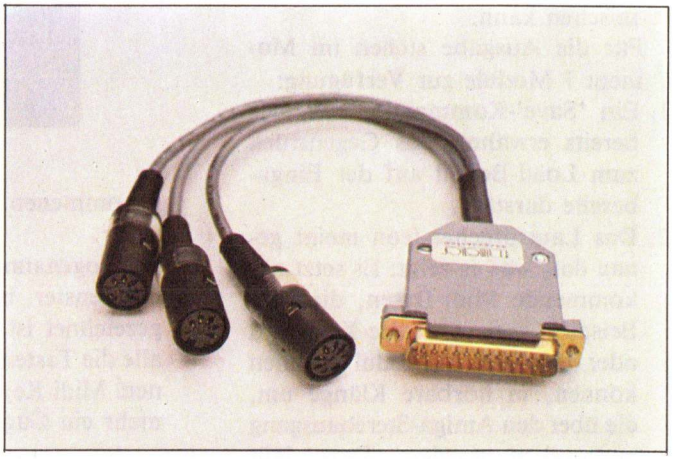
Die wichtigste und schwierigste Frage für Musiker ist freilich: Reichen die Fähigkeiten des Sequencers für professionelle Musikproduktionen aus? Der Sequencer muß sich immerhin mit professionellen Produkten

auf dem Atari ST oder dem Apple MacIntosh messen lassen. Sicher kann man sagen, daß SoundScape das bisher professionellste Musikprogramm für den Amiga ist. An die Konkurrenz auf anderen Rechnern kommt SoundScape jedoch nicht heran, da fehlen einfach noch eine ganze Reihe von Features. Vor allem die Auflösung ist mit einer 96tel-Note zu gering. Selbst auf dem C 64 war die Standard-Auflösung ja schon eine 192tel-Note! Die Quantisierung ist zwar recht gut, kommt aber an die Qualität der Konkurrenz noch nicht heran. Stattdessen leistet das Programm einiges, was der Profi wohl nur äußerst selten verwenden wird. 'Ernsthaftes' Arbeiten mit den Amiga-Samples dürfte wohl etwas schwierig sein. Wer sich als Amiga-Besitzer mit Midi beschäftigen will, ohne wirklich professionell mit dem Sequencer arbeiten zu wollen, für den dürfte SoundScape – besonders im Zusammenspiel mit dem Sampler-Modul, das auch interessante Experimente mit Klängen für vergleichsweise wenig Geld erlaubt – eine ausgezeichnete Wahl sein. Auf jeden Fall kann man mit dem Programm viel Spaß haben. Und wer weiß, welche Module die Hersteller noch ausbrüten?

Bezugsquelle: PDC (345,– DM)



ECE-MIDI-Interface stellt die Verbindung zwischen dem AMIGA und einer MIDI-kompatiblen Anlage her. Der Anschluß erfolgt beim AMIGA am seriellen Port (RS-232). Als Ausgang werden drei DIN-Buchsen (In.Thru.Out) angeboten. Der serielle Port des AMIGA ist durchgeschleift und mit einem Schalter versehen. Dadurch kann auch weiterhin ein serieller Drucker oder ein Modem betrieben werden. (PDC, Bad Homburg, DM 140,–)

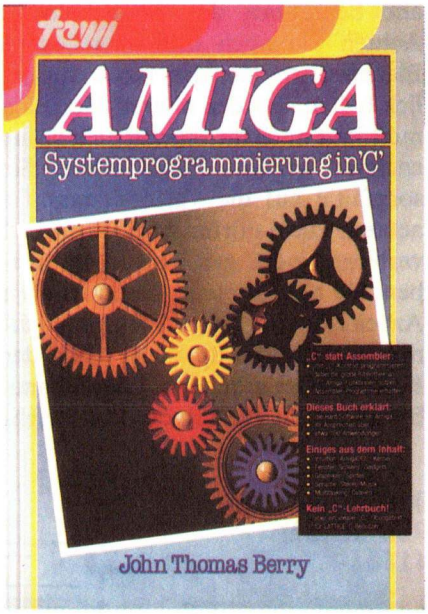


Auch das MIMETICS-Interface verbindet den seriellen Port des AMIGA mit einer MIDI-kompatiblen Anlage. Die Ausführung ist allerdings einfacher, der serielle Port ist nicht durchgeschleift. (PDC, Bad Homburg, DM 115,–)

AMIGA Systemprogrammierung in 'C'

Deutschsprachige Literatur, die sich speziell mit der Systemprogrammierung des AMIGA befaßt, ist zur Zeit leider noch recht dünn gesät. Umso erfreulicher ist, daß im Hause 'tewi' mitlerweile ein Buch erschienen ist, das diese Marktlücke schließt. John Thomas Berrys Arbeit 'Inside the AMIGA' wurde ins Deutsche übersetzt und ist unter dem Titel 'AMIGA Systemprogrammierung in C' erhältlich. Sie beschäftigt sich hauptsächlich mit der Programmiersprache 'C', so daß zum Abtippen der sehr zahlreichen Programmbeispiele ein C-Compiler notwendig ist. Berry verwendete den C-Übersetzer von Lattice, doch Besitzer von anderen Compilern können die Programme leicht anpassen.

Zu Beginn erläutert der Autor allgemeine Merkmale des Amiga wie beispielsweise den Aufbau der Hardware oder der Software. Dieses Kapitel ist zwar etwas knapp ausgefallen, gibt aber doch einige hilfreiche Informationen. Das zweite Kapitel erläutert die Programmierumgebung des AMIGA. Hier wird näher auf die Anwendung der Sprache 'C' oder auf das Arbeiten des Linkers eingegangen. Im dritten Kapitel widmet sich der Autor ausführlich der Programmierung von Intuition. Er behandelt alle Komponenten dieser Library wie Fenster, Bildschirme, Menüs, Gadgets usw. Fast immer sind kleine Programmbeispiele den einzelnen Kapitelteilen hinzugefügt, die das gerade Besprochene in praktischer Anwendung erläutern. Einige Erklärungen finde ich wiederum zu kurz, so wird beispielsweise die Erzeugung von sogenannten Alerts äußerst dürftig besprochen. Kapitel 4 greift die Prozeßkontrolle sowie das AmigaDOS auf. Die einzelnen Teile bleiben zwar meist an der Oberfläche, bieten aber dennoch gute Informationen. Kapitel 5 befaßt sich erneut mit Intuition und beinhaltet das Zeichnen mit dieser Bibliothek. Hier werden Funktionen und Strukturen erklärt, die mit der grafischen Gestaltung eng zusammenhängen.



Kapitel 6 behandelt die Sprites des AMIGA. Der Autor erläutert den Aufbau, die Bewegung sowie die Animation von Sprites in kleinen Programmbeispielen. Kapitel 7 greift ein weiteres, sehr komplexes und spezielles Thema des AMIGA auf: Die Programmierung von Klängen. Hier beschreibt der Autor die Tonerzeugung zunächst im allgemeinen, dann auf Computer bezogen und schließlich speziell die Erzeugung von Klängen auf dem Amiga. Das achte Kapitel beschäftigt sich mit der Sprachausgabe des Rechners und deren Programmierung. Kapitel 9 befaßt sich mit der Programmierung von Disketten-Dateien. Der Autor gibt hier wieder einen allgemeinen Einstieg und geht dann auf die Programmierung ein.

Fazit

Das Buch 'AMIGA Systemprogrammierung in C' von John Thomas Berry ist für Systemprogrammierung durchaus zu empfehlen. Auch für Einsteiger kann dieses Werk durchaus in Betracht gezogen werden. Zwar sind einige Abschnitte etwas knapp und dürftig geraten, sie geben aber dennoch ausreichende Informationen an den Leser weiter. Auch die mangelnde deutschsprachige Literatur über die Programmierung des AMIGA-Sytems machen das Buch für jeden Programmierer interessant. Die 454 Seiten sind also ihr Geld wert.

John Thomas Berry:
AMIGA Systemprogrammierung in 'C'.
te-wi Verlag GmbH
354 Seiten, DM 59,–.

Ein MEGABYTE im AMIGA 1000

Mit ca. 120.– DM und den eigenen Lötkünsten ist es möglich ein Megabyte im Amiga unterzubringen. Die Voraussetzung für den Einbau dieser Modifikation ist allerdings eine ruhige Löthand und eine schon existierende 256 Kbyte Erweiterung im Frontpanel.

Die verwendeten Ram Bausteine vom Typ 41464 finden auch von Commodore selbst Verwendung. Dieser Typ ist im Gegensatz zu dem bekannteren 41256 in vier Bit-Reihen organisiert. Wir verwenden 16 Stück dieses ICs. Diese werden aus Platzgründen direkt in zwei Ebenen auf die Orginalrams aufgelötet. Es ergibt sich also folgende Organisation:

2 Reihen mal 4 Bit in 4 Ramchips zu 64 K = 2x4x4x64 k = 256 Kbyte mal zwei Ebenen = 512 Kbyte

Mit diesen 512 Kbyte plus den originalen 256 und der Ramerweiterung unter der Frontplatte besitzt Ihr Amiga 1000 nach Abschluß dieses Bauvorschlags ein Megabyte Speicher. In unserem Prototyp kammen Rams der Firma Nec zum Einsatz. Die Zugriffszeit sollte weniger als 150 Nanosekunden betragen. Desweiteren benötigen Sie acht Widerstände je 33 Ohm 1/4 Watt und ca. 40 cm. Flachbandkabel oder isolierte Litze.

Bevor Sie Ihr Gerät öffnen machen Sie sich bitte bewußt, daß Sie eventuell noch vorhandene Garantieansprüche mit dem Entfernen der Schrauben verlieren.

Zerlegen

Haben Sie sich alle Teile besorgt und sind bis hierher vorgedrungen, so können Sie nun das Gerät öffnen. Entfernen Sie dazu alle Kabel (Netz-, Monitor-, Tastatur- sowie eventuell Druckerkabel) und die 256 Kbyte Erweiterung unter der Frontabdeckung. Danach können Sie die fünf Schrauben auf der Unterseite des Rechners lösen. Drehen Sie den Rechner jetzt vorsichtig um und heben Sie den Gehäusedeckel ab. Entfernen Sie nun alle Schrauben des Abdeckblechs. Achten Sie auch auf umgebogene Blechlaschen, die das Blech zusätzlich halten. Bei den älteren Modellen muß jetzt noch die Huckepackplatine entfernt werden. Dabei sollten Sie sorgfältig auf die Verbindungsstifte achten. Diese dürfen auf keinen Fall verbiegen oder gar abbrechen. Haben Sie alle bis jetzt beschriebenen Arbeitsgänge erledigt, können Sie direkt am Frontport die acht 41464 der Hauptplatine erkennen. Gleich daneben liegen die ICs mit der Kennzeichnung U1G (74F138), U1H (74F138), U2H (74F399) und am Laufwerk U6P (DPALCAS). Bei der Ausführung mit Huckepackplatine sind die ICs U1H (74F138), U1I (74F138), U1J (74F399) und auf der Huckepackplatine U6K (DPALCAS) von Bedeutung.

RAMs gepackt

Die für den Einbau vorgesehenen 16 RAM Chips müssen erst speziell prepariert werden. Biegen Sie hierzu den Pin 16 eines jeden ICs rechtwinklig ab (siehe Zeichnung) und löten Sie je zwei der RAMs mit einem Abstand von ca. 1 mm aufeinander (Pin an Pin außer Pin 16). Sie haben nun acht RAM-Pärchen mit jeweils zwei abgebogenen Pins (Nr.16). Bevor Sie diese 'Doppelchips' in Ihren Rechner einlöten müssen Sie zwei Verbindungen auf der Hauptplatine auftrennen. Dies ist jeweils der Pin 3 der oben aufgeführten 74F138 TTLs (U1G und U1H im PAL Amiga bzw. U1H und U1I im Amiga mit Huckepack). Am besten gelingt man dies mit einen scharfen kleinen Seitenschneider oder einem Skalpell (Vorsicht mit den Fingern). Trennen Sie die Verbindung möglichst nah an der

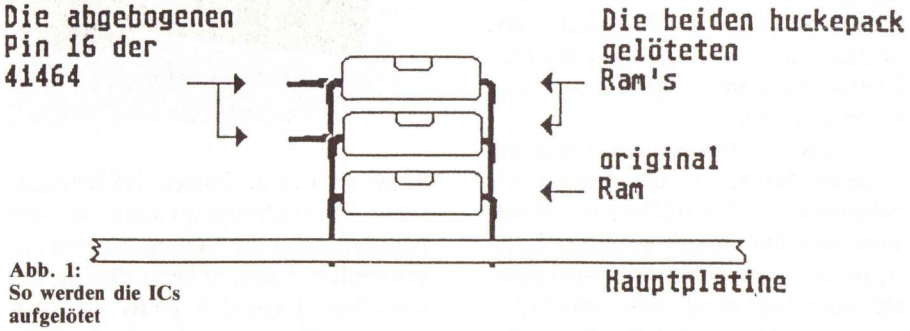


Abb. 1: So werden die ICs aufgelötet

Platine, so daß Sie den am IC verbleibenden Rest etwas nach oben biegen können (hier wird später ein Draht angelötet) .

und eingelötet

Löten Sie nun die Doppel-Rams auf die Rams der Hauptplatine. Verfahren Sie hier genauso wie beim Aufeinanderlöten der Doppel-RAMs. Arbeiten Sie bitte mit einem feinen Elektroniklötkolben (max. 15 – 20 Watt) nach Möglichkeit mit geerde-

ter Lötspitze. (Elektroniklötdraht versteht sich ja von selbst).

Verdrahtet

Nachdem jetzt alle Bauelement präpariert sind, kann die Schaltung gemäß der Zeichnung verdrahtet werden. Vergessen Sie hier bitte nicht die Widerstände und achten Sie darauf, daß die an Pin 3 des 74F138 angelöteten Drähte keinen Kontakt zur Platine haben. Haben Sie alle Drähte gezogen kann ein erster Funktionstest vorgenommen werden. Die Besitzer der Huckepackversion müssen natürlich die Platine wieder ordnungsgemäß einsetzen.

Software

Der Rechner sollte sich nach einer weiteren Überprüfung der Kabel beim Einschalten normal verhalten. Booten Sie nun Ihr System mit der Kickstart V1.1 . Der Rechner muß sich auch hier normal verhalten. Der freie Speicherplatz sollte sich jetzt noch innerhalb der 512 Kbyte Grenze bewegen, da V 1.1 das zusätzliche RAM nicht selbstständig erkennt. Dieser Speicher muß mit dem Befehl 'addmem $080000 $0fffff' erst eingebunden werden. Jetzt sollte der Rechner ca. 900 KByte frei melden. Für Kickstart V1.2 ist vor dem 'addmem'-Befehl noch ein 'make512' aufzurufen, da der zusätzliche Speicher unter Umständen als Chip Memory erkannt wurde.

Der Zusammenbau erfolgt dann in umgekehrter Reihenfolge. Vergessen Sie hier bitte keine Schraube am Abdeckblech, da dieser Umstand unter ungünstigen Verhältnissen zu lauten Laufwerksgeräuschen führen kann. Doch nun viel Spaß mit dem Mega-Amiga (Der ST läßt grüßen).

(GC)

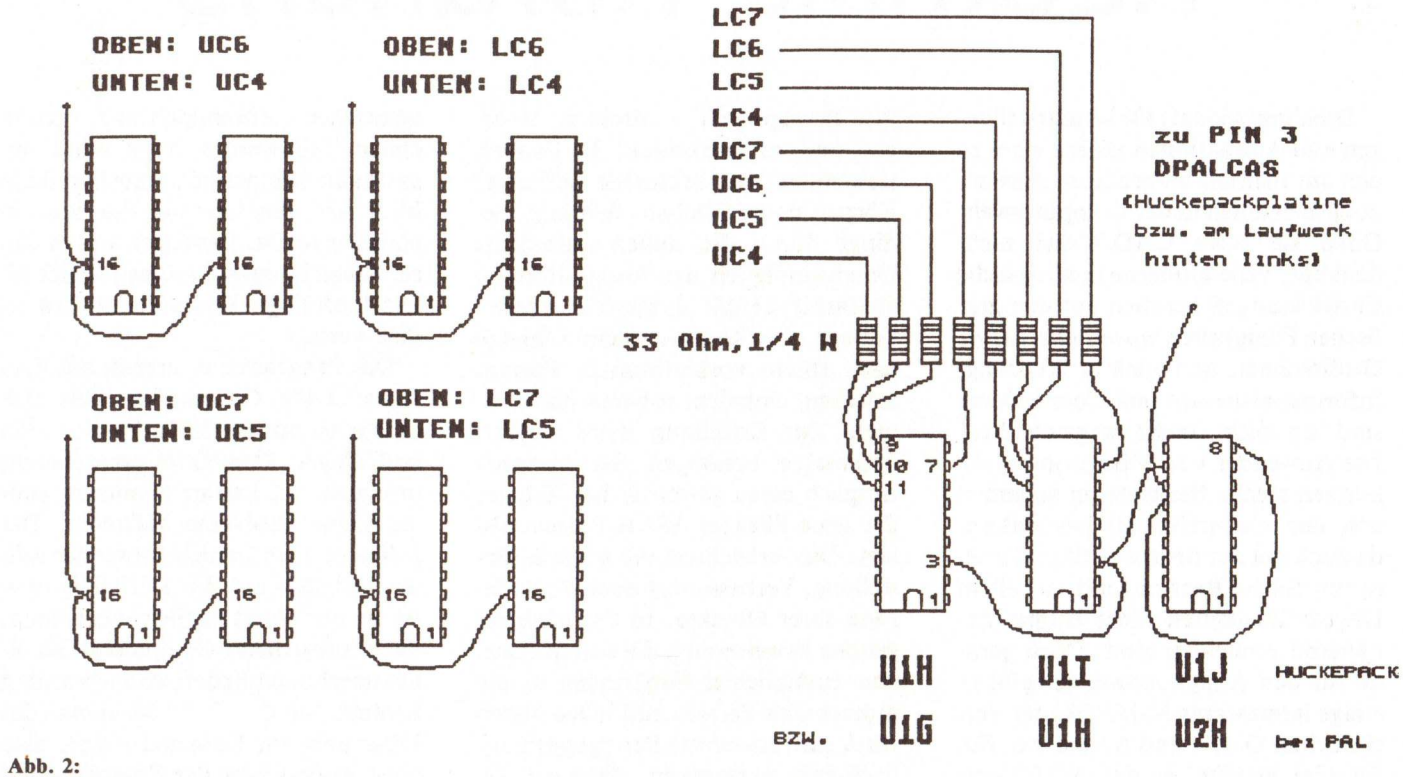


Abb. 2:
Einbauplan

Stückliste:

16 x 41464 – 12 oder – 15
8 x 33 R 1/4 Watt
ca. 40 cm 8-fach Flachbandkabel oder Litze

LISTING

Wenn Sie an dreidimensionalen Ansichten interessiert sind, sollten Sie sich diesen Artikel und das dazugehörige Listing etwas genauer ansehen. In diesem ersten Teil eines Programmprojekts geht es um Grundlagen der 3D-Vektorprogrammierung und um ein Modul, das schon für sich dem Anwender interessante Einsichten gewährt.

Realtime Rotations

Dreidimensionale Objektdarstellungen und Animationen zählen wohl zu den am meisten verbreiteten Anwendungsbereichen in der Computerwelt: Ohne sie wäre CAD/CAM nicht denkbar, viele moderne medizinische Entwicklungen beruhen auf den grafischen Fähigkeiten wissenschaftlicher Großrechner, und auch in Werbung, Informationswesen und Unterhaltung sind sie nicht mehr wegzudenken. Die Anwender von Mikrocomputern können solche Beispiele im allgemeinen nur ehrfurchtsvoll bewundern, da auch mit der neusten Mikro-Generation solche Rechen- und vor allem Displayfähigkeiten nicht einmal annähernd erreichbar sind. Doch gerade für den Amiga-Anwender gibt es einige interessante Möglichkeiten vereinfachter Grafik und Animation. Ein Beispiel hierfür ist der 3D-Object-Animator.

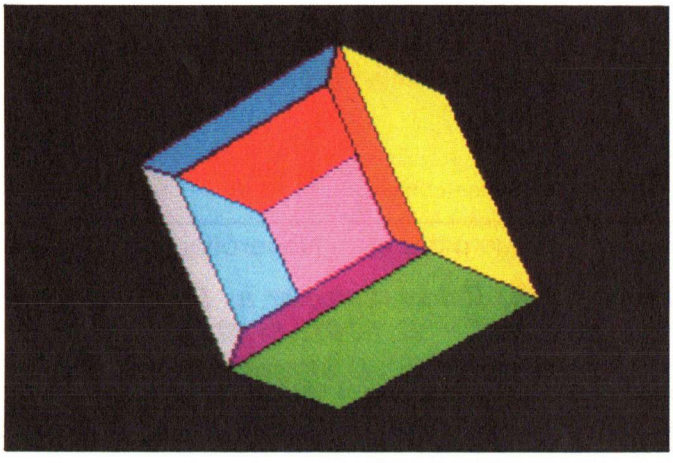
Der 3D-Object-Animator

Dieses Programm soll Ihnen die Möglichkeit geben, räumliche Objekte zu definieren und sie anschließend – unter ständiger Einflußnahme auf ihre Bewegungen – direkt zu bewegen und zu betrachten. Es handelt sich dabei um vektoriell definierte Körper, deren Flächen allerdings, bedingt durch die außergewöhnliche Geschwindigkeit des Amiga-Blitters, vielfarbig gefüllt dargestellt werden können. Sie können solche Objekte nach einem vorbestimmten Format erstellen, einladen, rotieren und zoomen. Zur Erstellung Ihrer Objektdatensätze benötigen Sie übrigens lediglich einen gewöhnlichen Editor, der seine Files im ASCII-Format ablegt. Dies erleichtert die schnelle Erstellung, Verbesserung oder Veränderung Ihrer Objekte. In der nachfolgenden Erweiterung, die sich in Form von zusätzlichen Funktionen in die momentane Version einbinden lassen wird, sind erweiterte Bewegungsmöglichkeiten vorgesehen, die zwar die Ablaufgeschwindigkeit (aufgrund vermehrter mathematischer Operationen) herabsetzen werden, den Anwender aber durch die Möglichkeit entschädigen, Sequenzen verschiedener Bewegungsabläufe „aufzunehmen“ und sie sich hinterher in einer Art Video anzusehen. Zusammen mit diesem ersten Teil werden dann schon respektable Dimensionen (auch im Hinblick auf den Umfang des Sourcecodes) erreicht. Deswegen soll in dieser Ausgabe zunächst eine für sich allein lauffähige Vorabversion behandelt werden.

Das Programm ist erstellt mit dem Aztec 3.40a C-Compiler, läßt sich aber auch mit der 3.20a-Version zum lauffähigen Programm compilieren, und auch bei Lattice-Benutzern sollten keine Probleme auftreten. Der 3.20a- und der Lattice-Anwender sollten lediglich auf die Definition von PI in der Variablenliste verzichten, da es sonst beim Compilieren zu einer unschönen Redefinition-Warning kommt. Auf dem 3.40a kann man das Programm im Default-Format, also ohne Änderungen der Compiler- und Linker-Optionen, compilieren. 3.20a-Anwender sollten darauf achten, daß die Einbindung der Motorola-Fast-FloatingPoint-Routinen im Compiler aktiviert ist, und den Linker mit den Optionen < –lm –lc > starten.

Die Funktionen des 3D-Object-Animators

Der Object-Animator bietet die Möglichkeit, Objekte um ihre drei Raumachsen zu drehen und in der Größe zu verändern. Sämtliche Funktionen werden unter Intuition auf der Tastatur abgefragt, denn erstens ist diese Art der Eingabe sehr schnell, zweitens bleibt der Bildschirm frei für das Objekt, und drittens hätte die Eröffnung eines Requesters den Sourcecode erheblich verlängert. Die Kontrolle über die Drehbewegungen liegt auf dem Zehnerblock der Tastatur; die 8 und die 2 verändern die Drehgeschwindigkeit um die X-Achse positiv bzw. negativ, die 9 und die 1 ver-



ändern die Z-Drehung positiv bzw. negativ, und die 4 und die 6 sind für positive bzw. negative Y-Drehungs-Veränderungen zuständig. Zum Zoomen stehen die Tasten „g" wie „größer" und „k" wie „kleiner" zur Verfügung. Ein als ASCII-Datensatz erstelltes Objekt laden Sie ein, indem Sie „n" wie „neues Objekt" drücken („l" wie „Laden" liegt zu nahe an der k-Taste, so daß man sich beim Zoomen leicht vertippen könnte), worauf das Programm Sie nach dem Namen Ihres Files fragt, den Sie ohne Extension übergeben müsen (alle Objekt-Datensätze benötigen auf Diskette die Extension „.3d"). Findet das Programm das File nicht oder mißlingt das Öffnen aus einem anderen Grund, so meldet es das und fährt danach fort, das (eventuell) vorher geladene Objekt zu bewegen. Andernfalls wird Ihr Objekt eingeladen, sämtliche Drehungen auf Null und der Zoom auf die Voreinstellung von 10 gesetzt. Die letzte Option ist die „q"-Taste, deren Betätigung zum sofortigen Verlassen des Programms führt.

Aller Anfang ist die Mathematik

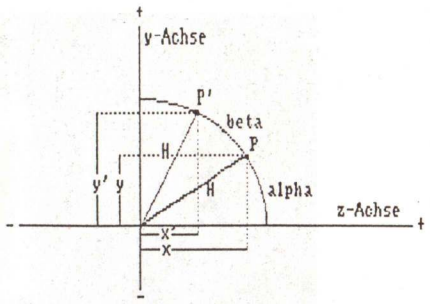
Dies gilt natürlich nicht für die Reihenfolge im Sourcecode (hier wird die Berechnung des Objekts in seiner aktuellen Position in der Funktion transform() abgearbeitet), aber insofern für das Konzept eines Grafikprogramms, als daß man von vorneherein festlegen muß, nach welchem Prinzip, mit welcher Genauigkeit und welchen Toleranzen eine 3D-Grafik erstellt werden soll, was später ja auch mit dem Datenformat korrespondieren muß. Im Falle des 3D-Object-Animators haben wir es mit einigen speziellen Festlegungen zu tun, die nachfolgend erläutert werden.

Um dem Rechner die immense Arbeit einer ständigen Neuberechnung von Sinus- und Cosinuswerten zu ersparen, werden diese nach dem Start des Programms in 1-Grad-Schritten errechnet und in zwei Listen mit entsprechender Feldgröße abgelegt. Um zu vermeiden, daß hierfür Float-Variablen zwischen −1 und 1 notwendig sind (was dazu führen würde, daß sämtliche Zwischenergebnisse als Floats gehandhabt werden müßten), wird jeder Sinus- und Cosinuswert mit dem Faktor 16384 multipliziert. Warum aber ausgerechnet 16384? Erstens, weil beim Multiplizieren mit einem derart großen Wert noch die dritte Stelle hinter dem Komma des ursprünglichen Float-Wertes für das Integer-Ergebnis relevant ist, und zweitens, weil Multiplizieren bzw. (in der Objektberechnung nötiges) Dividieren mit diesem Wert, einem Vielfachen von 2 (genaugenommen 2 14) vom Prozessor besonders schnell erledigt werden kann. Da die Rundungsfehler diese Prinzips für eine nichtwissenschafliche Anwendung sehr gering sind, kann man durch seine Anwendung einen erheblichen Geschwindigkeitsvorteil bei der Objektberechnung verbuchen.

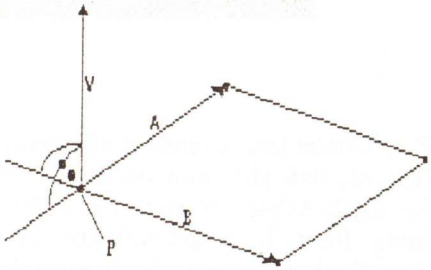
Zur Berechnung der Ansicht des Körpers bezieht sich das Programm nur auf die Sichtbarkeit der einzelnen Flächen, aus denen er zusammengesetzt ist. Hierfür gibt es noch andere Möglichkeiten, zum Beispiel das Sortieren der Flächen nach ihren durchschnittlichen Raumtiefen und anschließendes Zeichnen von hinten nach vorne, wobei weiter vorn liegende Flächen die hinteren übermalen. Diese Technik birgt aber auch Ungenauigkeiten in sich und führt nicht immer zu korrekten Ansichten (die man eigentlich nur durch das Ray-Tracing-Verfahren erhält), so daß sie den Zeitverlust, den sie mit sich bringt, in dieser Vorabversion (bei der vor allem auf hohe Rechengeschwindigkeit Wert gelegt wird) nicht rechtfertigt. In der nachfolgenden Erweiterung werden dann beide Techniken, die der Sichtbarkeitsprüfung und die der Tiefensortierung, miteinander kombiniert, da es nicht mehr auf höchste Realtime-Ausführungsgeschwingigkeit ankommt.

In der vorliegenden Version berechnet der Animator, ob eine Fläche dem Beobachterblickpunkt zu- oder abgewandt ist, und zeichnet sie im zweiten Fall. Grundsätzlich kann er nach diesem Verfahren nur rein konvexe Körper (Körper, deren Kanten alle Außenkanten sind) darstellen, aber mit einem kleinen Trick bei der Erstellung des Datensatzes, der noch genau erläutert wird, lassen sich auch etwas komplexere und teilkonkave Körper darstellen.

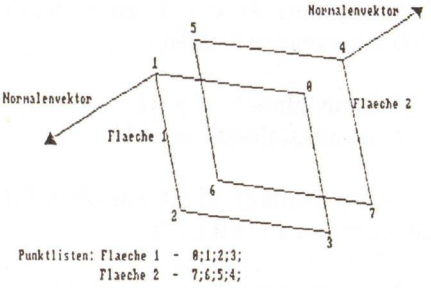
Die Berechnung des Objekts in seiner aktuellen Position wird nach dem Prinzip der Punkttransformation in der Funktion transform() durchgeführt. In einer nach der Anzahl der Objektpunkte indizierten Schleife durchläuft jeder der Punkte drei Rotationsmatrizen, von denen jede für die Drehung um eine Raumachse zuständig ist. Diese Rotationsmatrixen folgen streng den trigonometrischen Grundprinzipien, die hier anhand eines kleinen Beispiels diskutiert werden sollen. Hierbei soll ein Punkt P mit den Raumkoordinaten x = 100, y = 50, z = 0 um 30 Grad um die Z-Achse rotiert werden. Graphisch dargestellt ist dies in Abbildung 1, wobei die X-Achse der Waagerechten und die Y-Achse der Senkrechten entsprechen, während die Z-Achse senkrecht auf dem Mittelpunkt dieses



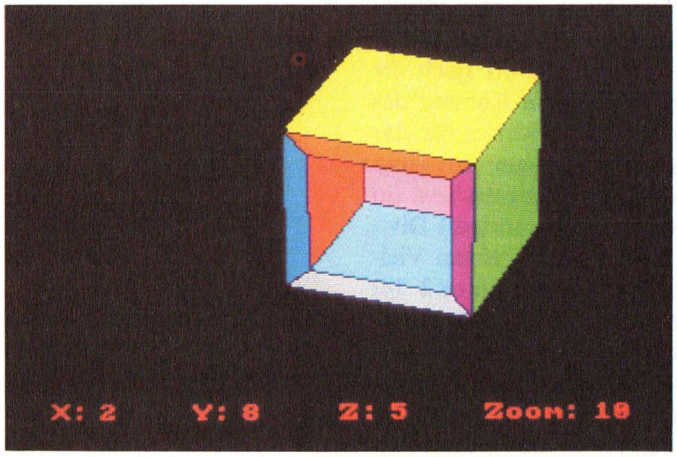
Graphische Darstellung der Bezie hungen zwischen Winkel und Objekt koordinaten



Darstellung des Normalenvektors



Gestaltung der Punktlisten bei dem Betrachter zu- und abgewandten Fldchen



Koordinatenkreuzes steht. Offensichtlich ist, daß sich nun bei Rotation um die Z-Achse (die quasi einer Drehung Ihrer Kickstart-Ausgabe auf dem Tisch entspricht) die Z-Koordinate nicht verändert, also ist z' = z. Um x' und y' zu berechnen, muß man sich zuerst einmal die Beziehungen für den Winkel zwischen der Verbindungslinie H von P zum Punkt 0,0,0 vergegenwärtigen:

1. sinus(alpha) = y/H
2. cosinus(alpha) = x/H

Dementsprechend gilt natürlich für den rotierten Punkt P':

3. sinus(alpha + beta) = y'/H
4. cosinus(alpha + beta) = x'/H

Der Winkel beta entspricht in diesem Beispiel 30 Grad. Für die weitere Vorgehensweise sind die Additionstheoreme der Winkelfunktionen (die hier nicht hergeleitet werden sollen, da dies den Rahmen sprengen würde, mathematisch interessierte Leser könnten in einem Mathematiklexikon o.ä. nachschlagen) notwendig; diese lauten für Sinus und Cosinus:

sinus(alpha + beta) =
sinus(alpha) ★ cosinus(beta) + cosinus(alpha) ★ sinus (beta)

cosinus(alpha + beta) =
cosinus(alpha) ★ cosinus(beta) – sinus(alpha) ★ sinus (beta)

Somit kann man die Ausdrücke

x'/H = cosinus(alpha + beta), und
y'/H = sinus(alpha + beta)

ersetzen durch

x'/H = cosinus(alpha) ★ cosinus (beta) – sinus(alpha) ★ sinus (beta) und
y'/H = sinus(alpha) ★ cosinus(beta) + cosinus(alpha) ★ sinus (beta).

Multipliziert man jetzt beide Seiten der letzten Gleichungen mit H, so erhält man

x' = cosinus(alpha) ★ H ★ cosinus(beta) – sinus(alpha) ★ H ★ sinus (beta), und
y' = sinus(alpha) ★ H ★ cosinus (beta) + cosinus(alpha) ★ H ★ sinus (beta).

Durch Einsetzen der Festlegungungen 1 und 2 kommt man auf

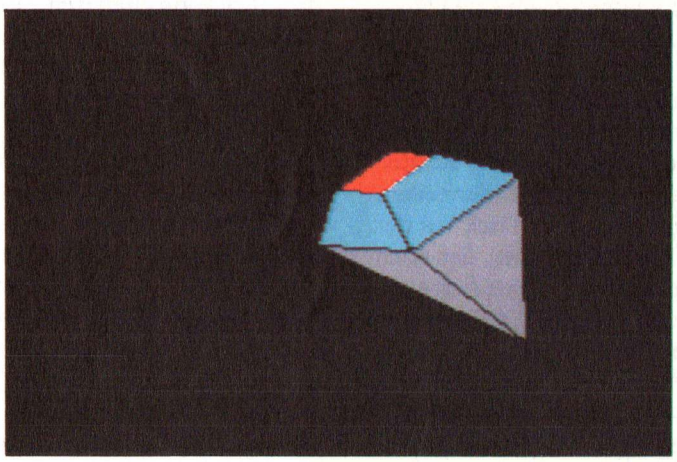
x' = x ★ cosinus(beta) – y ★ sinus(beta), und
y' = y ★ cosinus(beta) + x ★ sinus(beta).

Hiermit ist die Rotationsmatrix um die Z-Achse komplett; sie lautet insgesamt

x' = x ★ cosinus(beta) – y ★ sinus(beta)
y' = y ★ cosinus(beta) + x ★ sinus(beta)
z' = z.

Daraus ergeben sich für x', y' und z' die Werte 61.6, 93.9 und 0. Analog zu dieser Vorgehensweise werden die Punkte um die anderen Raumachsen rotiert. Nach jeder Rotation müssen die mit Sinus- und Cosinuswerten multiplizierten Koordinaten noch durch 16384 dividiert werden, da es sonst zu krassen Verzerrungen käme.

Die erhaltenen Raumkoordinaten eines Punktes werden nachfolgend mittels Zentralprojektion in zweidimensionale Bildkoordinaten umgerechnet, wozu die jeweilige X- und Y-Koordinate durch die Z-Koordinate geteilt werden muß. Da der Mittelpunkt des räumlichen Koordinatensystems ohne Umrechnung allerdings auf dem 0,0-Punkt des Ausgabewindows und auf der Bildschirmebene läge, muß zum Z-Wert ein Faktor addiert werden, was die Verschiebung des Mittelpunkts in die räumliche Tiefe bewirkt. Zu den erhaltenen Bildkoordinaten müssen nun nur noch 160 bzw. 100 addiert werden, was jeweils der halben Bild-



Kick Start-ABO

Straße/Postfach

PLZ Ort

Telefon: _____

_____

_____

Bitte mit 60 Pf. frankieren

Postkarte

***Heim*-VERLAG**
Heidelberger Landstraße 194

6100 Darmstadt-Eberstadt
Telefon 06151/56057

Name _____

Vorname _____

Straße _____

PLZ/Ort _____

Alter _____

Das finde ich an KICKSTART
toll/nicht so toll:

Bitte mit 60 Pf. frankieren

Postkarte

MERLIN Computer GmbH
Kick START Redaktion

Industriestraße 26

6236 Eschborn

Kick Start-Kleinanzeigen

Name

Vorname

Firma

Straße/Postfach

PLZ Ort

_____
Datum Unterschrift

Bei Jugendlichen unter 18 Jahren bitte Unterschrift des Erziehungsberechtigten.

Bitte mit 60 Pf. frankieren

Postkarte

Anzeigenservice
***Heim*-VERLAG**
Heidelberger Landstraße 194

6100 Darmstadt-Eberstadt
Telefon 06151/56057

Amiga Kick Start-Abonnement

Ja, bitte senden Sie mir die Amiga-Computer Fachzeitschrift ab _____
für mindestens 1 Jahr (11 Hefte) zum ermäßigsten Preis von jährlich DM 70,— frei Haus.
(Ausland: Nur gegen Scheck-Voreinsendung DM 90,— Normalpost.)
Der Bezugszeitraum verlängert sich nur dann um ein Jahr, wenn nicht 6 Wochen vor Ablauf des Abonnements gekündigt wird.

Name _____

Vorname _____

Straße/Nr. _____

___ ____ _____
PLZ Ort

Gewünschte Zahlungsweise bitte ankreuzen

☐ Bequem und bargeldlos durch Bankeinzug

Konto-Nr. _____ BLZ _____

Institut _____ Ort _____

☐ Ein Verrechnungsscheck über DM_____ liegt bei.

☐ Gegen Rechnung

Garantie:
Diese Bestellung kann ich schriftlich innerhalb einer Woche (rechtzeitige Absendung genügt) widerrufen. Dies bestätige ich durch meine 2. Unterschrift.

Datum _____ Unterschrift _____

Datum _____ Unterschrift _____

Leser-Umfrage

(bitte tragen Sie hier Ihre Kreuzchen ein!)

1. ☐ 500 ☐ 1000 ☐ 2000
2. ☐ 1/2 M ☐ 1 M ☐ 2 M
3. ☐ ja ☐ nein
4. ☐ ja ☐ nein
5. Welchen! __
6. ☐ ja ☐ nein
7. ☐ ja ☐ nein
8. ☐ XT ☐ AT ☐ 68020
9. ☐ ja ☐ nein
10. ☐ ja ☐ nein
11. ☐ ja ☐ nein
12. Welche? ____
13. ☐ AMIGA 500 ☐ AMIGA 2000
14. ☐ Bridgeboard XT ☐ Bridgeboard AT
15. ☐ 500 K Speicher ☐ 1 M Speich. o.>
16. ☐ Zweitlaufwerk ☐ Festplatte
17. _____
18. ☐ PASCAL
19. ☐ BASIC
20. ☐ C
21. ☐ Assembler
22. ☐ _____
23. ☐ Softwaretests
24. ☐ Hardwaretests
25. ☐ Programmierkurse
26. ☐ Anwendungsbeispiele
27. ☐ Listings
28. ☐ Aktuelle Meldungen
29. ☐ Spiele
30. ☐ MS-DOS auf Amiga
31. ☐ Tips & Tricks
32. ☐ Grafik
33. ☐ Musik
34. ☐ Hardwarebasteleien
35. ☐ Buchkritiken
36. _____
37. ☐ Büro ☐ Wissenschaft/Technik
38. ☐ ja ☐ nein
39. ☐ beruflich ☐ privat
40. Welche? _____

Amiga Kick Start Kleinanzeigen-Auftrag

Bitte veröffentlichen Sie für mich folgende Kleinanzeige in der angekreuzten Rubrik

Biete an ☐ Hardware ☐ Software **Ich suche** ☐ Hardware ☐ Software ☐ **Tausch** ☐ **Kontakte** ☐ Verschiedenes ☐

30 Buchstaben je Standardzeile – incl. Satzzeichen und Wortzwischenräume.
Groß- und Kleinbuchstaben verwenden, fettgedruckte Wörter unterstreichen.

Bearbeitung **nur gegen Vorausscheck** über den entsprechenden Betrag (keine Überweisung)

☐ privat = DM 7,– je Zeile incl. MwSt.
☐ gewerblich = DM 15,– je Zeile + MwSt.
☐ Chiffregebühr = DM 10,–

☐ Scheck über DM_____ ist beigefügt

Bei Angeboten: Ich bestätige, daß ich alle Rechte an den angebotenen Sachen besitze.

Datum _____ Unterschrift _____

Absenderangaben auf der Rückseite nicht vergessen

schirmauflösung entspricht, um die Bildkoordinaten auf den Bildschirmmittelpunkt zu relativieren.

Nachdem nun alle Punkte transformiert worden sind, müssen die Sichtbarkeiten der Flächen berechnet werden. Hierzu bedient man sich der Vektor- und Determinantenrechnung, da man den Normalenvektor V, welcher senkrecht auf einer Fläche steht, benötigt. Dieser errechnet sich aus dem Vektorprodukt zweier Vektoren A und B, die von einem Punkt P einer Fläche auf die benachbarten Punkte zeigen (Abbildung 2). Die Komponenten dieses Vektors sind in der Mathematik definiert als

$$x(V) = \begin{vmatrix} y(A) & z(A) \\ y(B) & z(B) \end{vmatrix} \quad y(V) = \begin{vmatrix} x(A) & z(A) \\ x(B) & z(B) \end{vmatrix} \quad z(V) = \begin{vmatrix} x(A) & y(A) \\ x(B) & y(B) \end{vmatrix}.$$

Dies ist gleichzusetzen mit

$$x(V) = y(A) \star z(B) - z(A) \star y(B),$$
$$y(V) = x(A) \star z(B) - z(A) \star x(B) \text{ und}$$
$$z(V) = x(A) \star y(B) - y(A) \star x(B).$$

Von Interesse ist nun lediglich die Z-Komponente des errechneten Vektors, da diese die räumliche Tiefe des Punktes angibt, auf den der Vektor zeigt, allerdings relativ zur räumlichen Tiefe des Punktes P, auf dem der Vektor steht. Aus diesem Grund wird zu z(V) der Wert von z(P) addiert, so daß man die absolute Tiefe des Zielpunkts erhält. Aus der Anschauung ergibt sich nun, daß eine Fläche dem Beobachterblickpunkt dann zugewandt ist, wenn die Tiefe des Punkts, auf den V zeigt, kleiner ist als die Tiefe des Punkts P, von dem der Vektor ausgeht. Erkennt das Programm diesen Fall, so wird der Entscheidungswert für die Sichtbarkeit auf TRUE gesetzt. Diese Verfahren muß natürlich auf alle Flächen angewandt werden. Es sei hier noch angemerkt, daß es keine Allgemeingültigkeit für sich in Anspruch nimmt, da es sich um eine Vereinfachung der in Wirklichkeit noch etwas komplexeren Berechnungsweise handelt; dennoch arbeitet es im vorliegenden Programm mit hoher Genauigkeit, weswegen ihm in Anbetracht seiner kurzen Ausführungszeiten der Vorzug gegeben wurde.

Funktionsvielfalt im Überblick

Nun noch eine kleine Beschreibung der verschiedenen Funktionen, mit denen der 3D-Object-Animator in der vorliegenden Grundversion arbeitet.

init_trig() erledigt die weiter oben erwähnte Berechnung je einer Sinus- und Cosinusliste. Aus diesem Grund wird es gleich zu Anfang des Programms aufgerufen.

graph_screens() öffnet zuerst die zur Benutzung von Intuition und Betriebssystemgrafikbefehlen erforderlichen Libraries. Als nächstes werden zwei Screens geöffnet, deren Typ in der globalen Variablenliste definiert ist (zwei Screens sind notwendig, da ein flackerfreier Bildaufbau nur dann gewährleistet ist, wenn ein neues Bild auf einem im Hintergrund liegenden Screen gezeichnet wird und dieser erst nach dem Zeichnen in den Vordergrund geschaltet wird). Nun werden vier Windows initialisiert und geöffnet, von denen zwei für die grafische Ausgabe und zwei für die Informationszeile vorgesehen sind (jeweils eines von jeder Sorte pro Screen). Beide Screens wie auch beide Graphik- und Infowindows sind exakt vom gleichen Typ, mit der winzigen Ausnahme, daß nur das zweite Infowindow aktiv und somit für Tastatureingaben annahmebefähigt ist (ständiges Aktivieren und Deaktivieren der Infowindows im Screenumschalttakt würde nämlich sehr viel Zeit kosten). Es handelt sich übrigens deswegen um Low-Resolution-Screens, da bei doppelter Bildschirmauflösung und 4 Bitplanes die Bitplane-DMA beginnt, dem 68000er Prozessor Bus-Zyklen zu „stehlen" und dies den Gesamtablauf wie auch die Grafikausgabe des Programms verlangsamen würde.

Die Routine object_drawer() ist der eigentliche Mainpart des Programms. Hier werden in einer endlosen, nur durch Eingabe von „q" zu verlassenden Schleife die Funktion transform() aufgerufen, der im Hintergund liegende Bildschirm gelöscht, das Objekt darauf gezeichnet und dieser dann nach vorne geschaltet. Vor jedem zweiten Bildschirmschalten wird außerdem lies_tastatur() aufgerufen, um die aktuellen Usereingaben abzufragen.

Bei transform() handelt es sich um die Berechnungsroutine, deren Funktionsabläufe schon weiter oben eingehend erläutert wurden.

Die Funktion lies_tastatur() liest einzelne Zeichen über den Message-Port des aktiven Infowindows und verwertet diese den Tastenbelegungen zufolge. Wird ein Drehwert oder der Zoom verändert, so gibt sie den aktualisierten Wert sofort auf beiden Infowindows aus. Wird die Taste „n" gewählt, ruft sie die Objektfileladeroutine file_loader() auf. Außerdem liefert lies_tastatur() einen Entscheidungswert an die aufrufende Funktion object_drawer() zurück, welcher bei Betätigen der „q"-Taste TRUE wird und object_drawer() zum Abbruch bewegt.

write_int() dient dazu, eine Integervariable in einen String zu schreiben und dann mittels der Betriebssystemfunktion Text() auf den Infowindows auszugeben. write_int() benötigt als Parameter den auszugebenden Wert, die gewünschte x- und y-Position der Ausgabe und den Rast-Port des gewünschten Ausgabewindows.

file_loader() dient zum Einlesen eines Objektdatensatzes von Diskette. Bei Aufruf wird die laufende Animation gestoppt und vom User der Name des zu ladenden Objekts erfragt. Kann file_loader() den gewünschten Datensatz nicht finden, meldet die Funktion dies und fährt nach einem kurzen Delay mit der letzten Animation fort. Andernfalls wird das Objekt eingelesen, und die Drehwerte werden auf Null zurückgesetzt. Zur Zuweisung der gelesenen Werte bedient sich file_loader() der Funktion lies_int(), welche einen String (nur numerische Zeichen; Buchstaben, Spaces etc. werden ignoriert) bis zum nächsten Semikolon liest, ihn in einen Integerwert umwandelt und zurückgibt. file_loader() prüft nicht, ob die Anzahl der Werte im Datensatz mit den Festlegungen übereinstimmt; achten Sie deshalb darauf, daß Ihre Datensätze korrekt sind, da das Programm ansonsten an dieser Stelle hängenbleiben oder sehr ominöse Objekte darstellen könnte.

info_text() beschreibt die Infowindows mit mit dem unveränderlichen Grundtext.

ausstieg() wird bei Verlassen des Programms aufgerufen, um sämtlichen okkupierten Speicherplatz an das DOS zurückzugeben.

Die Erstellung der Objektdatensätze

Die Objektdatensätze, die der 3D-Object-Animator verarbeiten soll, müssen sich streng an die festgelegte Vereinbarung halten. Das Format sieht folgendermaßen aus:

1. Anzahl der Objektpunkte
2. Anzahl der Objektflächen
3. pro Punkt je eine x-, y- und z-Koordinate
4. pro Fläche die Anzahl der Punkte der Fläche
5. pro Fläche eine Punktliste, in der die Anzahl der Punkte mit der vorher getroffenen Vereinbarung der Punktzahl dieser Fläche übereinstimmen muß
6. pro Fläche der gewünschte Farbwert der Fläche

Die einzelnen Werte müssen durch Semikolons voneinander getrennt sein; Spaces und Kommentare ignoriert die Laderoutine, außer, wenn in letzteren numerische Zeichen enthalten sind, dies ist also unbedingt zu vermeiden. Achten Sie auch darauf, daß alle Punktlisten der Flächen linksherum definiert sind (dies gilt für den Fall, daß die Fläche in der Grundstellung des Objekts sichtbar, also nach vorne gewandt, ist; ist sie Ihnen abgewandt, so verkehrt sich das Ganze natürlich. Beachten Sie dazu bitte Abbildung 3), da andernfalls der vom Programm berechnete Normalenvektor auf der Rück- oder besser Innenseite der Fläche steht und die Sichtbarkeit falsch berechnet wird. Sollten von Ihnen gestaltete Datensätze nicht das gewünschte Ergebnis bewirken, so ist die Überprüfung der Punktlisten meist ein geeigneter Ansatzpunkt zum Debugging des Datensatzes. Körper mit Innenflächen lassen sich übrigens erstellen, indem man die innenliegenden Flächen zuerst in die Liste schreibt, so daß die äußeren Flächen diese bei eventueller simultaner Sichtbarkeit und Überlappung übermalen. Dies Prinzip können Sie mehrfach anwenden, so daß ein komplexer Körper quasi in einzelnen Schalen von Innen nach Außen gezeichnet wird. Als Beispiele finden Sie am Ende des Listings zwei Datensätze, einer davon kommentiert. Der zweite ist ein Würfel mit 5 Innenflächen, deren Punktlisten als die ersten fünf definiert sind (bei diesem Beispiel stehen mehrere Punktlisten in einer Zeile; sie sind durch einige Leerstellen voneinander getrennt). Kennzeichnen Sie bitte alle Ihre Objekte mit der Extension „.3d“, da die Laderoutine nur derart gekennzeichnete Datensätze annimmt (diese Extension müssen Sie beim Einladen nicht angeben). Die maximalen Werte der x-, y- und z-Koordinaten sollten übrigens zwischen −2000 und 2000 liegen. Wählen Sie allerdings vom Betrag her auch nicht zu kleine Werte (kleiner als ca. 200), da ansonsten durch die Rundungsfehler in der Integerberechnung sichtbare Verzerrungen auftreten.

So, das wär's dann erst einmal gewesen. Hoffentlich sind Sie nicht erschreckt von der Länge des Sourcecodes; er ist eben nicht nach rationalen Gesichtspunkten gestaltet, sondern an der Ablaufgeschwindigkeit orientiert. Das Projekt wird demnächst fortgesetzt mit Erweiterungsmodulen bzw. -funktionen, die Ihnen erweiterte Bewegungskontrollmöglichkeiten, höheren Bedienungskomfort (was dann in jedem Fall nötig sein wird) und vor allem der Möglichkeit der Aufnahme von Animationssequenzen bieten werden. Bewahren Sie sich den Source deshalb sorgfältig auf, denn sonst müssen Sie große Teile davon neu eingeben. Bis dahin aber erst einmal viel Spaß mit diesem Modul und Ihren selbsterstellten rotierenden Objekten!

(Wolf Dietrich)

```
/*************************************************************/
/*                  The 3D-Object-Animator                   */
/*                 written by Wolf Dietrich                  */
/*              especially for  KICKSTART  9/87              */
/*************************************************************/
#include <exec/types.h>
#include <graphics/gfxmacros.h>
#include <stdio.h>
#include <math.h>
#include <exec/devices.h>
#include <exec/memory.h>
#include <intuition/intuition.h>
#include <ctype.h>
#include <fcntl.h>
#include <functions.h>

#define CONTINUE        for(;;)
#define MAXPPF          20      /* max. Anzahl Punkte pro Flache */
#define MAXFL           40      /* max. Anzahl Flachen           */
#define MAXP            200     /* max. Anzahl Punkte            */
#define FAK             100000L    /* Tiefenprojektionsfaktor    */
#define PI              3.1415926

struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;

struct NewScreen graph_screen =
{
 0, 0, 320, 214,                    /* einfacher Low-Res Screen */
 4, 0, 1, NULL, CUSTOMSCREEN,       /* etwas hoher als normal,  */
 NULL, NULL, NULL, NULL,            /* 16 Farben                */
};

struct ViewPort   *vp_1, *vp_2;
struct Screen     *screen_1, *screen_2;
struct NewWindow  graph_window_1,graph_window_2,
                  info_nwindow_1,info_nwindow_2;
struct Window  *window_1, *window_2,
               *info_window_1, *info_window_2;
struct RastPort *rp_1, *rp_2, *rp_3, *rp_4;

struct TmpRas temp_ras;
struct AreaInfo area_info;

BYTE *area_plane;

char *filename, *zielchar, zielstring[10];
UWORD areabuf[100];
/* Hier konnen Sie naturlich Ihre eigenen Farben einsetzen */
USHORT farbliste[] = {0x000,0x00f,0x0f0,0xf00,
                      0x0ff,0xf0f,0xff0,0xfff,
                      0xf6b,0x3a9,0x492,0x777,
                      0xbbb,0x07c,0xc40,0x90a  };
int sinus[360],cosin[360],zoom;
int pkt_zahl = 0,flaech_zahl = 0,pkt_p_flaech[MAXFL] ;
int obj_xturn,obj_yturn,obj_zturn,xdreher,ydreher,zdreher;
int sichtbar[MAXFL],plst[MAXFL][MAXPPF],filenumber;
long sp[MAXP],ze[MAXP],cx[MAXP],cz[MAXP],cy[MAXP],farbe[MAXFL];
float fzoom;


main()
{
 init_trig();

 graph_screens();

 /* Zuweisung von Rast- und ViewPorts furs Screenhandling */

 rp_1 = window_1->RPort;
 rp_2 = window_2->RPort;
 rp_3 = info_window_1->RPort;
 rp_4 = info_window_2->RPort;
 vp_1 = &screen_1->ViewPort;
 vp_2 = &screen_2->ViewPort;

 /* Farben aus der Farbliste in die Colormap der Screens laden */

 LoadRGB4(vp_1,farbliste,16L);
 LoadRGB4(vp_2,farbliste,16L);

 /* Infowindows auf Hintergrundfarbe setzen */

 SetRast(rp_3,0L);
 SetRast(rp_4,0L);

 /* Grenzen des Betriebssystemclippings setzen, aber Vorsicht: */
 /* das Clipping arbeitet nicht immer hunderprozentig          */

 rp_1->Layer->ClipRect->bounds.MinX = (SHORT)0;
 rp_1->Layer->ClipRect->bounds.MinY = (SHORT)0;
```

```
 rp_1->Layer->ClipRect->bounds.MaxX = (SHORT)319;
 rp_1->Layer->ClipRect->bounds.MaxY = (SHORT)199;
 rp_2->Layer->ClipRect->bounds.MinX = (SHORT)0;
 rp_2->Layer->ClipRect->bounds.MinY = (SHORT)0;
 rp_2->Layer->ClipRect->bounds.MaxX = (SHORT)319;
 rp_2->Layer->ClipRect->bounds.MaxY = (SHORT)199;

 /* Die nun folgenden Aufrufe initialisieren Speicherplatz und  */
 /* Stukturen, die die AreaDraw-Routine benotigt                */

 InitArea(&area_info,&areabuf[0],40L);

 rp_1->AreaInfo = &area_info;
 rp_2->AreaInfo = &area_info;
 rp_1->TmpRas   = &temp_ras;
 rp_2->TmpRas   = &temp_ras ;

 area_plane = AllocMem(8000L,MEMF_CHIP);

 rp_1->TmpRas->RasPtr = area_plane;
 rp_1->TmpRas->Size   = (long)RASSIZE(320,200);
 rp_2->TmpRas->RasPtr = area_plane;
 rp_2->TmpRas->Size   = (long)RASSIZE(320,200);

 /* Jetzt geht's in den Hauptteil des Programms  */

 object_drawer();

 ausstieg();
}

graph_screens()
{
 IntuitionBase=(struct IntuitionBase *)
   OpenLibrary("intuition.library",0L);
 GfxBase=(struct GfxBase *)
   OpenLibrary("graphics.library",0L);

 screen_1=(struct Screen *)OpenScreen(&graph_screen);
 screen_2=(struct Screen *)OpenScreen(&graph_screen);

 graph_window_1.LeftEdge = 0;
 graph_window_1.TopEdge  = 0;
 graph_window_1.Width    = 320;
 graph_window_1.Height   = 200;
 graph_window_1.DetailPen = 0;
 graph_window_1.BlockPen = 1;
 graph_window_1.Title    = NULL;
 graph_window_1.Flags    = SIMPLE_REFRESH;
 graph_window_1.IDCMPFlags = NULL;
 graph_window_1.Type       = CUSTOMSCREEN;
 graph_window_1.FirstGadget= NULL;
 graph_window_1.CheckMark  = NULL;
 graph_window_1.Screen     = screen_1;
 graph_window_1.BitMap     = NULL;
 graph_window_1.MinWidth   = 320;
 graph_window_1.MinHeight  = 200;
 graph_window_1.MaxWidth   = 320;
 graph_window_1.MaxHeight  = 200;

 window_1=(struct Window *)OpenWindow(& graph_window_1);

 graph_window_2.LeftEdge = 0;
 graph_window_2.TopEdge  = 0;
 graph_window_2.Width    = 320;
 graph_window_2.Height   = 200;
 graph_window_2.DetailPen = 0;
 graph_window_2.BlockPen = 1;
 graph_window_2.Title    = NULL;
 graph_window_2.Flags    = SIMPLE_REFRESH;
 graph_window_2.IDCMPFlags = NULL;
 graph_window_2.Type       = CUSTOMSCREEN;
 graph_window_2.FirstGadget= NULL;
 graph_window_2.CheckMark  = NULL;
 graph_window_2.Screen     = screen_2;
 graph_window_2.BitMap     = NULL;
 graph_window_2.MinWidth   = 320;
 graph_window_2.MinHeight  = 200;
 graph_window_2.MaxWidth   = 320;
 graph_window_2.MaxHeight  = 200;

 window_2=(struct Window *)OpenWindow(&graph_window_2);

 info_nwindow_1.LeftEdge = 0;
 info_nwindow_1.TopEdge  = 200;
 info_nwindow_1.Width    = 320;
 info_nwindow_1.Height   = 13;
 info_nwindow_1.DetailPen = 0;
 info_nwindow_1.BlockPen = 1;
 info_nwindow_1.Title    = NULL;
 info_nwindow_1.Flags    = SIMPLE_REFRESH;
 info_nwindow_1.IDCMPFlags = NULL;
 info_nwindow_1.Type       = CUSTOMSCREEN;
 info_nwindow_1.FirstGadget= NULL;
 info_nwindow_1.CheckMark  = NULL;
 info_nwindow_1.Screen     = screen_1;
 info_nwindow_1.BitMap     = NULL;
 info_nwindow_1.MinWidth   = 320;
 info_nwindow_1.MinHeight  = 13;
 info_nwindow_1.MaxWidth   = 320;
 info_nwindow_1.MaxHeight  = 13;

info_window_1=(struct Window *)OpenWindow(&info_nwindow_1);

 info_nwindow_2.LeftEdge = 0;
 info_nwindow_2.TopEdge  = 200;
 info_nwindow_2.Width    = 320;
 info_nwindow_2.Height   = 13;
 info_nwindow_2.DetailPen = 0;
 info_nwindow_2.BlockPen = 1;
 info_nwindow_2.Title    = NULL;
 info_nwindow_2.Flags    = SIMPLE_REFRESH|ACTIVATE;
 info_nwindow_2.IDCMPFlags = VANILLAKEY;
 info_nwindow_2.Type       = CUSTOMSCREEN;
 info_nwindow_2.FirstGadget= NULL;
 info_nwindow_2.CheckMark  = NULL;
 info_nwindow_2.Screen     = screen_2;
 info_nwindow_2.BitMap     = NULL;
 info_nwindow_2.MinWidth   = 320;
 info_nwindow_2.MinHeight  = 13;
 info_nwindow_2.MaxWidth   = 320;
 info_nwindow_2.MaxHeight  = 13;

 info_window_2=(struct Window *)OpenWindow(&info_nwindow_2);
}

object_drawer()
{
 int loop,pind;
 int abbruch = FALSE;

 zoom = 10;
 fzoom = (float)zoom;
 fzoom = (float)(fzoom/10);

 SetOPen(rp_1,0L);
 SetOPen(rp_2 ,0L);
 SetAPen(rp_3,3L);
 SetAPen(rp_4,3L);

 info_text();

 CONTINUE
 {
  transform();

  /* Die schnellste Art, die Bitmaps eines Screens zu clearen: */

  BltClear(rp_1->BitMap->Planes[0],8000L,0L);
  BltClear(rp_1->BitMap->Planes[1],8000L,0L);
  BltClear(rp_1->BitMap->Planes[2],8000L,0L);
  BltClear(rp_1->BitMap->Planes[3],8000L,0L);
/* Au3ere Zeichenschleife, indiziert nach der Anzahl d. Flachen */

  for(loop=0;loop<flaech_zahl;loop++)
  {
   if(sichtbar[loop])
   {
    SetAPen(rp_1,farbe[loop]);
    AreaMove(rp_1,sp[plst[loop][0]],ze[plst[loop][0]]);

/* Innere Zeichenschl., indiziert n. d. Anzahl der Punkte  */
/* der Flache, die gerade gezeichnet wird                  */

    for(pind=1;pind<pkt_p_flaech[loop];pind++)
     AreaDraw(rp_1,sp[plst[loop][pind]],ze[plst[loop][pind]]);

/* Wenn alle Punkte verbunden sind, Flache schlie3en u. fullen */

    AreaEnd(rp_1  );
    sichtbar[loop] = FALSE;
   }
  }
  ScreenToFront(screen_1);

  abbruch = lies_tastatur();
  if(abbruch) break;

  transform();

  BltClear(rp_2->BitMap->Planes[0],8000L,0L);
  BltClear(rp_2->BitMap->Planes[1],8000L,0L);
  BltClear(rp_2->BitMap->Planes[2],8000L,0L);
  BltClear(rp_2->BitMap->Planes[3],8000L,0L);

  for(loop=0;loop<flaech_zahl;loop++)
  {
   if(sichtbar[loop])
   {
    SetAPen(rp_2,farbe[loop]);
    AreaMove(rp_2,sp[plst[loop][0]],ze[plst[loop][0]]);
    for(pind=1;pind<pkt_p_flaech[loop];pind++)
     AreaDraw(rp_2 ,sp[plst[loop][pind]],ze[plst[loop][pind]]);
    AreaEnd(rp_2  );
    sichtbar[loop] = FALSE;
   }
  }
  ScreenToFront(screen_2);
 }
}

init_trig()
{
 int loop;

 for(loop=0;loop<360;loop++)
 {
  sinus[loop] = ((int)(sin(loop*PI/180)*16384));
  cosin[loop] = ((int)(cos(loop*PI/180)*16384));
 }
}

ausstieg()
{
 FreeMem(area_plane,8000L);
 CloseWindow(window_1);
```

```
 CloseWindow(window_2);
 CloseWindow(info_window_1 );
 CloseWindow(info_window_2);
 CloseScreen(screen_1 );
 CloseScreen(screen_2);
 CloseLibrary(GfxBase);
 CloseLibrary(IntuitionBase);
 exit(TRUE);
}

transform()
 {
 int loop;
 long u,v,z,x,y,hz,hx,hy,vg_wert1,vg_wert2;
 long va_x,va_y,va_z,vb_x,vb_y,vb_z,normv_z;
 long trans_x[MAXP],trans_y[MAXP],trans_z[MAXP];

 obj_xturn+= xdreher;
 if(obj_xturn<0) obj_xturn+= 360;
 if(obj_xturn>359) obj_xturn-= 360;
 obj_yturn+= ydreher;
 if(obj_yturn<0) obj_yturn+= 360;
 if(obj_yturn>359) obj_yturn-= 360;
 obj_zturn+= zdreher;
 if(obj_zturn<0) obj_zturn+= 360;
 if(obj_zturn>359) obj_zturn-= 360;

 for(loop=0;loop<pkt_zahl ;loop++)
 {
  /* Drehung um x-Achse   */
  hx = cx[loop];
  hy = cosin[obj_xturn]*cy[loop] - sinus[obj_xturn]*cz[loop];
  hz = sinus[obj_xturn]*cy[loop] + cosin[obj_xturn]*cz[loop];

  hy/= 16384;
  hz/= 16384;

  /* Drehung um y-Achse   */
  x = cosin[obj_yturn]*hx - sinus[obj_yturn]*hz;
  y = hy;
  z = sinus[obj_yturn]*hx + cosin[obj_yturn]*hz;

  x/=16384;
  z/=16384;

  /* Drehung um z-Achse   */
  hx = cosin[obj_zturn]*x - sinus[obj_zturn]*y;
  hy = sinus[obj_zturn]*x + cosin[obj_zturn]*y;
  hz = z;

  hx/= 16384;
  hy/= 16384;

  trans_x[loop] = hx;
  trans_y[loop] = hy;
  trans_z[loop] = hz;

  hz = (long)((hz+FAK)/2048 );

  u=(long)((hx/hz)*fzoom);
  v=(long)((hy/hz)*fzoom);

  sp[loop]=160+u;
  ze[loop]=100+v;
 }

 for(loop=0;loop<flaech_zahl ;loop++)
 {
  /* Berechnung der Komponeten des A- und B-Vektors  */

  va_x = trans_x[plst[loop][0]] - trans_x[plst[loop][1]];
  va_y = trans_y[plst[loop][0]] - trans_y[plst[loop][1]];
  va_z = trans_z[plst[loop][0]] - trans_z[plst[loop][1]];

  vb_x = trans_x[plst[loop][2]] - trans_x[plst[loop][1]];
  vb_y = trans_y[plst[loop][2]] - trans_y[plst[loop][1]];
  vb_z = trans_z[plst[loop][2]] - trans_z[plst[loop][1]];

  /* Berechnung der Z-Komponente des Normalenvektors  */

  normv_z = (va_x * vb_y) - (va_y * vb_x);

  vg_wert1 = normv_z+trans_z[plst[loop][1]];
  vg_wert2 = trans_z[plst[loop][1]];
  if(vg_wert1 > vg_wert2) sichtbar[loop] = TRUE;
 }
}

int lies_tastatur()
{
 int abbruch = FALSE;

 switch(info_window_2->MessageKey->Code)
 {
  case '8':xdreher+=1;
           write_int(xdreher,30L,9L,rp_3);
           write_int(xdreher,30L,9L,rp_4);
           break;
  case '2':xdreher-=1;
           write_int(xdreher,30L,9L,rp_3);
           write_int(xdreher,30L,9L,rp_4);
           break;
  case '9':zdreher+=1;
           write_int(zdreher,150L,9L,rp_3);
           write_int(zdreher,150L,9L,rp_4);
           break;
  case '1':zdreher-=1;
           write_int(zdreher,150L,9L,rp_3);
           write_int(zdreher,150L,9L,rp_4);
           break;
  case '4':ydreher+=1;
           write_int(ydreher,90L,9L,rp_3);
           write_int(ydreher,90L,9L,rp_4);
           break;
  case '6':ydreher-=1;
           write_int(ydreher,90L,9L,rp_3);
           write_int(ydreher,90L,9L,rp_4);
           break;
  case 'g':zoom+= 1;
           write_int(zoom,235L,9L,rp_3);
           write_int(zoom,235L,9L,rp_4);
           fzoom = (float)zoom;
           fzoom = fzoom/10;
           break;
  case 'k':if(zoom>0) zoom-= 1;
           write_int(zoom,235L,9L,rp_3);
           write_int(zoom,235L,9L,rp_4);
           fzoom = (float)zoom;
           fzoom  =  fzoom/10;
           break;
  case 'n':info_window_2->MessageKey->Code = NULL;
           file_loader();
           break;
  case 'q':abbruch = TRUE;
           break;
  default :break;
 }
 info_window_2->MessageKey->Code = NULL;
 return(abbruch);
}

file_loader()
{
 int count,loop,pos_zaehl,singlechar;
 char eingabe[19], *append = ".3d\0";

 pos_zaehl = 0;
 SetRast(rp_4,0L);
 Move(rp_4, 10L,9L);
 Text(rp_4,"Filename:",9L);
 ScreenToFront(screen_2);

 /* Einlesen des Filenamens    */
 do
 {
  if(info_window_2->MessageKey->Code)
  {
   singlechar = info_window_2->MessageKey->Code;
   if(singlechar == 13) break;
   if(singlechar !=  8)
   {
    eingabe[pos_zaehl] = (char)singlechar;
    ++pos_zaehl;
    Move(rp_4,90L,9L);
    Text(rp_4,&eingabe,(long)pos_zaehl);
    info_window_2->MessageKey->Code = NULL;
   }
   else
   {
    if(pos_zaehl>0)
    {
     --pos_zaehl;
     eingabe[pos_zaehl] = '\0';
     Move(rp_4,90L,9L);
     Text(rp_4,"                ",15L);
     Move(rp_4,90L,9L);
     Text(rp_4,&eingabe,(long)pos_zaehl);
     info_window_2->MessageKey->Code = NULL;
    }
   }
  }
 }
 while((singlechar != 13) && (pos_zaehl<14));

 info_window_2 ->MessageKey->Code = NULL;
 for(count=pos_zaehl;count<(pos_zaehl+4);count++)
  eingabe[count] = append[count-pos_zaehl];
 filename = &eingabe[0];
 filenumber = open(filename,O_RDONLY);
 if(filenumber<0)
 {
  SetRast(rp_4,0L);
  Move(rp_4,20L,9L);
  Text(rp_4,"File kann nicht geoeffnet werden!",33L);
  Delay(200L);
  SetRast(rp_4,0L);
  info_text();
  write_int(xdreher,30L,9L,rp_4);
  write_int(zdreher,150L,9L,rp_4);
  write_int(ydreher,90L,9L,rp_4);
  write_int(zoom,235L,9L,rp_4);
  write_int(xdreher,30L,9L,rp_3);
  write_int(zdreher,150L,9L,rp_3);
  write_int(ydreher,90L,9L,rp_3);
  write_int(zoom,235L,9L,rp_3);
  ScreenToFront(screen_1);
 }
 else
 {
  pkt_zahl = lies_int();
  flaech_zahl = lies_int();
  for(count=0;count<pkt_zahl;count++)
  {
   cx[count] = (long)lies_int();
   cy[count] = (long)lies_int();
   cz[count] = (long)lies_int();
  }
  for(count=0;count<flaech_zahl;count++)
```

```
   pkt_p_flaech[count] = lies_int();
  for(count=0;count<flaech_zahl;count++)
   for(loop=0;loop<pkt_p_flaech[count];loop++)
    plst [count][loop] = lies_int();
  for(count=0;count<flaech_zahl;count++)
   farbe[count] = (long)lies_int();
  close(filenumber);
  obj_xturn = 0;
  obj_yturn = 0;
  obj_zturn = 0;
  xdreher = 0;
  ydreher = 0;
  zdreher = 0;
  zoom = 10;
  fzoom = 1.0;
  SetRast(rp_3,0L);
  SetRast(rp_4,0L);
  info_text();
  ScreenToFront(screen_1);
 }
}

int lies_int()
{
 int wert,loop,pos_zaehl;
 pos_zaehl = 0;

 do
 {
  read(filenumber,zielchar,1);
  if(zielchar[0] == '-')
  {
   zielstring[pos_zaehl] = zielchar[0];
   ++pos_zaehl;
  }
  if(isdigit((int)zielchar[0]))
  {
   zielstring[pos_zaehl] = zielchar[0];
   ++pos_zaehl;
  }
 }
 while((zielchar[0] != ';') && (zielchar[0] != '\0'));
 wert = atoi(zielstring);
 for(loop=0;loop<10;loop++)
  zielstring[loop] = '\0';
 return(wert);
}

write_int(int_wert,output_xpos,output_ypos,ziel_port)
int int_wert;
long output_xpos,output_ypos;
struct RastPort *ziel_port;
{
 char output_string[8];
 int int_wert_buffer,zahl,s_factor,d_counter;
 int pos_zaehl;
 int int_wert_stellen;

 pos_zaehl = 0;
 int_wert_stellen = 5;

 if(int_wert<0)
 {
  output_string[pos_zaehl++] = '-';
  int_wert_buffer = -int_wert;
 }
 else
  int_wert_buffer =  int_wert;

 for(d_counter=10000;d_counter>=1;d_counter/=10)
 {
  if(int_wert_buffer/d_counter)break;
  int_wert_stellen -= 1;
 }

 switch(int_wert_stellen)
 {
  case 5:s_factor = int_wert_buffer / 10000;
         output_string[pos_zaehl++] = 48 + s_factor;
         int_wert_buffer -= s_factor * 10000;
  case 4:s_factor = int_wert_buffer / 1000;
         output_string[pos_zaehl++] = 48 + s_factor;
         int_wert_buffer -= s_factor * 1000;
  case 3:s_factor = int_wert_buffer / 100;
         output_string[pos_zaehl++] = 48 + s_factor;
         int_wert_buffer -= s_factor * 100;
  case 2:s_factor = int_wert_buffer / 10;
         output_string[pos_zaehl++] = 48 + s_factor;
         int_wert_buffer -= s_factor * 10;
  case 1:s_factor = int_wert_buffer / 1;
         output_string[pos_zaehl++] = 48 + s_factor;
         int_wert_buffer -= s_factor * 1;
         break;
  case 0:output_string[pos_zaehl++] = 48;
 }
 output_string[pos_zaehl] = '\0';
 Move(ziel_port,output_xpos,output_ypos);
 Text(ziel_port,"     ",4L);
 Move(ziel_port,output_xpos,output_ypos);
 Text(ziel_port,&output_string,(long)pos_zaehl);
}

info_text()
{
 SetRast(rp_3,0L);
 SetRast(rp_4,0L);
 Move(rp_3,10L,9L);
 Text(rp_3,"X:",2L);
 Move(rp_3,70L,9L);
 Text(rp_3,"Y:",2L);
 Move(rp_3,130L,9L);
 Text(rp_3,"Z:",2L);
 Move(rp_3,190L,9L);
 Text(rp_3,"Zoom:",5L);
 Move(rp_3,235L,9L);
 Text(rp_3 ,"10",2L);
 Move(rp_4,10L,9L);
 Text(rp_4,"X:",2L);
 Move(rp_4,70L,9L);
 Text(rp_4,"Y:",2L);
 Move(rp_4,130L,9L);
 Text(rp_4,"Z:",2L);
 Move(rp_4,190L,9L);
 Text(rp_4,"Zoom:",5L);
 Move(rp_4,235L,9L);
 Text(rp_4,"10",2L);
}

/*****************   Ende des 3d-Object-Animators  ***************/


   Nachfolgendend noch zwei Objektbeispiele ( bitte getrennt
        eingeben, benennen und mit dem Animator laden


/************ Eine Art stilisiertes Raumschiff ****************.**/

9;          Anzahl Punkte
9;          Anzahl Flaechen

 x        y        z      Koordinaten von [Anzahl Punkte]

 0;      -1000;     500;
-1000;   -1000;     0;
 0;      -1000;    -500;
 1000;   -1000;     0;
 0;       0;        1000;
-2000;    0;        0;
 0;       0;       -1000;
 2000;    0;        0;
 0;       3500;     0;

 Anzahl der Punkte [Flaechen]

 4; 4; 4; 4; 4; 3; 3; 3; 3;

 Punktliste [Anzahl Flaechen] [Punktzahl pro Flaeche]

 0; 1; 2; 3;
 0; 4; 5; 1;
 1; 5; 6; 2;
 2; 6; 7; 3;
 0; 3; 7; 4;
 4; 8; 5;
 4; 7; 8;
 7; 6; 8;
 5; 8; 6;

 Farbliste [Anzahl Flaechen]

 3; 9; 9; 9; 9;11;11;11;11;

/*********************************************************************/

/************ Ein Wurfel mit Innenflachen ***********************/

16;    14;

 2000;   2000;  2000;      -2000;   2000;  2000;
-2000;  -2000;  2000;       2000;  -2000;  2000;
 2000;   2000; -2000;      -2000;   2000; -2000;
-2000;  -2000; -2000;       2000;  -2000; -2000;
 1500;   1500;  2000;      -1500;   1500;  2000;
-1500;  -1500;  2000;       1500;  -1500;  2000;
 1500;   1500; -1000;      -1500;   1500; -1000;
-1500;  -1500; -1000;       1500;  -1500; -1000;

 4;4;4;4;4;4;4;4;4;4;4;4;4;4;

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

  8;11;3;4;5;6;7;9;10;11;12;13;14;15;
```

Dem AZTEKEN auf der Spur

Mit der neuen Compilerversion 3.4 liefert die Firma Manx nun ein Programmierwerkzeug, das C-Programmierer aufhorchen läßt.

Das Compilerpaket wird in der hier getesteten 'kleinen' Version auf zwei 3,5-Zoll-Disketten geliefert. Das zugehörige Handbuch ist in einem stabilen Schuber untergebracht. Es bezieht sich auf die etwas ältere Version 3.2, alle notwendigen Ergänzungen wurden jedoch mit eingeheftet. Der Benutzer kann sich also sein Handbuch 3.4 leicht selbst zusammenstellen. Die Dokumentation ist sehr ausführlich, wenn auch in englischer Sprache geschrieben. Sie geht allerdings nur auf die programmspezifischen Teile des Pakets ein (Datenformate, unterstützte Funktionen, Bedienung von Compiler und Linker usw.); als Lehrbuch für C ist sie natürlich nicht gedacht.

Der erste Eindruck

Nachdem problemlos eine Sicherheitskopie gezogen werden konnte, bootete der Rechner (A500, A1000 und A2000) direkt von der mit #1 gekennzeichneten Diskette. Als Minimalkonfiguration ist ein Amiga mit 512 KByte RAM, am besten mit zwei Laufwerken, notwendig. Der Compiler kommt im Betrieb mit erfreulich geringem Speicher aus, so daß schon ab einem Megabyte mit der zeitsparenden Ramdisk gearbeitet werden kann. Keine Probleme hat der Besitzer einer Festplatte: Er kann das komplette Programmpaket auf dem schnellen Speichermedium installieren und so die turn-a-round Zeiten erheblich verkürzen.

Unverträglichkeiten

Die Zeit, die vom Editieren bis zum fertigen Programm vergeht, ist gerade bei der Umsetzung von Quelltexten, die mit anderen Systemen entwickelt wurden (z. B. Lattice), entscheidend. Diese Tatsache liegt zum größten Teil am Datenformat der Integerzahlen (ganze Zahlen). Der Aztec-Compiler benutzt aus Gründen der Geschwindigkeitsoptimierung 16 Bit Integerzahlen. Das Konkurenzprodukt aus dem Hause Lattice arbeitet hier mit 32 Bit Datenbreite. Dies führt besonders bei Zeigerarithmetik und großen Schleifenzählern (z.B. for(i = 0;i < 10000000;i + +)) zu falschen Rechnungen oder Endlosschleifen. Außerdem wird von vielen Public-Domain-Quellen der Fehler 'pointer to int conversion' erzeugt. Man kann dieses Problem jedoch durch eine 32 Bit Integer-Option des Compilers (Beispiel: cc +L quelle.c) umgehen. Dies erfordert jedoch auch einen Linkvorgang mit der 32 Bit Library (Beispiel: ln quelle.o -lc32). Da diese Vorgehensweise sowohl Zeit als auch Speicher kostet, sollte ein anderer Weg eingeschlagen werden, dieses Problem zu lösen:

- Alle an Systemroutinen übergebenen Werte müssen als long-Werte vorliegen. (Nachgestelltes 'L', z. B. 500 wird zu 500L oder (long) z.B. fehler = OpenDevice(...) wird zu (long)fehler = OpenDevice(...))

- Definition aller Systemroutinen als (long). (Kann automatisch durch das Includen von functions.h erreicht werden.)

Kompatibel

Mit Quelltexten von anderen Aztec-Implementierungen ergaben sich jedoch keine ernsten Probleme. Wenn man sich bei seinen Programmentwicklungen an den Sprachkern von Aztec C hält, ist das größte Problem die Umsetzung der Quelltexte auf die unterschiedlichen Diskettenformate.

Neuheiten

Die Compilersoftware läuft in der neuen Version mit der Workbench V1.2. A1000-Besitzer müssen ihr System dazu unbedingt mit Kickstart V1.2 hochfahren. Eine Installation ist jedoch auch auf Systemen mit der Version 1.1 möglich. Kopieren Sie dazu bitte alle compilereigenen Dateien der Diskette #1 auf eine bootfähige Diskette der Version 1.1. Kopieren Sie unter keinen Umständen Systemdateien, z. B. die CLI-Befehlsdateien; dies führt mit Sicherheit nach Indien.

Schneller

Der Compiler liefert im Gegensatz zur alten Version einen schnelleren und auch etwas kompakteren Code. Dies fällt besonders im Vergleich mit dem Lattice-Compiler angenehm auf.

Vielseitiger

Ebenfalls wird jetzt die 32 Bit Integer-Option vom Compiler vollständig unterstützt. Bei den Gleitkommazahlen stehen drei Formate zur Auswahl: Motorola Fast Floating Point, IEEE double precision Emulation und 68881 IEEE double precision Emulation.

Der Sprachschatz wurde um die Aufzählungstype 'enum' erweitert.

Ein Bonbon

Ein besonderer Leckerbissen ist der im Paket enthaltene Assembler. Er arbeitet mit dem Compiler eng zusammen. Dieser Umstand erlaubt es, mit Hilfe der Preprozessor-Anweisungen #asm und #endasm den Maschinencode direkt in den C-Quelltext einzubinden. Im Gegensatz zu den Möglichkeiten, die z. B. die

Inline-Funktion in Turbo Pascal bietet (nur Eingabe der schon übersetzten Hexcodes), kann hier der Assemblerteil einfach in der Motorola Mnemonic-Schreibweise eingefügt werden. Dabei gilt die gleiche Syntax wie im Standalone-Betrieb des Assemblers.

Der Assembler ist als eigenständiges Programmierwerkzeug anzusehen. Tabelle 1 zeigt eine Übersicht über die Fähigkeiten des Assemblers. Dabei werden die Prozessoren 68000, 68010, 68020 und 68881 unterstützt. Die neue Version arbeitet jetzt auch problemlos mit den von Commodore vertriebenen Headerfiles zusammen.

Die Liste aller verfügbaren Assemblerdirektiven gibt einen Einblick in Leistungsfähigkeit dieses Werkzeuges.

```
EQU         label     equ       <expression>

REG         label     reg       <register list>

PUBLIC    [label]     public    <symbol>

GLOBAL    [label]     gobal     <symbol>,<size>

BSS       [label]     bss       <symbol>,<size>

ENTRY     [label]     entry     <symbol>

END

CSEG

DSEG

DC          [label]   dc.b      <value>
            [label]   dc        <value>
            [label]   dc.w      <value>
            [label]   dc.l      <value>
            [label]   dc.b      "string"

DCB         [label]   dcb.b     <size>[,<value>]
            [label]   dcb       <size>[,<value>]
            [label]   dcb.w     <size>[,<value>]
            [label]   dcb.l     <size>[,<value>]

DS          [label]   ds.b      <size>
            [label]   ds        <size>
            [label]   ds.w      <size>
            [label]   ds.l      <size>

NEAR         near     code\data

FAR          far      code\data

LIST

NOLIST

MLIST

NOMLIST

CLIST

NOCLIST

INCLUDE   include    <file>

MACRO    [label]     macro     <symbol>

ENDM

MEXIT

IF

ELSE

ENDC
```

Gut gelinkt

Die größte Neuerung beim aktuellen Linker ist das veränderte Objektformat. Dies bedeutet für alle Umsteiger von älteren Versionen, daß sie ihre bis dato erstellten Objekt- und Librarymodule noch einmal mit der neuen Compilerversion übersetzen müssen, um sie weiter verwenden zu können.

Allerdings unterstützt der Linker nun Objektmodule und Libraries im Amiga-Standardformat (dies sind alle Module, die mit Alink bzw. Blink zusammenarbeiten). Als Dreingabe wurde die Linkzeit im Vergleich zur Vorgängerversion noch ein wenig verkürzt. Ebenso werden nun im Quelltext redefinierte Labels bemerkt und angezeigt.

Eingemachtes

Der Compiler unterstützt alle Standardgrundtypen von Variablen.

– char
8 Bit breit. Mit oder ohne Vorzeichen. Wird mit einer Char-Variable gerechnet, wird sie auf 16 Bit Integer erweitert. signed char wird auf einen Bereich von – 127 bis + 127 gebracht, unsigned wird zu 0 bis + 255.

– short
16 Bit breit. Ebenfalls signed und unsigned. Voreingestellt ist signed.

– int
16 Bit oder 32 Bit breit (je nach + L-Option). Signed und unsigned. Voreingestellt ist signed.

– long
32 Bit breit. Signed und unsigned. Voreingestellt ist signed.

– float
32 Bit breit. Es werden jedoch 64 Bit Speicher belegt.

– double
Wie float.

– pointer
32 Bit breit.

Für Experten

Um alle Fähigkeiten des Assemblers ausschöpfen zu können, muß auf den Inhalt einiger Register geachtet werden. Die Register D0 – D3 sowie A2 und A3 sind für Registervariablen verwendbar. D4 – D7 enthalten Zwischenwerte während der Berechnung von Ausdrücken. A4 ist für Programme reserviert, die das 'small memory model' unterstützen. Bei diesen Programmen werden Daten und Sprünge relativ zu A4 adressiert bzw. ausgeführt. Diese Maßnahme macht die Programme schneller und kürzer. A5 enthält den Stackframe-Zeiger der momentan ausgeführten Funktion. Die Register A0, A1 und A6 dienen als temporäre Zwischenspeicher. A7 zeigt auf die Stackspitze des Programms. Werden Assemblerteile innerhalb einer C-Funktion verwendet, sollten möglichst alle verwendeten Register restauriert werden.

Schwächen

Das Compilerpaket ist in der vorliegenden 'kleinen' Version ein durchaus brauchbares Programmierwerkzeug. Manx hätte allerdings seinen bei den teureren Versionen mitgelieferten Debugger auch in dieses Paket aufnehmen sollen. Gerade der Hobbyprogrammierer wird am Anfang einige Probleme mit dem Compiler haben.

Zudem wäre ein Handbuch für den deutschsprachigen Raum sicher eine gute Sache. Es ist sonst eigentlich nicht zu bemängeln. Übrigens ist in ihm ein kleiner verkaufsstrategischer Trick versteckt: Im Handbuch sind alle Tools der mächtigeren und natürlich auch teureren Versionen aufgeführt. Der Anwender soll also Lust auf die großen Versionen bekommen, zumal der Einfachheit halber nur die neuen Disketten gekauft werden müssen – das Handbuch ist ja bereits komplett.

Fazit

Der Käufer erhält jedoch insgesamt 'viel Compiler' für 'viel Geld'. Der zur Zeit mit rund 430 Mark recht teure Compiler bietet im Gegensatz zu der billigeren Lattice-Lösung zum Teil erheblich schnelleren Code und auch kürzere Programme – auch wenn der unbedarfte C-Programmierer ob der genannten Inkompatibilitäten in ernsthafte Schwierigkeiten geraten kann.

(GC)

+	schneller und kompakter Code
+	wahlweise 16 Bit Integer (schneller)
+	verbesserter Linker (auch Alink-kompatibel)
+	Paket erweiterbar

–	relativ hoher Preis
–	kein Debugger enthalten
–	Schwierigkeiten bei Systemaufrufen (Commodore verwendet 32 Bit Integer in den Systemlibraries)

Endlich kann man dem AmigaBasic Beine machen: Der AC/Basic-Compiler macht es möglich. Er ist dabei sehr anwenderfreundlich und vielseitig. Die meisten Basic-Programme ließen sich auf Anhieb compilieren und sind nun selbständig lauffähig, d. h. sie benötigen weder einen Compiler noch AmigaBasic, um ausgeführt zu werden.

Der AC/Basic-Compiler ist bis auf wenige Ausnahmen kompatibel mit dem AmigaBasic-Dialekt. Deshalb bietet er sich dafür an, mit diesem Basic erstellte Programme zu compilieren. Der Quelltext für den Compiler muß nicht mit AmigaBasic erstellt werden, diese Aufgabe kann jeder beliebige Editor erfüllen (z. B. der System-Editor ED).

Compilieren – ganz einfach

Um Programme, die mit AmigaBasic erstellt wurden, compilieren zu können, müssen sie im ASCII-Format vorliegen. Dazu werden sie von AmigaBasic mit SAVE „filename",A abgespeichert. Der Editor ED speichert generell nur ASCII-Dateien, aber bei vielen anderen (Text-)Editoren ist es notwendig, diese Option speziell anzugeben. Danach wird die Datei in den Compiler geladen. Bei den meisten Programmen können die Grundeinstellungen des Compilers belassen werden. Der Compiler arbeitet danach die verschiedenen Stufen durch und erzeugt ein Programm auf Diskette. Dieses kann nun direkt angewählt und gestartet werden.

Außerdem...

Natürlich ist das Compilieren von Programmen nicht immer so einfach wie gerade beschrieben, außerdem liefert das System noch erheblich mehr Informationen, die unter Umständen von Bedeutung sein könnten. Die wichtigsten Informationen erhält man beim Durchlaufen von PASS 1 und 3 (der Compilier-Vorgang wird in mehreren Abschnitten – pass – durchgeführt). Sie beziehen sich fast ausschließlich auf den verwendeten Speicherplatz und sind deshalb von besonderer Bedeutung (siehe auch Bild 2).

'Total'
gibt dabei den Speicherplatz an, den das Programm insgesamt benötigt, was vor allem beim Multitasking wichtig ist.

'Complete'
gibt die Größe des Programms an.

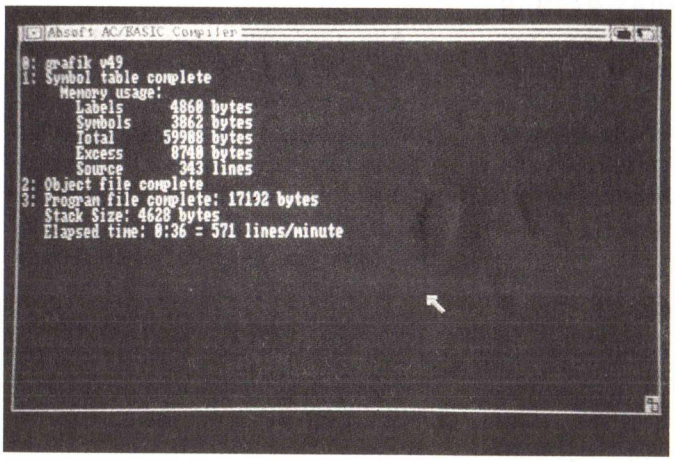
'Stack Size':
Diese Größe ist besonders wichtig, denn der Stack ist ein Speicher für wichtige Systemdaten, wie z. B. Rücksprungadressen von Unterprogrammen. Wenn dieser Bereich zu klein gewählt wird, dann kommt es unweigerlich zu einem Systemzusammenbruch. CLI-Benutzer sollten dies immer bedenken, denn hier wird keine automatische Größenänderung vorgenommen wie von der Workbench aus.

Sollten bei diesen Vorgängen Fehler auftreten, so wird der Durchlauf abgebrochen und die Fehler werden im Klartext ausgegeben.

OPTIONS & METACOMMANDS

Der Compiler bietet eine Vielzahl von Options an, in die man sich mit der Zeit einarbeiten sollte, um die optimalen Ergebnisse in Bezug auf die Programmgröße und vor allem die Ausführungsgeschwindigkeit zu erreichen.

Option 'A':
Use Long Adressing In bestimmten Situationen benötigt der Compiler mehr als 16 Bit zur Adressierung. Wenn dies der Fall ist, so meldet er sich.



Das Menü des Compilers

Option 'C':
Enabel Run-time Tests Diese Option ist besonders in der Entwicklungsphase eines Programmes wichtig, weil hier das compilierte Programm dazu veranlaßt wird, weitreichende Fehlerüberprüfungen vorzunehmen, was natürlich zu Zeitverlusten führt.

Option 'D':
Compile for Decimal Math Der Compiler unterstützt zwei Darstellungsarten für Gleitkommazahlen: Binär und dezimal. Der Interpreter arbeitet jedoch nur binär, deshalb bleibt dieser Punkt meist unberührt.

Option 'N':
Process Run-time Events Einige AmigaBasic-Befehle bedienen sich der Interruptsteuerung (z.B. ON TIMER/ MENU/MOUSE/ COLLISION/ERROR/BREAK GOSUB...). Um diese Ereignisse verwalten zu können, muß die Option eingeschaltet sein.

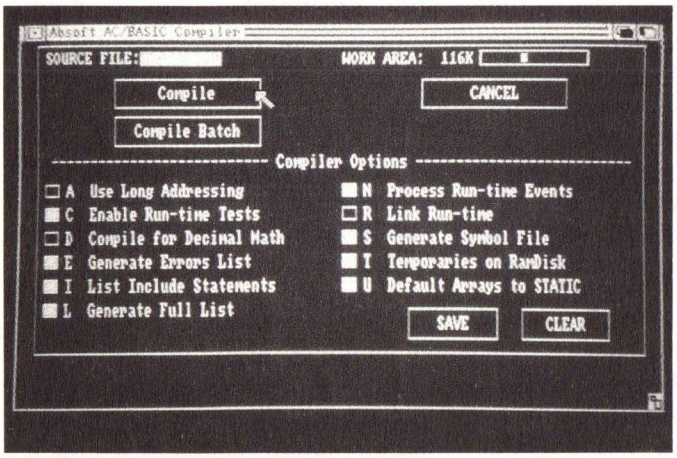
Opiton 'R':
Link Run-time-Library Ein compiliertes Programm benötigt zu seiner Ausführung zusätzlich eine der Runtime-Bibliotheken (BAS.RL/BAS. DL). Um ein unabhängig vom Compiler laufendes Programm zu erhalten, muß diese Bibliothek zum Programm 'dazugelinkt' werden. Man erhält dadurch ein einziges Programm, das, auf eine andere Diskette kopiert, lauffähig ist.

Option 'U':
Default Arrays to STATIC AmigaBasic läßt zu, daß Felder innerhalb des Programms gelöscht und neu dimensioniert werden und somit statisch (STATIC) sind.

Option 'N':
Generate Full List Der Compiler generiert eine komplette Liste des Quell-Programms (eventuell mit Fehlermeldung). Außerdem wird eine Cross-Reference-Liste erstellt.

Option 'E':
Generate Error List Zu jeder fehlerhaften Zeile wird eine Fehlermeldung ausgegeben.

Option 'I':
List Include Statements Wenn diese Option aktiviert ist, dann werden auch die Include-Files ausgegeben.



Compiliervorgang am Beispiel eines Amiga Basic-Programms

Option 'S':
Generate Symbol Files Diese Option erzeugt eine Symbolliste, die zum Debuggen benötigt wird.

Option 'T':
Temporaries on Ramdisk Der Compiler legt die Zwischencodes auf der Ramdisk ab, wodurch der Vorgang erheblich schneller geht als beim Zwischenspeichern auf Diskette.

Alle Optionen lassen sich auch über METACOMMANDS steuern, d. h. sie können innerhalb des Listings an einer beliebigen Stelle an- und wieder ausgeschaltet werden, so daß nur ein bestimmter Bereich des Programms mit dieser Option behandelt wird. Ein weiterer Befehl erlaubt es, bestimmte Programmbereiche vom Compiler nicht bearbeiten zu lassen. Damit können unter Umständen die AmigaBasic-Befehle umgangen werden, die der Compiler nicht verarbeiten kann. Eine sehr hilfreiche Option des AC/Basic-Compilers ist, daß man beim Compilieren andere Basic-Programme hinzufügen kann. Dies können vollständige und separat entwickelte Module sein oder aber auch Variablendeklarationen, die in mehreren Programmen verwendet werden.

CLI und Batchfiles

Der AC/Basic-Compiler kann auch vom CLI aus bedient werden. Dies führt auch dazu, daß er sich mit Batchfiles ansprechen läßt, wodurch sich mehrere Programme hintereinander, mit verschiedenen Optionen, compilieren lassen.

Abweichungen

Einige der Einschränkungen, die sich durch manche Einstellmöglichkeiten des Compilers ergeben, wurden bereits unter dem Punkt 'Options' angesprochen.

Einige Befehle des Interpreters werden jedoch vom Compiler nicht erkannt: CONT, DELETE, LIST, LLIST, LOAD, MERGE, NEW, SAVE, TRON, TROFF.

Der Compiler bietet außerdem eine weitere Kontrollstruktur (SELECT CASE), die der Interpreter nicht unterstützt. Wichtig ist auch, daß Unterprogramme (SUB) auch dynamisch angelegt werden können, denn dadurch sind auch rekursive Aufrufe möglich.

Einige andere AmigaBasic-Befehle sind, meist mit kleineren Änderungen, an den Compiler anzupassen, allerdings ist es mit diesem Wissen nicht schwer, Programme gleich kompatibel zu gestalten.

Benchmarks

Der Geschwindigkeitsvorteil, den ein compiliertes Programm gegenüber einem interpretierten hat, kann recht erheblich sein. Dies zeigen vor allem die Benchmark-Tests, die jedoch hauptsächlich arithmetische Operationen und Datenmanipulationen testen. Bei Systemaufrufen und auch bei Diskettenzugriffen nähern sich die Zeiten jedoch stark an, denn entweder wird von beiden, dem Compiler und dem Interpreter, die gleiche Systemroutine aufgerufen, oder die physikalischen Eigenschaften der Hardware, z. B. des Diskettenlaufwerks, lassen sich nicht verändern.

Test	AmigaBasic	AC/Compiler
Bench1	0.42	0.078
Bench2	1.68	0.10
Bench3	3.84	0.48
Bench4	4.52	0.40
Bench5	4.92	0.48
Bench6	8.82	1.40
Bench7	13.68	3.18
Bench8	17.40	3.70
Summe	55.99	10.06
Sieb	55.06	4.84

Fazit

Die Bedienung des AC/Basic-Compilers ist im Grunde sehr einfach. Selbst Benutzer, die noch nie mit einem Compiler gearbeitet haben, können ihre mit AmigaBasic erstellten Programme problemlos compilieren. Basic-Befehle wie PEEK oder POKE können jedoch Probleme aufwerfen, die sich nicht so leicht beseitigen lassen. Doch wer damit ein wenig hantiert, der wird auch eine Lösung für den Compiler finden. Das Resultat ist ein stattlicher Zeitgewinn bei der Ausführung der Programme. Außerdem entfällt das Laden des Basic-Interpreters und der damit verbundene Zeitaufwand.

Dem fortgeschrittenen Programmierer bietet der Compiler ein wertvolles Hilfsmittel zur Optimierung von Programmen. Die Geschwindigkeit des Compiliervorgangs wird diejenigen, die nur den Basic-Interpreter kennen, sicher etwas stören, die Zeiten sind jedoch durchaus akzeptabel.

Diese recht kurzen Zeiten kommen auch daher, daß das gesamte Compilersystem sehr kompakt ist. Dadurch entfallen lange Ladezeiten. Die Dokumentation des AC/Basic-Compilers ist in englischer Sprache und sehr umfangreich. Der erste Teil zeigt die Bedienung und den Umgang mit den Optionen, während der zweite eine komplette Referenz des Sprachumfangs ist. In den Anhängen werden Datenformate, der Aufruf von Maschinen-Unterprogrammen und die Fehlermeldungen erläutert. Der Compiler ist ein ausgereiftes und zuverlässiges Produkt, die Dokumentation ist sehr ausführlich und mit vielen Programmen versehen, die sich auch auf der Diskette befinden. Die Compilierzeiten sind gut, und auch der Preis ist, verglichen mit anderen Compilern, angemessen. Das Produkt kann deshalb jedem Basic-Programmierer empfohlen werden, der das nicht gerade schnelle AmigaBasic auf Trab bringen will. Außerdem bietet sich mit dem Compiler die Möglichkeit, Programme weiterzugeben oder zu verkaufen, ohne gleich den Quell-Text preiszugeben.

(mn)

```
REM ***** SIEB *****
DEFINT a-z
DIM zahlen(8190)
start=TIMER

zaehler=0
FOR i=0 TO 8190
   zahlen(i)=1
NEXT i
FOR i=0 TO 8190
   IF zahlen(i) THEN
      prim=i+i+3
      k=i+prim
      WHILE k<=8190
         zahlen(k)=0
         k=k+prim
      WEND
      zaehler=zaehler+1
   END IF
NEXT i

PRINT TIMER-start
PRINT zaehler
```

\+ kompatibel zu AmigaBasic und Microsoft-Basic
\+ schelle Ausführungszeiten
\+ unabhängige Programme
\+ Einbinden anderer Module
\+ erlaubt Rekursion

– kleine Abweichungen zu AmigaBasic
– kein eigener Editor

AC/Basic-Compiler

RAAB-Bürotechnik
Freidhofstr. 36
8605 Hallstadt
09 51 / 7 18 48

Preis: 398,– DM

Einkaufsführer

Hier finden Sie Ihren Commodore/Amiga Fachhändler

Anzeigenschluß Heft 10/87: 21. August 1987

1000 Berlin

RUNOW
Büroelektronik
Keithstraße 26 · 1000 Berlin 30
☎ 26 111 26

2000 Hamburg

Bit Computer Shop
Osterstraße 173 · 2000 Hamburg 20
Telefon: 040/494400
Createam
Computer Hard & Software
Bramfelder Chaussee 300 · 2000 Hamburg 71
Telefon Sa. Nr. 040/641 50 91

Gerhard u. Bernd Waller GbR
Computer & Zubehör-Shop
Kieler Straße 623
2000 Hamburg 54
☎ 040/570 60 07 + 570 52 75

GMA mbH
Systemhändler
Wandsbeker Chaussee 58
2000 Hamburg 76

2160 Stade

BERGAU
Büromaschinen · EDV-Systeme
Neue Straße 5, 2160 Stade
Telefon: (04141) 23 64 + 23 84

2390 Flensburg

ecl electronic computer laden ohg
Norderstr. 94–96 · D-2390 Flensburg
(0461) 28181 & 28193

2800 Bremen

Berthge & Strutz KG
St.-Jürgen-Straße 46 – 50
2800 Bremen 1
Tel. 04 21 - 70 00 57 - 59

2940 Wilhelmshaven

Radio Tiemann
Commodore-Systemfachhändler
Markstr. 52
2940 Wilhelmshaven
Telefon 0 44 21 - 2 61 45

2960 Aurich

FRANZ-JOSEF KIRFEL
GEORGSWALL 18a
2960 AURICH 1
TELEFON 0 49 41 - 6 23 55

3000 Hannover

COM DATA
Am Schiffgraben 19 · 3000 Hannover 1
Telefon 05 11 - 32 67 36

3250 Hameln

Witte Bürotechnik
Inh. Günther Kessels
Deisterstr. 53
3250 Hameln 1
Tel. 0 51 51 - 1 20 22 / 1 20 23

3470 Höxter

Schidlack + Sohn KG
Am Markt 8 · 3470 Höxter
Telefon 0 52 71 - 79 29 / 29 29

3500 Kassel

Hermann Fischer GmbH
Commodore-Systemfachhändler
Rudolf-Schwander-Str. 5-13
3500 Kassel
Tel. (05 61) 70 00 00

4000 Düsseldorf

CCS Commodore
Computer Schule GmbH
Immermannstr. 65
4000 Düsseldorf 1
Tel. 02 11 - 35 32 15

Hier könnte
Ihre Anzeige
erscheinen
Anruf genügt: Heim - Verlag
Tel. 0 61 51 - 5 60 57

4200 Oberhausen

LaSch
das Buch und Software Haus
Inh. Rainer Langner u. Franz Schnitzler GbR
Nohlstraße 76 · 4200 Oberhausen 1
Telefon 02 08 / 80 90 14

4600 Dortmund

Elektronik
Computer
Fachliteratur
AMIGA-Software
4600 Dortmund 1, Güntherstraße 75, Tel. (02 31) 57 22 84
city-elektronik

4650 Gelsenkirchen-Horst

ComTron

Hard- und Software, Literatur
Bauteile, Service, Versand
Groß- und Einzelhandel
Poststr. 15 · 4650 Gelsenkirchen-Horst
Tel. 0209/52572

4800 Bielefeld

hardware software organisation service **CSF**

CSF COMPUTER & SOFTWARE GMBH
Heeper Straße 106–108
4800 Bielefeld 1
Tel. (05 21) 6 16 63

5000 Köln

BÜRO MASCHINEN **braun**
AM RUDOLFPLATZ GmbH
5000 KÖLN 1
RICHARD-WAGNER-STR. 39
RUF: 02 21/21 91 71

5060 Bergisch-Gladbach

Computer Center
Buchholzstraße 1
5060 Bergisch-Gladbach
Telefon 0 22 02 - 3 50 53

5200 Siegburg

Computer Center
Luisenstraße 26
5200 Siegburg
Telefon 0 22 41/6 68 54

5500 Trier

bürocenter **LEHR**
Güterstr. 82 - 5500 Trier
☎ 06 51 - 2 50 44
Fordern Sie unsere Zubehör-Liste an.

5768 Sundern

C.S.C.
Computer & Software-Center

Hauptstr. 2 · 5768 Sundern
Telefon 0 29 33 - 20 46

6000 Frankfurt

GES-COMPUTER
GESELLSCHAFT FÜR EDV UND SOFTWARE mbH

Filiale Frankfurt
Hartmann-Ibach-Str. 63
6000 Frankfurt 60
Tel.: (0 69) 46 20 41

Filiale Hanau
Steinheimer Str. 22
6450 Hanau
Tel.: (0 61 81) 2 48 26

Hardware Software Service
Büro-Computer + Organisations GmbH
Computer bco
Commodore TOSHIBA ATARI OKIDATA
Ihr Partner, wenn es um Computer geht
6000 FRANKFURT/M. 1, Oeder Weg 7-9, ☎ 069/55 04 56/57

6200 Wiesbaden

Henneveld KG

Schossbergstr. 21
6200 Wiesbaden
Tel. 0 61 21 - 27 70

6250 Limburg

wir
bürosysteme vertriebs gmbh

diezer strasse 10
6250 limburg
tel. 0 64 31 - 2 00 30

6300 Gießen

Ihre Tür zur Zukunft:
KARSTADT **computer-center**
hardware·software·problemlösungen
Gießen, Seltersweg 64, Telefon (06 41) 70 04 - 318

6380 Bad-Homburg

PDC GmbH
Produkte u. Details Computerverbund

Louisenstr. 115
Ladenpassage Alter Bahnhof
6380 Bad-Homburg
Tel. 0 61 72 - 2 47 48

Bad-Homburgs erster Commodore Computerladen

6457 Maintal

Landolt-Computer
Beratung · Service · Verkauf · Leasing
Autorisierter Commodore-Händler
Wingertstr. 112 · 6457 Maintal/Dörnigheim
Telefon 0 61 81 - 4 52 93

6500 Mainz

Henneveld KG

Münsterstr. 15
6500 Mainz
Tel. 0 61 31 - 24 01

6520 Worms

Georg Steinmetz oHG

Neumarkt 4 + 10 · 6520 Worms
Telefon 0 62 41 - 68 68

6680 Neunkirchen

Shop 64
Computer GmbH

Saarbrücken * Saarlouis
Homburg * St. Ingbert

Neunkirchen
06821/23713
Commodore Systemhändler

6750 Kaiserslautern

Gotthold Bürocenter - Ihr Partner für's Büro
Jac. Gotthold Nachf. KG · Kaiserslautern · Im Altstadtparkhaus · Tel. 6 30 71

6800 Mannheim

Computer-Center
am Hauptbahnhof GmbH

L 14, 16-17
6800 Mannheim 1
Tel. (06 21) 2 09 83 / 84

GAUCH+STURM
Computersysteme + Textsysteme
6800 Mannheim 24
Casterfeldstraße 74-76
☎ (06 21) 85 00 40 · Teletex 6 211 912

Wo ist der Amiga-Fachhändler in meiner Umgebung ?

Hier könnte Ihre Anzeige erscheinen
Rufen Sie uns an: Heim-Verlag 06151/56057

7000 Stuttgart

»If AMIGA, go to Schreiber«
Stuttgart's starker Computer-Laden.
SCHREIBER ELEKTRONIK
Alte Poststraße 2, 7000 Stuttgart-City
1. Stock, Telefon (0711) 22 70 99

7140 Ludwigsburg

B D T
BÜRO-DATEN-TECHNIK-VERTRIEBS GMBH
Kurfürstenstraße 18 · 7140 Ludwigsburg
Telefon 0 71 41 - 2 50 74

7500 Karlsruhe

MCT
Ges. f. Microcomputer
Troilingerstr. 3 · 7500 Karlsruhe 41
Tel. 07 21 / 47 27 95

7700 Singen

Schellhammer GmbH
Freibühlstr. 21 + 23
7700 Singen
✆ 0 77 31 / 82 02 - 00

7890 Waldshut-Tiengen

hettler-data
service gmbh
Lenzburger Straße 4
7890 Waldshut-Tiengen
Telefon 0 77 51 / 30 94

7900 Ulm

EDV-Systeme
Software-erstellung
Schulung

Systemhaus:
Frauenstr. 28
7900 Ulm/Donau
Tel. 07 31 / 2 80 76
Telex 712 973 csulm-d

COMPUTER STUDIO GMBH

7920 Heidenheim

Das Büro
Kastorstr. 11
7920 Heidenheim/Brenz
✆ 0 73 21 - 4 40 15

7990 Friedrichshafen

KUMATRONIK Datentechnik
Schwabstr. 19
7990 Friedrichshafen
Telefon 0 75 41 - 7 20 41

8000 München

ludwig
COMPUTER + BÜROTECHNIK
COMPUTER · SOFTWARE · PERIPHERIE
BERATUNG · TECHN. KUNDENDIENST
INGOLSTÄDTER STR. 62L
EURO-INDUSTRIE-PARK · 8000 MÜNCHEN 45
TELEFON 089/3113066 · TELETEX 898341

8120 Weilheim

COMPUTER STUDIO HUTTER GMBH
MÜNCHNER STR. 12 · 8120 WEILHEIM
TELEFON 08 81 / 12 23

8330 Eggenfelden

Hot Space
Computer-Centrum
R. Lanfermann
Schellenbruckstraße 6
8330 Eggenfelden
Telefon 0 87 21 / 65 73
Altöttinger Straße 2
8265 Neuötting
Telefon 0 86 71 / 7 16 10
Innstraße 4
8341 Simbach
Telefon 0 85 71 / 44 10

8390 Passau

Zimmermann
elektroland
8390 Passau
Kohlbruck 2a
☎ 08 51 / 5 20 07

8400 Regensburg

Computer-Laden
Karl Steinmetz
Gewerbepark C 62 · 8400 Regensburg
Telefon 09 41 - 4 82 99

Wo ist der Amiga-Fachhändler in meiner Umgebung ?

Hier könnte Ihre Anzeige erscheinen
Rufen Sie uns an: Heim-Verlag 06151/56057

8400 Regensburg

Zimmermann
elektroland
8400 Regensburg
Dr.-Gessler-Str. 8
☎ 09 41 / 9 50 85

8460 Schwandorf

Büro- u. Datensysteme GmbH
Ettmannsdorfer Str. 8
8460 Schwandorf
Tel. (0 94 31) 2 04 78 / 4 18 43

8500 Nürnberg

Orgaplus Datenverwaltung
G. Gailer KG
Fürther Str. 54 – 56
8500 Nürnberg 80
✆ 09 11 - 27 06 20

8600 Bamberg

BÜRO-ZENTRUM
A+R KUTZ
Am Kranen Tel. 0951 / 27808
8600 Bamberg

8700 Würzburg

SCHOLL
BÜROTEAM
Hardware · Software
Service · Schulung
computer center
am Dominikanerplatz
Ruf (0931) 50488

Schweiz

CH-8021 Zürich

Senn Computer AG
Langstrasse 31
Postfach
CH-8021 Zürich
Tel. 01 / 241 73 73
Telex 814 193 seco

Kleinanzeigen

BIETE HARDWARE

AMIGA 1000 Echtzeituhr-Modul Immer die richtige Uhrzeit für 100,– DM!! Neu 190,– 2 Monate alt! – Mit Akku-Pufferung und SUPER-Software.
Tlf. ab 19.30 Uhr 0208/22283

Notverkauf: **Amiga 1000** 512 KB 1081-Bildschirm, ca. 30 Disk. Disk-Box, Profi-Joystick, Maus Orig. Software (z. B. Pascal), Bücher, Zeitschriften, **DM 1999** Tel. 089/328949 Michael

---

Amig 1000 mit Sidecar, Festplatte, 1,5 MB viel Software
Tel.: 07159-8127

BIETE SOFTWARE

Public-Domain-Disketten
von Fred Fish für 4,– DM das Stück abzugeben. Letzte Ausgabe ist z. Zt. die Nr. 80
Tel.: 030/404 17 93 – abends – !

Amiga Public Domain Service
M Disk aus der ganzen Welt
I Franz Schnitzler
G Telefon: 0201/292000
A Brandhoffs-Delle 11, 43 Essen

Amiga Soft Für Amiga-Freaks 30 P.D Disk 180,- DM 0201/281301

Amiga Public Domain Disks Info-Disk = 10 DM / Beschr. der Prg's über 250 Screens / wird durch 1 bzw. 2 Disks ersetzt / Alt-User. M. Roenn, 3257 Springe 4, Ziegeleiweg 32 / 050418229 ab 19 Uhr

---

Public-Domain ★ über 200 Disk vorrätig! Fred Fish u.v.a. Info-Disk gegen DM 5,– anfordern. Ossowski, Veronika 33, 43 Essen

---

Amiga P.D Soft 30 Disk 180,- DM 60 Disk nur 300 DM 0201/281301

SUCHE SOFTWARE

500er sucht Soft. 02204/68231

TAUSCH

Suche, Tausche, Kaufe, Verkaufe Amiga-Public Domain. Info = 10 DM. Beschr. der Prg's über 250 Screens wird durch 1 Disk ersetzt. Abgabepreis bis zu 5 DM (je) Alt, Ziegeleiweg 32, 3257 Springe 4

KONTAKTE

Ich suche Kontakt zu anderen Amiga Programmers. Spez. Raum HD-MA. T. 06227-4699 Bernhard

---

Suche, u. benötige beste Software für den Amiga 2000!! Kontakte im In- u. Ausland erwünscht. Melden b. Reinhold: 02564/33640

Suche Amiga User in Hannover. Tel.: 0511/650240

VERSCHIEDENES

AMIGA, Erfa.-Austausch mit Grafikern gesucht (Raum FFM), Demo-Disk mit Business-Grafiken und Gestaltungs-Ideen, doppelseitig voll, DM 25,- incl., wer tauscht Graf.-Software, Info 06081/8590

Hilfe. Suche Druckertreiber für NEC P5XL-Amiga 2000. Wer weiß Rat? Suche Amiga-User. Konzakze. T. 08042-1231 abends

INSERENTENVERZEICHNIS

Impressum

KICKSTART

Chefredakteur:
Uwe Bärtels (Chefredakteur) (UB)
Markus Nerding (Stellvertreter) (MN)

Redaktion
Andreas Krämer (AK)
Gerald Carda (GC)
Dipl. Ing. Harald Schneider (HS)
Marcelo Merino (MM)
Harald Egel (HE)

Herausgeber:
„MERLIN" Computer GmbH
Industriestraße 26
Postfach 55 69
6236 Eschborn
Tel.: 0 61 96 / 48 18 11
FAX: 0 61 96 / 4 11 37

Ständige Mitarbeiter:
Christian Keller (CHK)
Christian Schormann (CS)
Wolf Dietrich (WD)
Roland Foerster (RF)

Public Relations:
Claus Peter Lippert

Verlag:
Heim Verlag
Heidelberger Landstraße 194
6100 Darmstadt 13
Tel.: 0 61 51 / 5 60 57
FAX: 0 61 51 / 5 56 89

Verlagsleitung:
Hans-Jörg Heim

Anzeigenverkaufsleitung:
Uwe Heim

Anzeigenpreise:
nach Preisliste Nr. 1, gültig ab 1.7.86

Produktion:
Klaus Schultheis
Patricia Illing

Grafische Gestaltung:
Rosina Altkorn

Fotografie:
Klaus Ohlenschläger
Markus Nerding

Titelgestaltung:
Fabian & Maier

Titelfoto:
Rainer Spirandelli

Satz:
Mohr Pfungstadt

Druck:
Ferling Druck Darmstadt

Bezugsmöglichkeiten:
Zeitschriftenhandel, Kauf- und Warenhäuser, Commodore-Fachhändler oder direkt beim Verlag

Kickstart erscheint 11 mal im Jahr
Einzelpreis: DM 7,–, ÖS 56,–, SFr. 7,–
Das Jahresabonnement kostet DM 70,– inkl. Versandkosten + MwSt.
Europ. Ausland DM 90,– inkl. Versandkosten

VORSCHAU:

Software:

TAURUS
eine mächtige relationale Datenbank

DeLuxe Video II
– was bringt die neueste Version?

Buchter
– Zeichenprogramm und Bilderconverter

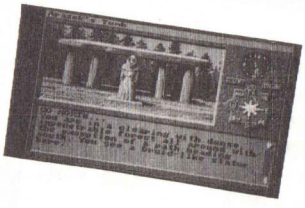
Hardware:

Wir machen Druck: Druckertests

Festplatten, einige Speicherriesen im Test

Listings:

BAM-Copy V1.0
...und einige Listings in AmigaBasic



Spiele:

KING OF CHICAGO,
das neue Superspiel von CINEMAWARE

ARAZOCS TOMB,
das erste Spiel von AEGIS

KARATE KID II, KARATE KING und andere...

Ab 18. September an Ihrem Kiosk!

Großer *AMIGA*-ST-Programmierwettbewerb

1. Preis:

30.000.- DM

Die Amiga- und ST-Welle rollt – und wir sind voll dabei. Wir haben bereits eine komplette Palette an Spielprogrammen vorgestellt und es geht weiter. Dazu suchen wir noch Programmierer, die professionelle Spiele (in C oder Assembler) schreiben können. Eine internationale Jury wird nach Einsendeschluß alle Programme bewerten und einen Hauptgewinner küren, der *30000,- DM* in bar erhält.
Achtung! Auch die Einsender, die nicht den 1. Platz erreichen und ein interessantes Programm geschrieben haben, erhalten bei uns *keinen Trostpreis*, sondern ein *attraktives Angebot!* Jeder Teilnehmer erhält außerdem als Dankeschön AMIGA- bzw. ST-Software von uns geschenkt, Einsendeschluß ist der 30.9.1987. Bitte fordern Sie noch heute die genauen Teilnahmebedingungen an. Nutzen Sie die Chance, international bekannt und (erfolg)reich zu werden wie schon viele andere Programmierer aus unserem Haus (z.B. Henrik Wening – Autor der ersten deutschen Nr. 1 in England – oder Udo Gertz – letztes Jahr von englischen Zeitungen mit 3 Oskars und 2 Awards ausgezeichnet).

SPITZEN-SOFTWARE

KINGSOFT

MADE IN GERMANY

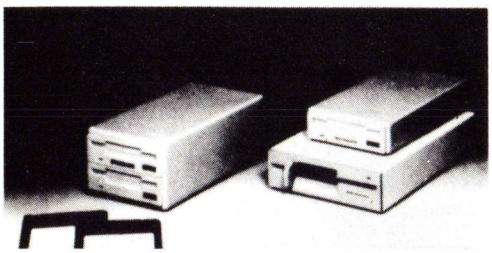
F. Schäfer · Schnackebusch 4 · 5106 ROETGEN
Telefon 0 24 08-51 19 · Telefax 0 24 08-52 13

AMIGA™

Soyka Datentechnik
Bochum 02 34 / 41 19 13

AMIGA™

Original AMIGO Made in Germany



F_1 3,5"-Einzelfloppy, anschlußf.	369,–	DM
F_2 3,5"-Doppelfloppy, anschlußf.	669,–	DM
F_3 5,25"-Einzelfloppy, anschlußf.	489,–	DM
Amigo-Sound (Mono)	129,–	DM
Amigo-Sound (Stereo)	199,–	DM
Amigo-Bootselector	39,90	DM
Amiga 500, 1 MB Erweiterung inkl. Uhr	299,–	DM
Amiga 2000, 1,5 MB Erweiterung	198,–	DM
Amiga 2000 Zweitlaufwerk	279,–	DM

NEC 1036 A + Interface + Kabel + Stecker
+ Anleitung zum Selbstbau einer 3,5"-

Amiga-Floppy 289,– DM

– 3,5"-Gehäuse, Kunststoff, beige mit Befestigungsmaterial	34,90	DM
– NEC 1036A/NEC 1035LP, 1 MB, 3,5"	259,–	DM
– NEC Multisync Monitor	1444,–	DM

SOFTWARE

SPIELESOFTWARE

Alien Fires	84,–
Amiga Karate	67,–
Arazok's Tomb	87,–
Arena	77,–
Barbarian	77,–
Bard's Tale	97,–
Chessmaster 2000	87,–
Cruncher Factory	27,–
Defender of the Crown	87,–
Demolition	27,–
Earl Weaver Baseball	77,–
Faery Tale	97,–
Fire Power*	47,–
Flightsimulator II	99,–
Flip Flop	37,–
Galactic Invasion*	47,–
Galaxy Fight	67,–
Goldrunner	87,–
Grand Prix*	87,–
Gunship*	87,–
Karate Kid II	77,–
Karate King	37,–
King of Chikago*	107,–
Land of Legends*	97,–
Mindbreaker	37,–
Pac Boy	37,–
Phalanx	27,–
Return to Atlantis*	97,–
Rocket Attack	37,–
S.D.I	99,–
Shanghai	77,–
Shooting Star	37,–
Silent Service	87,–
Sinbad	95,–
Sky Fighter	47,–
Space Fight	37,–
Starglider	77,–
Strip Poker	77,–
Surgeon	117,–
Terrorpods*	77,–
Turbo	47,–
Uninvited	97,–
Quiwi	57,–
Wintergames	57,–
Worldgames	57,–

ANWENDERSOFTWARE

Acquisition	597,–
Aegis SONIX V2.0	167,–
Aztec C Dev. V3.4a	597,–
Aztec C Com. V3.4a	897,–
City Desk	247,–
CLI Mate V1.2	67,–
Deluxe Music Con. Set	197,–
Deluxe Paint IIB	247,–
Deluxe Video V1.2	247,–
Dynamic Cad	997,–
Dynamic Word*	377,–
Express Paint*	187,–
Galileo	207,–
Grabbit	67,–
Instant Music	97,–
Lattice C V3.1	347,–
Lisp	417,–
LPD Writer	247,–
LPD Filer*	247,–
LPD Planer*	247,–
Macro Assembler	197,–
Marauder II	107,–
Music X*	547,–
Metacomco Pascal	197,–
Modula II Standard	187,–
Modula II Developers	297,–
Pagesetter europ. Ver.	327,–
Planetarium*	97,–
Printmaster Plus	107,–
Scribble	197,–
K.-Seka Assembler	137,–
Shell (Cli deluxe)	137,–
Superbase (deutsch)	227,–
Toolkit	97,–
UBM Text V2.2	247,–
Ucsd Pascal	167,–
Vip Professional	447,–
Vizawrite (deutsch)	197,–

* = in Kürze lieferbar

AMIGA REFERENCE MANUALS:

Hardware	62,50
Intuition	62,50
Exec	62,50
Libraries and Devices	88,–

PUBLIC DOMAIN SOFTWARE

10 Disks	89,–
30 Disks	249,–
inkl. Disketten	

Komplette Softwareliste mit ca. 300 Prg. anfordern!

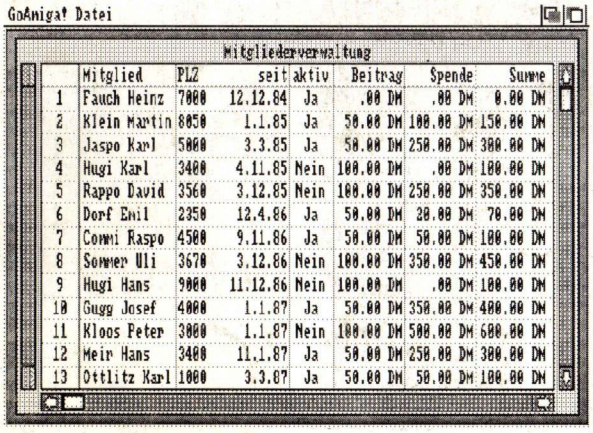
Harald Soyka ★ Hattinger Straße 685 ★ 4630 Bochum 5

GO AMIGA! Datei

DM 199.-

Die professionelle Dateiverwaltung!
Spezielle Serie von Datendisketten zusätzlich erhältlich

- *Bild + Tonverwaltung*
- *Listenmodus*
- *Formularmodus*
- *32000 Datensätze pro Datei*
- *32 Felder pro Datensatz*
- *10 Feldtypen*
- *unterstützt Umlaute*
- *Schnelles Sortieren + Suchen*
- *Multitasking*
- *unterstützt alle RAM*
- *kompatibel zu Harddisk*
- *Handbuch deutsch*



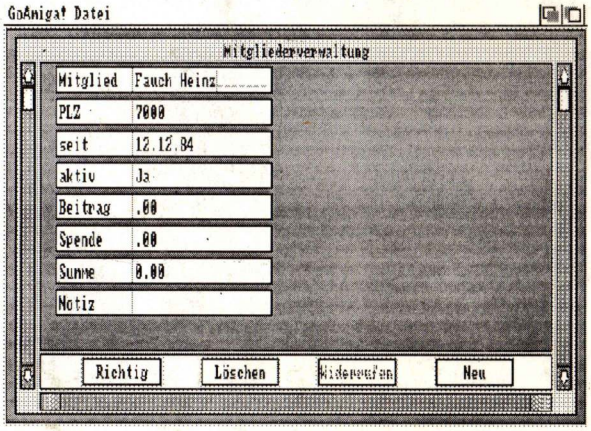
	Mitglied	PLZ	seit	aktiv	Beitrag	Spende	Summe
1	Fauch Heinz	7000	12.12.84	Ja	.00 DM	.00 DM	0.00 DM
2	Klein Martin	8050	1.1.85	Ja	50.00 DM	100.00 DM	150.00 DM
3	Jaspo Karl	5000	3.3.85	Ja	50.00 DM	250.00 DM	300.00 DM
4	Hugi Karl	3400	4.11.85	Nein	100.00 DM	.00 DM	100.00 DM
5	Rappo David	3560	3.12.85	Nein	100.00 DM	250.00 DM	350.00 DM
6	Dorf Emil	2350	12.4.86	Ja	50.00 DM	20.00 DM	70.00 DM
7	Commi Raspo	4500	9.11.86	Ja	50.00 DM	50.00 DM	100.00 DM
8	Sommer Uli	3670	3.12.86	Nein	100.00 DM	350.00 DM	450.00 DM
9	Hugi Hans	9000	11.12.86	Nein	100.00 DM	.00 DM	100.00 DM
10	Gugg Josef	4000	1.1.87	Ja	50.00 DM	350.00 DM	400.00 DM
11	Kloos Peter	3000	1.1.87	Nein	100.00 DM	500.00 DM	600.00 DM
12	Meir Hans	3400	11.1.87	Ja	50.00 DM	250.00 DM	300.00 DM
13	Ottlitz Karl	1000	3.3.87	Ja	50.00 DM	50.00 DM	100.00 DM

Listenmodus

- Einfacher Überblick aller Daten
- Freie Definition von Spaltenbreite, Anordnung und Spaltenbündigkeit (links-rechtsbündig, zentriert)
- Beträge anzeigen mit internationalen Währungen
- Schnelles Wechseln zwischen selektierter Liste und Gesamtliste
- Komfortable Druckersteuerung mit verschiedenen Schriften (Pica, Elite, Fine)
- Individueller Druck (Seitenhöhe, Seitenbreite, Kürzungen usw.)
- Automatisches Anwählen einer Telefonnummer

Formularmodus

- Individuelle Formulargestaltung durch einfaches Feldverschieben mit der Maus
- Variable Feldlänge, auch wenn Daten eingegeben
- Optionales Großschreiben des ersten Buchstabens
- Unabhängiges Abspeichern zum Listenmodus
- Help Taste
- Etiketten beliebig ausdruckbar (bis zu 16-bahnig)
- Automatische Fehlerkorrektur



Bestellservice:

BRD: 0041-1-3115959
CH: 01-3115959

Geschäftszeiten:
10.00–12.30, 13.30–18.30 Uhr, außer montags,
Sa.: 10.00–16.00 Uhr.

Versand ins Ausland nur Vorkasse (Scheck, bar)
zzgl. DM 6,– Porto. Händleranfragen erwünscht.

softwareland
Franklinstraße 27
CH-8050 Zürich (Schweiz)