

fast geschenkt!
GOLD-EDITION

RGH-PROFAN AUF CD-ROM!

fast geschenkt!

19⁸⁰
DM

GOLD-EDITION



RGH Profan 4.5

Vom Windows-Anwender
zum Programmierer!



ME

fast geschenkt!
GOLD-EDITION

Ausgabe 9

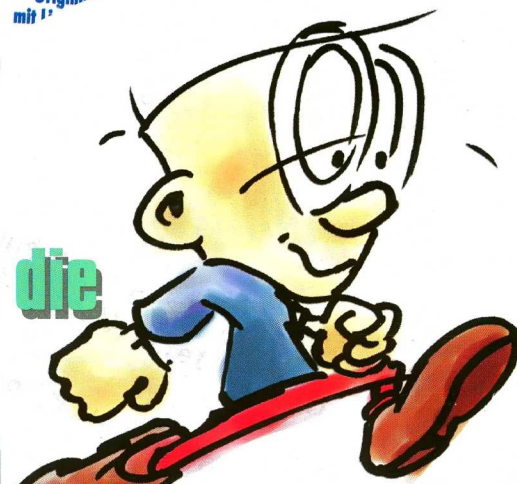
RGH Profan 4.5

Kommerziell
Original
mit 1

LIZENZIERT
DEUTSCHE
ORIGINALVERSION

Ihr Einstieg in die
Software-
Entwicklung
unter Windows

Programmierkurs im Heft



09

SHOPPING — Im — INTERNET

www.

Über 6.000 Artikel!

Länderspezifische Angebote für™



Der Zugang für **Endkunden** ist uneingeschränkt möglich. **Großhandelskunden** lassen sich bitte kostenlos registrieren. Sie erhalten dann Ihr Zugangs-Passwort zugestellt.
Gewerbenachweis / Behörden­siegel per Fax an: +49-7631-360-444

Entwicklungshelfer für den Einstieg



editorial



Programmieren unter Windows – ein Thema für sich? Zugegeben, liebe Leserin, lieber Leser: So einfach wie den Anwendern macht Windows es den Programmierern nicht. Die schöne bunte Oberfläche, die sich dem Anwender symbolhaft-intuitiv über Fenster und Dialoge erschließt, hat nicht nur die

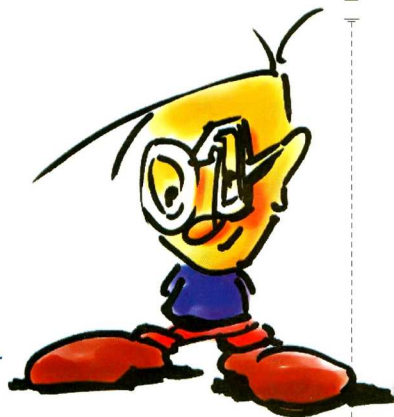
Programmanwendung, sondern auch die Programmentwicklung verändert. Das nicht mehr kommando-, sondern ereignisorientierte Betriebssystem zwingt zum Umdenken und provoziert automatisch einen anderen Programmierstil. Es genügt nicht mehr, einen linearen Problemlösungsweg festzulegen, der Schritt für Schritt abgearbeitet werden muß. Die interaktive Programmsteuerung im Dialog mit dem Anwender setzt voraus, daß alle Events und Eventualitäten von Anfang an in die Planung und Entwicklung einbezogen werden. Nicht mehr allein auf das Was, sondern auf das Wie kommt es an – wie der Benutzer sich im Programm zurechtfindet und wie er mit dem Programm seine Probleme zu lösen vermag.

Doch nicht nur Betriebssysteme, auch Programmiersprachen entwickeln sich weiter – im Konzept und Aufbau wie auch im Bedienungskomfort. Warum sollten nicht auch die Programmentwickler von den

Annehmlichkeiten einer grafisch orientierten Arbeitsumgebung profitieren? Kein Programmierer braucht heute noch Hunderte von Befehlen nebst allen syntaktischen Regeln und Formalitäten im Kopf zu haben! Wie Windowsprogramme haben auch Windows-Programmiersprachen heute eine Oberfläche vorzuweisen, die Funktionen und Werkzeuge und dazu ganz selbstverständlich auch eine Online-Hilfe und eine Referenz über Menüs und Schaltflächen bereitstellt.

Daß die Programmentwicklung auch unter Windows kein Reservat für Insider ist, beweist *RGH-PROFAN*. Die Programmiersprache, die Sie in diesem Heft kennenlernen werden, ist ein idealer *Entwicklungshelfer* für den Umstieg und Einstieg in die Windows-Programmierung. Besondere Vorkenntnisse sind nicht erforderlich. Sie brauchen nicht einmal zu wissen, wie Windows intern funktioniert. *RGH-PROFAN* ist so angelegt, daß die nötigen Kenntnisse und Erfahrungen – learning by doing – ganz *profan* in der Praxis erworben werden können. Was dabei zu beachten ist und wie Sie im einzelnen vorgehen, das erfahren Sie in diesem Heft!

Marianne Steible



IMPRESSUM

Redaktion: Dr. Marianne Steible (Chefredakteurin), Andreas Hett
Redaktionsanschrift: 79426 Buggingen, PEARL-Str. 1
Internet: <http://www.trendverlag.de>
E-Mail: fgred@trendverlag.de
Hotline: 07631-360-300
Heft-CD: Pearl Agency, Buggingen
Layout, Satz und Belichtung: Christian Lampe, STYLE OF THE ARTS,
 Grafische Medien-Produktionsgesellschaft mbH,
 Tel.: 07631-360-950
Illustrationen: Wolfgang Ruf
Titel: Michael H. Sichler
Verlag: TREND Redaktions- und Verlagsgesellschaft mbH
 PEARL-Straße 1, 79426 Buggingen
 Tel.: 07631-360-270, Fax: 07631-360-444
Vertrieb: TREND Redaktions- und Verlagsgesellschaft mbH
 PEARL-Straße 1, 79426 Buggingen

Vertriebsleitung: Reinhard Jansohn
Verlagskoordination: Anja Bek
Außendienst: TREND Büro West, Jägerstr. 23,
 46149 Oberhausen, Tel.: 0208-644110,
 Fax: 0208-644367
Betreuung Grossa/Bahnhoftsbuchhandel:
 Sylvia Czichollas, Holger Klein,
 Christian Linke, Damir Rogina
Heftproduktion:
 Stephan Striegel, Tel.: 07631-360-502;
 Vernon Warren, Tel.: 07631-360-512
Druck: Marco, Toulouse
Druckauflage: 60.000
Erscheinungsweise: bis zu acht Ausgaben im Jahr
Verkaufspreis: (incl. 15% Mwst.) DM 19,80
Bankverbindung:
 Spar- und Kreditbank Bad Krozingen,
 BLZ 680 632 54, Kto-Nr.: 159492

Bezugsmöglichkeiten:
 Bestellung nimmt der Verlag entgegen. Alle in dieser Zeitschrift veröffentlichten Beiträge sind urheberrechtlich geschützt. Kein Teil dieser Zeitschrift darf außerhalb der Grenzen des Urheberrechtsgesetzes ohne schriftliche Genehmigung des Verlags in irgendeiner Form reproduziert werden. Alle übrigen Rechte bleiben vorbehalten.
 Für unverlangt eingesandte Manuskripte, Fotos, Datenträger etc. wird keine Haftung übernommen. Die Rücksendung ist ausgeschlossen. Die von den Autoren einzelner Beiträge oder von Interviewpartnern aufgestellten Behauptungen und vertretenen Ansichten sowie der Inhalt der Anzeigen müssen nicht der Auffassung der Redaktion entsprechen. Der Verlag übernimmt außerhalb der gesetzlich vorgeschriebenen presserechtlichen Verantwortlichkeit keinerlei Haftung für die Richtigkeit der veröffentlichten Beiträge.



Die Heft-CD

Installation und Inhalt: Um *RGH-Profan* auf der Festplatte zu installieren, legen Sie die CD-ROM in Ihr CD-ROM-Laufwerk ein und starten Windows. Wenn Sie unter Windows 95 arbeiten, klicken Sie auf den Start-Button und rufen den Befehl *Ausführen* auf. Unter Windows 3.1(1) wählen Sie den Befehl *Ausführen* im DATEI-Menü des Programm-Managers oder im Datei-Manager.

In der folgenden Befehlszeile geben Sie

D:\START.EXE

ein (vorausgesetzt, Ihr CD-ROM-Laufwerk wird mit D: angesprochen). Darauf wird automatisch die CD-Menüoberfläche eingeblendet. Dort finden Sie folgende Auswahlfelder:

RGH-PROFAN	Ihr Einstieg in die Windows-Programmierung
Demos	Demos ausgewählter kommerzieller Programme
Virenschutz	Aktuellste Versionen der Thunderbyte Anti-Viren-Software
Bücher	Tips und Leseproben zu und aus aktuellen Bestsellern
Online	Vollversion von T-Online-Decoder
Musiktrack	Pidgin Styles „Chance“
Abo-Service	Ihr Vorteil: <i>Fast Geschenk Gold</i> im Abonnement

Per Mausklick auf das entsprechende Auswahlfeld können Sie die gewünschte Anwendung direkt aufrufen bzw. installieren. In der Setup-Routine von *RGH-Profan* öffnet sich zunächst ein Auswahl Fenster, aus dem Sie neben der Programminstallation auch zwei Demoverionen von PROFAN-Anwendungen – *PC-Gotchi* und *Bob Squish* – aufrufen können. Ein Klick auf *PROFAN 4.5a* startet das Setup-Programm, in dessen Verlauf Sie nur das Zielverzeichnis auf Ihrer Festplatte angeben müssen. Nach erfolgreicher Installation können Sie *RGH-PROFAN*-Editor mit dem gewohnten Klick auf das Icon starten.

Hinweis: Um einige Beispieldateien im *RGH-PROFAN*-Editor zum Laufen zu bringen, müssen Sie die Pfadangaben im Quellcode aktualisieren oder den Include-Pfad über den gleichnamigen Befehl im Menü *OPTIONEN* erweitern. Bitte lesen Sie die jeweiligen Readme-Dateien bzw. die im Quellcode enthaltenen Anweisungen aufmerksam durch.

Das Programmpaket: Der Umfang des Programmpakets *RGH-PROFAN 4.5* wurde exklusiv für die *FAST GESCHENKT GOLD-EDITION* um einige Verzeichnisse mit zusätzlichen Tools und Beispiel-Applikationen erweitert. Wenn Sie sich den Inhalt der Heft-CD über den Dateimanager bzw. den Explorer ansehen, finden Sie neben dem Verzeichnis *PROFAN* mit den Programmdateien von *RGH-PROFAN 4.5a* folgende Ordner:

BONUS Sieben Zusatzprogramme von verschiedenen Autoren, mit denen Sie die Entwicklungsmöglichkeiten von *RGH-PROFAN* erweitern (Erstellung eigener Toolbars, Fortschrittsbalken, Bubble-Help, DLLs zum Rechnen mit Datumsangaben, Entfernen und Laden von Windows-Modulen u.a.)

HISTORIE Vollversionen von *RGH-Profan 1.4* – lauffähig auch unter Windows 3.0! – und *RGH-Profan 2.6*

PROFAN6 Info- und Hilfedateien zum Update *RGH-PROFAN 6.0*

PROFPROG Zwölf Spiele und Anwendungen

TOOLS

Zusätzliche Tools für Programmierer: Eine Setup-Routine, eine Demoverision des Rechenprogramms *GRIDDY* und die *PRF-Tools™ 1.5* von Thomas Hölzer (*PRFTOOLS*), die *RGH-PROFAN* von der Optimierung des Quellcodes über einen RGB-Farbmixer bis zu einem komfortablen Icon-Manager wirkungsvoll unterstützen. Die Entwicklungsumgebung *VISUALIS* von OHS steht in einer Demoverision zur Verfügung.

SUPPORT AUS ERSTER HAND:

PROFAN IM INTERNET

Kommen Sie bei der Programmierung mit *PROFAN* nicht weiter? Dann nutzen Sie das *PROFAN*-Forum unter der Adresse

<http://www.conceptd.com/profan/board.htm>

wo Sie auf einen regen Austausch über Programmierfragen zwischen erfahrenen und auch weniger geübten *PROFAN*-Programmierern stoßen. Auch eine *NewsGroup* (de.comp.lang.misc) befaßt sich mittlerweile mit *PROFAN*. Darüber hinaus ist der *PROFAN*-Autor Roland G. Hüßmann selbst mit einer Homepage im Internet vertreten:

<http://profan.home.pages.de>
oder

<http://www.fortunecity.com/skyscraper/bradbury/116/index.html>

Hier finden Sie interessante Links, Downloads und aktuelle Infos zum Programm. Unabhängig davon können Sie sich auch direkt per E-Mail an den Autoren wenden:

rgH-soft@t-online.de

Beispielhaft für die zahlreichen *PROFAN*-Seiten im Internet sei auch eine Homepage genannt, an der ein *PROFAN*-Programmierer kaum vorbeikommen wird:

<http://members.aol.com/THoelzer/homepage.htm>

Für Berliner ist sicher auch die *WINWORLD*-Mailbox interessant, die am Wochenende von Freitag 08.00 Uhr bis Montag 07.00 Uhr zu erreichen ist:

PROFAN-SUPPORT-MAILBOX: 030-8549191

Die zahlreichen Tips und Tricks, die aktuellen Shareware-Versionen und die vielen Beispielprogramme im Quellcode lohnen einen Besuch allemal!!

CD-Defekt?



Sollte Ihre Heft-CD zerkratzt sein oder einen Material- oder Transportschaden aufweisen, erhalten Sie vom Verlag Ersatz. Beachten Sie bitte die im Lizenzvertrag (Seite 50) unter Punkt 3 aufgeführten Umtauschbedingungen, und schicken Sie die ausgefüllte Registrierung mit ein.

Inhalt

Programmeinführung:

Programmieren – ganz *profan*

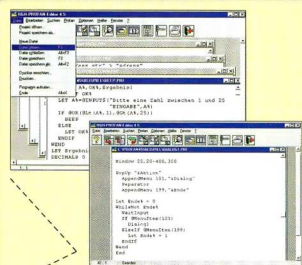
- Die Arbeitsumgebung: Dem Editor ist nichts zu schwer 7
- Die Befehlssyntax: Do you speak *Profan*? 8
- Fenster und Dialoge: Oberflächenkosmetik 10

Programmierkurs:

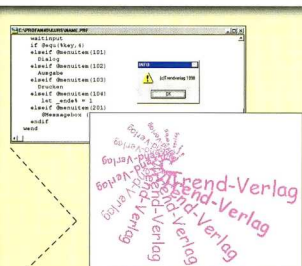
Der Sprung ins kalte Wasser

- Die Gestaltung des Programmfensters 44
- Der Aufbau des Menüs 44
- Die Definition des Dialogfensters 45
- Die Zusammenführung der Quelltexte 45
- Der Einbau der Events in die WHILE-Schleife 46
- Die Programmierung der Ereignisprozeduren 47
- Die Erstellung der fertigen EXE-Datei 48

- Editorial 3
- Die Heft-CD 4
- Das besondere Angebot für unsere Leser 49
- Lizenzkunde 50
- Impressum 3



Programmieren – ganz *profan*: In der Einführung erfahren Sie, wie die Programmiersprache aufgebaut ist.



Der Sprung ins kalte Wasser: Im Praxiskurs haben Sie Gelegenheit, Schritt für Schritt Ihr erstes Windows-Programm zu entwickeln.



A stylized, cartoonish illustration of a vintage computer system. It features a large CRT monitor with a blue screen, a keyboard, a mouse, and a floppy disk. The illustration is set against a dark blue background with a white curved line suggesting a desk surface.

beziehen: Dateneingabe, Datenverarbeitung und Datenausgabe. Wie aber, bitte schön, bringt man eine Maschine dazu, Programmweisungen in menschlicher Sprache entgegenzunehmen und auszuführen? Der Computer kann bekanntlich nur zwei Zeichen unterscheiden – die Eins und die Null. Die Eins signalisiert: Es gibt Strom, die Null: Es gibt keinen. Da die Sprachlogik des Computers nur diese beiden Signale kennt, bleibt auch die Kommunikation zwischen Mensch und Maschine auf dieses binäre System beschränkt. Keine sonderlich erweiternden Aussichten für angehende Programmierer – nur gut, daß es Programmiersprachen gibt, die Dolmetscher spielen! Um sich Ihrem Windows-Rechner verständlich zu machen und Anweisungen für eigene Windows-Programme zu formulieren, brauchen Sie keine endlosen Kolonnen von Nullen und Einsen einzugeben, sondern schalten ganz einfach einen solchen Dolmetscher ein, der darauf spezialisiert ist, Ihre Programmstrukturen entgegenzunehmen und in die Maschinensprache zu übersetzen.

Warum nicht *RGH-PROFAN*? Die Programmiersprache für Windows stellt alles bereit, was Sie brauchen, um ein lauffähiges Windows-Programm zu entwickeln:

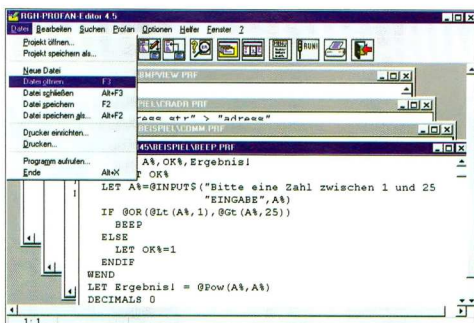
- einen Editor für die Eingabe des Programmcodes
- eine überschaubare Entwicklungsumgebung mit allen Grafikmöglichkeiten, die Windows bietet
- einen Interpreter zum Entwickeln und Testen
- einen Compiler und einen Linker, die den Programmcode in ein lauffähiges Windows-Programm verwandeln

Von den gängigen Windows-Programmiersprachen unterscheidet sich RGH-PROFAN durch den besonders einfachen und benutzerfreundlichen Zugschnitt. Selbst Anwender, die noch nie programmiert haben, bringen mit RGH-PROFAN problemlos ein einfaches kleines Windows-Programm zustande. Auch für Umsteiger, die von DOS zu Windows wechseln wollen, ist PROFAN geradezu prädestiniert: Da PROFAN-Programme ganz konventionell mit einem Editor geschrieben werden, in enger Anlehnung an die Befehlssyntax von BASIC, bereitet der Umstieg von DOS zu Windows sehr viel weniger Schwierigkeiten, als wenn ein objektorientiertes Entwicklungskonzept verfolgt wird.

Im übrigen braucht das anwenderfreundliche Konzept durchaus nicht als Einschränkung verstanden zu werden: Die Entwicklungsumgebung von RGH-PROFAN, die für die interaktive Programmsteuerung unter Windows eine ganze Reihe von vorgefertigten Kontrollelementen bereitstellt, gleichzeitig aber auch eine freie, von Helfern unterstützte Oberflächengestaltung zuläßt, nimmt Anfängern wie auch Fortgeschritten einiges an Programmieraufwand ab. Warum jeden Dialog und jede Listbox mühsam selbst erstellen, wenn fertige Komponenten da sind, die nur noch eingesetzt werden müssen? Für kleinere Windows-Programme reichen die mitgelieferten Standardelemente meist vollkommen aus; und wo maßgeschneiderte Dialoge und Menüs benötigt werden, lassen sich diese mit Unterstützung der Helfer ohne allzu großen Programmieraufwand auch in eigener Regie entwickeln. Daher unabhängig stellt RGH-PROFAN aber auch alle windowstypischen Grafikbefehle bereit, so daß auch versierte Windows-Programmierer ein reichhaltiges Repertoire von Entwicklungsmöglichkeiten vorfinden.

Über die Standardfunktionen hinaus hat **RGH-PROFAN** einige Extras anzubieten, die man hinter einem Einsteiger-Kit wohl kaum erwarten würde: Wer würde vermuten, daß **RGH-PROFAN** unter anderem ein SQL-fähiges Datenbank-Entwicklungssystem liefert, mit dem komplette, dBASE III-kompatible Datenbankanwendungen programmiert werden können? Auch für Multimedia-Fans hat **RGH-PROFAN** einiges zu bieten: Dank Unterstützung der MCI-Schnittstelle können alle in Windows integrierbaren Multimedia-Player angesteuert werden. Wieviel reizvoller ist die Programmierung von Anwendungen und vor allem von Spielen, wenn diese mit dem passenden Sound und mit attraktiven Geräuschkulissen unterlegt werden können!

Die Programmeinführung in diesem Heft muß sich freilich auf die Grundlagen der Programmierung mit **RGH-PROFAN** beschränken. Sie erfahren, wie **PROFAN**-Programme aufgebaut sind, und lernen die einzelnen Programmkomponenten sowie das Handwerkzeug für die Fenster- und Dialoggestaltung kennen. Nachdem Ihnen die Grundkomponenten und die Arbeitsumgebung schon ein wenig vertraut sind, haben Sie im im folgenden Workshop Gelegenheit, den Sprung in die Praxis zu wagen und Ihr erstes



Sieben auf einen Streich: Im Arbeitsfenster des Multi-File-Editors können mehrere Quelldateien parallel bearbeitet werden. So lassen sich Prozeduren und Programmschleifen problemlos kopieren und verschieben.

PROFAN-Programm selbst zu entwickeln. Wer freilich tiefer in das Programmiersystem eindringen möchte, sei hier schon auf die Einführung und den Programmierkurs im Unterverzeichnis \DOKU sowie auf die Online-Dokumentation verwiesen, die eine vollständige Sprachreferenz mit allen Variablen, Funktionen und Befehlen anbietet. Um sich einen ersten Eindruck zu verschaffen, was sich mit RGH-PROFAN alles anfangen läßt, sollten Sie auch nicht darauf verzichten, sich das DEMO-Programm und die vielfältigen kleinen BEISPIEL-Anwendungen anzusehen, die alle komplett mit RGH-PROFAN entwickelt wurden.

Die Arbeitsumgebung:

Dem Editor ist nichts zu schwer

Die Entwicklung eines Computerprogramms beginnt mit der Problemanalyse. Wenn Sie sich darüber im klaren sind, was für ein Problem Sie lösen wollen und wie das Ergebnis aussehen soll, überlegen Sie sich, wie der Lösungsweg untergliedert werden kann und welche Arbeitsschritte abgearbeitet werden müssen. Wenn Sie den Lösungsweg skizzieren und die Ablauflogik festgelegt haben, kann die eigentliche Programmierung beginnen. Jetzt ist es an der Zeit, RGH-PROFAN zu starten: Mit dem gewohnten Klick oder Doppelklick auf das Programm-Icon meldet sich der PROFAN-Editor, der eine komplette Programmierwerkstatt bereithält.

Wie jedes Windows-Programm stellt auch der PROFAN-Editor seine Funktionen und Werkzeuge über ein Menü und eine Symbolleiste zur Verfügung. Das Menüangebot paßt sich der Arbeitssituation an: Solange keine Programmdatei geöffnet ist, sind nur die Menüs DATEI, HELFER, FENSTER und HILFE zu sehen. Erst wenn eine Programmdatei zur Bearbeitung geladen wird, kommen die restlichen Menüs BEARBEITEN, SUCHEN, PROFAN und OPTIONEN zum Vorschein.

Im Menü DATEI ist – wie nicht anders zu erwarten – die Datei- und die Projektverwaltung zuhause. Was sind Projekte? Projekte sind so etwas wie Sammelordner, die keinen anderen Sinn und Zweck haben, als die an einem Programm beteiligten Dateien zusammenzufassen: Wenn zusammengehörige Programmdateien, Icons etc. unter einem Projektnamen gespeichert werden, sind sie schneller verfügbar, als wenn sie einzeln herausgesucht und geladen werden müssen. Ansonsten ist das DATEI-Menü für die üblichen Aufgaben wie *Neuanlegen*, *Öffnen*, *Speichern* und *Drucken* zuständig. Einziges Extra: Der Befehl *Programm aufrufen*, mit dem ein beliebiges Programm aus RGH-PROFAN heraus gestartet werden kann.

Das Menü BEARBEITEN stellt die üblichen Editor-Funktionen. Außer den gängigen Blockfunktionen wie *Ausschneiden*, *Kopieren*, *Einfügen* und *Löschen* finden Sie hier auch den Befehl *Rückgängig*, der Codeeingaben schrittweise zurücknimmt; die Option *Widerrufen* hebt wiederum den

jeweils letzten *Rückgängig*-Befehl auf. Mit dem Befehl *Setze Marker* können Sie einzelne Zeilen mit Markierungsmarken versehen, und der Menüpunkt *Zwischenablage* präsentiert den Inhalt des Clipboards. Ergänzt werden die Editor-Funktionen durch die verschiedenen Suchoptionen im Menü SUCHEN, die vor allem bei umfangreicheren Programmen die Bearbeitung enorm erleichtern.

Nach der Eingabe und der Bearbeitung des Programmcodes soll das Programm natürlich auch ausgeführt werden. Die entsprechenden Befehle werden über das Menü PROFAN abgerufen. Der Befehl

- *Ausführen* schaltet den Interpreter ein, der das im Quellcode eingegebene und gespeicherte PRF-Programm von der Festplatte liest;
- *Compilieren* startet den Compiler – dieser compiliert die PRF- zu einer PRC-Datei, die vom Runtime-Modul PROFRUN oder von einer PROFAN-Programmdatei ausgeführt werden kann und dabei deutlich schneller und im Schnitt nur halb so groß wie die PRF-Datei ist;
- *Starten* aktiviert das Runtime-Modul, das compilierte PROFAN-Programme ausführt;
- *EXE-Datei erstellen* ruft den Linker auf, der PRC-Dateien mit dem Runtime-Modul zu eigenständigen EXE-Programmdateien linkt.
- *Screensaver erstellen* veranlaßt den Linker, PRC- zu SCR-Dateien zu linkern, die als Bildschirmschoner in die Systemsteuerung eingebunden werden können.

Mit den Einstellungen im Menü OPTIONEN können Sie die Bildschirmanzeige regulieren sowie das Runtime-Modul für das aktuelle Projekt und die Schriftart für den Quellcode auswählen.

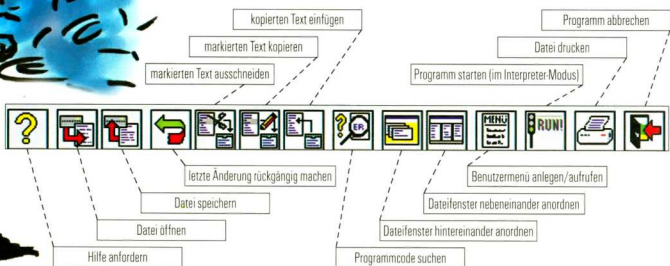
Hinweis: Achtung: Vergewissern Sie sich vor der Eingabe von Quelltext, daß eine der fünf Systemschriften eingestellt ist – andernfalls kann es vorkommen, daß der Cursor bei der Texteingabe nicht mitläuft, sondern hinterherhinkt!

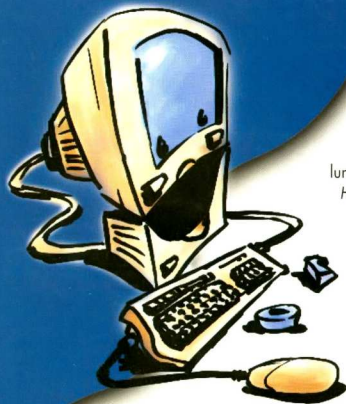
Nicht verzichten sollten Sie auch auf die Option, einen *Include-Pfad* zu definieren. Mit dem *Include*-Befehl

\$ I <Dateinamen>

können ganze Module während des Interpretierens und Compilierens dazugeladen werden. Der Vorteil: Häufig gebrauchte Module lassen sich beliebig oft eingeben, ohne daß sie jedesmal neu eingelesen oder eingefügt werden müssen.

Das Menü HELFER schließlich stellt verschiedene Designer bereit, mit denen Datenbankstrukturen und Datenbanktabellen bearbeitet, Menüs entworfen und Fenster und Dialoge für die interaktive Programmsteuerung unter Windows gestaltet werden





können. Der Fenster- und der Dialog-Designer realisieren so etwas wie ein visuelles Entwicklungskonzept: Sowie der jeweilige Helfer aktiviert und das Haupt- oder Dialogfenster in der gewünschten Größe aufgezogen ist, erscheint eine Objektpalette, aus der die einzelnen Kontrollelemente ausgewählt und in das Formular eingefügt werden können. Wie das vor sich geht und was dabei passiert, wird uns noch an anderer Stelle beschäftigen.

Blieben der Vollständigkeit halber noch die Menüs FENSTER und HILFE zu erwähnen, die freilich keiner weiteren Erläuterung bedürfen. Da RGH-PROFAN ein prozedurales Entwicklungskonzept verfolgt, sollen nun zunächst die Grundkomponenten der Programmiersprache vorgestellt werden.

Die Befehlssyntax: Do you speak Profan?

Sowie Sie mit dem Befehl DATEI\neue Datei bzw. DATEI\Datei öffnen eine Programmdatei anlegen oder laden, öffnet der Editor ein Dateifenster – in diesem Fenster werden Sie Ihre Programmangeweisungen codieren und damit den sogenannten Quell- oder Programcode erzeugen, der dann interpretiert und kompiliert und zu einem eigenständigen Windows-Programm gelinkt werden kann. Mit RGH-PROFAN formulieren Sie Ihre Programmangeweisungen in einer Sprache, die zwar nicht gerade den alltäglichen Sprachgepflogenheiten entspricht, mit der Umgangssprache – wenn auch Englisch – aber immerhin so weit verwandt ist, daß die einzelnen Instruktionen inhaltlich und sprachlich nachvollzogen werden können. Damit der eingebaute Interpreter die Programmangeweisungen in die Maschinensprache übersetzen kann, ist die Befehlssprache an einen vorgegebenen Code und an eine ganz bestimmte Syntax gebunden, die wie in jeder Sprache gelernt werden muß.

Variablen: Computerprogramme bestehen normalerweise aus einem Datenteil und einem Befehlsteil. Die Daten, die mit Hilfe von Befehlen und Funktionen verarbeitet werden sollen, werden üblicherweise in Variablen abgelegt. Variable sind Platzhalter für Werte, die während des Programmablaufs variieren können, und setzen sich aus einem Namen und einem Wert zusammen – eben dem Wert, durch den der Platzhalter im Programmablauf ersetzt wird. Der zugewiesene Wert kann eine Zahl, ebenso aber auch eine Zeichenkette sein. Das an den Variablenamen angehängte Sonderzeichen kennzeichnet den Datentyp: ---

Typ	Suffix	Wert
Integer	%	ganze Zahlen (von -32768 bis +32768)
Longinteger	&	ganze Zahlen (von -2 Milliarden bis +2 Milliarden)
Float	!	Fließkomma-Zahlen (mit einer Genauigkeit von mind. 15 Stellen)
String	\$	Text (bis zu 255 Zeichen)
Bereich	#	Zeiger für einen Datenbereich

In RGH-PROFAN können Variablen nur eingesetzt werden, wenn sie vor ihrer ersten Verwendung – am besten am Anfang des Programms – mit dem vorangestellten Befehl

DECLARE

deklariert worden sind. Mit Ausnahme der Variablen, die innerhalb einer Prozedur oder eines Unterprogramms deklariert werden, gelten Variablen global für das ganze Programm. Der Variablenname darf inklusive Suffix maximal 32 Zeichen lang sein und kein Leerzeichen enthalten. Der Wert wird der Variablen im Programmablauf mit dem Befehl

LET

zugewiesen. Bei Stringvariablen, die für einen Text oder eine Zeichenkette stehen, muß der Textstring in Anführungszeichen stehen; dabei können – durch Komma oder Semikolon getrennt – auch mehrere Stringausdrücke angegeben werden. Bereichsvariablen, die auf einen bestimmten Datenbereich verweisen, muß mit dem Befehl

DIM

ein Speicherbereich in Bytes zugewiesen werden, der mit dem Befehl

DISPOSE

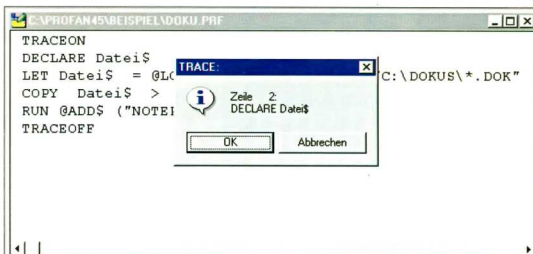
wieder freigegeben werden kann.

Systemvariablen: Neben den benutzerdefinierten Variablen, die als Platzhalter für Daten fungieren, gibt es noch Systemvariablen, die eingesetzt werden, wenn auf das System zugegriffen werden soll. Systemvariablen sind Platzhalter für Werte, die sich auf einen Zustand oder eine Aktion des Betriebssystems beziehen – beispielsweise auf die Ansteuerung eines Laufwerks oder auf die aktuelle Mausposition. Die entsprechenden Werte können bei der Programmierung über bestimmte Befehle und Funktionen ausgelesen und ausgewertet werden, zugewiesen aber werden sie vom System. Im Unterschied zu den benutzerdefinierten Variablen brauchen Systemvariablen, die durch ein Präfix gekennzeichnet sind, nicht deklariert zu werden.

Befehle: Was der Computer mit den eingegebenen Daten im Programmablauf anfangen soll, das sagen die Befehle, die RGH-PROFAN wie jede Programmiersprache für die Kommunikation mit dem Computer bereitstellt. Die Ein- und Ausgabe von Programmbeehlen kann im Text- wie auch im Grafikmodus erfolgen. Textmodus unter Windows? Damit Einsteiger und vor allem Umsteiger nicht auf die vertrauten Eingabe- und Ausgabebefehle aus der BASIC-Ära zu verzichten brauchen, simuliert der PROFAN-Editor einen DOS-Bildschirm im Textmodus, der wie der windowstypische Grafikmodus jederzeit verfügbar ist und nicht erst eigens eingeschaltet zu werden braucht. So ist es auch nicht weiter verwunderlich, daß viele von BASIC her bekannte Eingabe- und Ausgabebefehle auch im



Es hat geklappt: Die Quellcode-Datei wurde erfolgreich kompiliert und läuft nun deutlich schneller ab als im Interpreter-Modus.



Jede Menge Bugs? Kein Problem: Bei der Programmausführung im TRACE-Modus wird jede Programmzeile noch einmal in einem eigenen Kontrollfenster eingeblendet, so daß Syntaxfehler schnell aufgespürt und korrigiert werden können.

PROFAN- Befehlswortschatz auftauchen:

CLS	Bildschirm löschen
COLOR	Farbe für die Bildschirmausgabe einstellen
INPUT	Daten einlesen
LOCATE	den Cursor positionieren
PRINT	Daten ausgeben
WAITKEY	auf einen Tastendruck warten

Grundsätzlich funktionieren alle Befehle im Textmodus von RGH-PROFAN nicht anders als in BASIC, doch gibt es auch einige Unterschiede – so löscht CLS den Bildschirm nicht in der aktuellen Farbe, sondern in weiß, und auf den INPUT-Befehl darf keine Dateneingabe, sondern nur eine Variable folgen. Hinzu kommen schließlich noch einige weitere Befehle wie

FONT
NUMWIDTH/STRWIDTH
DECIMALS
TBOX

Mit diesen Befehlen lassen sich Zahlen und Texte auch im Textmodus formatieren.

Während Befehlseingaben im Textmodus sicherlich den Umstieg und Einstieg in die Windows-Programmierung erleichtern, werden erfahrenere Programmierer vermutlich den Text- und Grafikbefehlen im windowstypischen Grafikmodus den Vorzug geben. Fensterattribute werden mit den Befehlen

WINDOW
WINDOWSTYLE
WINDOWTITLE

festgelegt; für Textattribute stehen die Befehle

TEXTCOLOR
USEFONT
ORIENTATION

bereit. Mit
DRAWTEXT
wird der Text auf dem Bildschirm ausgegeben, mit

STARTPRINT/ENDPRINT
auf den Drucker bzw. wieder zurück auf den Bildschirm umgeleitet. Darüber hinaus stehen im Grafikmodus natürlich auch eine ganze Reihe von Grafikfunktionen zur Verfügung, die hier freilich nicht einzeln aufgezählt werden können.

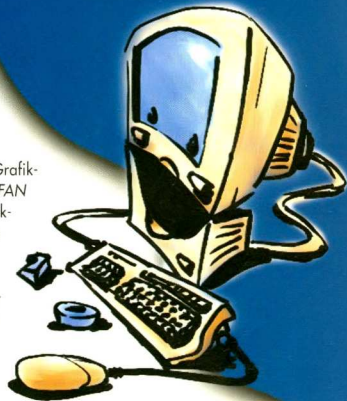
Erwähnt sei schließlich noch, daß mit dem Befehl

LOADEMP
Grafiken im Bitmap-Format geladen und angezeigt werden können. Auch auf die für Windows so typischen Icons brauchen Sie beim Programmieren mit RGH-PROFAN natürlich nicht zu verzichten. Um System-Icons von Windows einzusetzen, verwenden Sie den Befehl

DRAWSYICON
und fügen die gewünschte Nummer (0-4) und die Koordinatenwerte hinzu. Außer den Windows-Icons hat RGH-PROFAN auch noch eine Palette von eigenen Icons anzubieten, die mit dem Befehl
DRAWICON

und dem hinzugefügten Icon-
namen ausgegeben werden.

Über die vielfältigen Text- und Grafikbefehle hinaus stellt RGH-PROFAN noch eine Reihe von Datenbank-Befehlen sowie verschiedene Befehle für die Multimedia-Programmierung bereit, auf die hier aus Platzgründen nicht eingegangen werden kann. Informationen dazu entnehmen Sie der Einführung im PROFAN-Unterverzeichnis Doku.



Bedingungen: Bestünden Computerprogramme nur aus einer Sammlung von Befehlen, die der Reihe nach abgearbeitet werden müssen, wäre die Programmierung ein ziemlich langweiliges Unterfangen. Interessant wird die Sache erst, wenn die Programmanweisungen an Bedingungen geknüpft sind, die dem Computer Entscheidungen abverlangen. Die Kontrollstrukturen, die RGH-PROFAN einsetzt, um Verzweigungen einzubauen und Programmbefehle von bestimmten Bedingungen abhängig zu machen, werden ähnlich eingesetzt wie in anderen Programmiersprachen:

CASE/CASENOT
IF/IFNOT/ELSE/ELSEIF/ENDIF
WHILE/WHILENOT

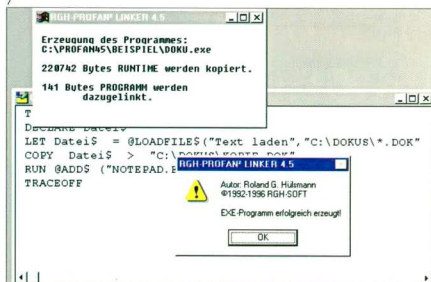
Geht einem Programmbefehl eine Bedingung mit CASE oder IF voraus, so wird der folgende Befehl nur ausgeführt, wenn der Wert im Bedingungs Ausdruck ungleich Null und die Bedingung somit erfüllt ist. Anders als bei CASE/CASENOT müssen IF-Bedingungen immer mit ENDF abgeschlossen werden. Während bei CASE/CASENOT die Bedingung und der Befehl in einer Zeile stehen, folgt der Befehl in einer IF-Bedingung immer in der nächsten Zeile. Mit ELSEIF können mehrere Bedingungen an einen Befehl geknüpft werden, der ELSE-Befehl hingegen wird ausgeführt, wenn die Bedingung nicht erfüllt wird. Der Bedingungs Ausdruck WHILE/WHILENOT schließlich verbindet die Bedingung mit einer Programmschleife: Die zwischen WHILE und ENDWHILE (oder auch WEND) stehende Programmanweisung wird solange ausgeführt, bis die Bedingung erfüllt ist.

Prozeduren: Um Platz zu sparen und unnötige Wiederholungen zu vermeiden, werden wiederkehrende Befehlssequenzen zu Prozeduren zusammengefaßt. Prozeduren sind also nichts anderes als Unterprogramme, die wie Variablen vor der ersten Verwendung definiert werden müssen. Die Definition wird eingeleitet mit dem Befehl

PROC
gefolgt vom Namen der Prozedur, und wird abgeschlossen mit
ENDPROC
Mit dem Befehl

PARAMETERS
können der Prozedur Parameter hinzugefügt werden. Parameter und Variablen,

Der letzte Akt: Die kompilierte Programmdatei wird zu einer ausführbaren EXE-Datei gelinkt. Jetzt läuft das neue Programm auch außerhalb von RGH-PROFAN auf jedem beliebigen Windows-Rechner!





die innerhalb der Prozedur deklariert werden, haben nur lokale Geltung außerhalb der Prozedur sind sie unbekannt. Globale Variablen dagegen, die außerhalb der Prozedur deklariert wurden, gelten auch innerhalb der Prozedur. Was die Verwendung angeht, unterscheidet sich eine Prozedur nicht von einem Befehl: Sobald sie aufgerufen wird, wird sie ausgeführt.

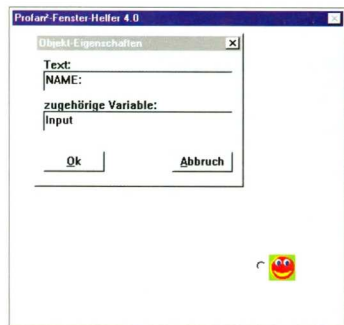
Funktionen: Für die Datenverarbeitung im Programmablauf werden neben den Befehlen auch Funktionen eingesetzt.

Funktionen sind letztlich auch nichts anderes als Befehle – mit dem einen Unterschied, daß Funktionen immer einen Wert zurückgeben, der durch die Parameter der Funktion bestimmt wird. Daraus ergibt sich, daß Funktionen überall im Programm eingesetzt werden können, wo variable oder konstante Werte stehen – auch dort, wo Sie vielleicht wie in anderen Programmiersprachen Operatoren erwarten würden; diese werden in RGH-PROFAN konsequent durch die entsprechenden Funktionen ersetzt.

Neben einer Reihe von Spezialfunktionen unterscheidet RGH-PROFAN zwei große Gruppen: Mathematische und Zeichenketten-Funktionen. Endet der Funktionsname mit dem Stringzeichen \$, wird meistens ein String als Ergebniswert zurückgegeben. Darüber hinaus besteht auch die Möglichkeit, eigene Funktionen einzusetzen, die mit dem Befehl

DEF definiert werden müssen. Auf den Befehl folgt der Name der Funktion, darauf in Klammern die Anzahl der Parameter und abschließend der Ausdruck, der die Funktion beschreibt. Gekennzeichnet werden Funktionen üblicherweise durch das Sonderzeichen @, das dem Funktionsnamen vorangestellt wird. Zwingend ist dieses Präfix jedoch nicht: Sollte Ihnen die Eingabe mit den Tasten [AltGr]+[Q] zu mühsam sein, können Sie auch darauf verzichten, was aber die Übersichtlichkeit des Programms im Quellcode unter Umständen beeinträchtigen kann.

Die Ausführung: Nachdem der Quellcode eingegeben ist, soll das Programm natürlich auch ausgeführt werden. Da der Computer freilich den Quelltext nicht lesen kann, muß der Interpreter aufgerufen werden, der den Programmcode von der Festplatte liest. Um das Programm dabei gleichzeitig zu testen, schalten Sie den TRACE-Modus ein, der mit dem einleitenden Befehl TRACEON aktiviert wird. Wenn das Programm in diesem Modus läuft, wird jede Programmzeile vor ihrer Ausführung noch einmal in einer eigenen Messagebox eingeblendet, so daß Syntaxfehler schnell aufgespürt und korrigiert werden können. Sind alle Fehler getilgt, schalten Sie den TRACE-Modus mit abschließendem TRACEOFF wieder aus, um das Programm darauf zu compilieren und eine ausführbare .EXE-Datei zu erstellen, die wie jede andere Programmdatei über den Dateimanager oder den Explorer gestartet werden kann.



Programmieren im Designer-Modus:
• Programm- und Dialogfenster werden auf einer gerasterten Zeichenfläche gestaltet. Die einzelnen Kontrollelemente brauchen nur ausgewählt, beschriftet und in der gewünschten Größe im Formular platziert zu werden. Den dazugehörigen Quellcode generiert das Programm!

PUNKT, PUNKT, KOMMA, STRICH ...

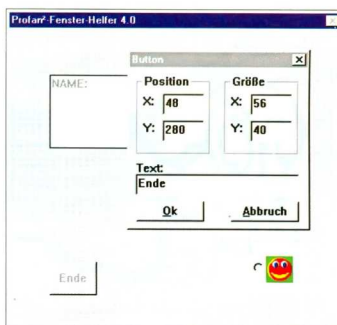


Programmiersprachen verlangen normalerweise, daß die Befehlssyntax ganz exakt eingehalten wird. Ein Leerzeichen zuviel oder zuwenig an der falschen Stelle kann dazu führen, daß die Befehlsausführung schlichtweg verweigert wird. Obwohl PROFAN in dieser Hinsicht etwas großzügiger ist, sollten Sie die folgenden Regeln beachten:

- Zwischen Befehl und Parameter muß immer exakt ein Leerzeichen stehen
- Leerzeichen vor Programmzeilen haben nur die Funktion, den Quelltext zu gliedern und die Übersicht zu erleichtern. Dasselbe gilt für Absätze mit [RETURN]
- Tabulatoren sind tabu! Verwenden Sie also Leerzeichen, um Befehlszeilen abzusetzen und einzurücken
- Bei Funktionen werden Parameter mal durch ein Leerzeichen, mal durch eines der anderen sechs Trennzeichen (, ; - = :) getrennt. Im Zweifelsfall schauen Sie in der Online-Referenz nach. Kontextbezogene Hilfestellung erhalten Sie prompt, wenn Sie das entsprechende Element im Quellcode markieren und die Tastenkombination [STRG] + [F1] drücken
- Sonderzeichen, die den Variablentyp festlegen, werden ohne Leerzeichen unmittelbar an den Variablenamen angehängt
- Der berühmte Klammeraffe @ vor Funktionen kann weggelassen werden, was freilich zu Verwechslungen führen und die Ausführung im Interpreter-Modus verlangsamen kann
- Groß- und Kleinschreibung bei Befehlen, Funktionen und Variablen ist irrelevant
- Befehlszeilen sind prinzipiell einzilig. Wenn Befehlszeilen um der Übersichtlichkeit willen über mehrere Bildschirmzeilen geschrieben werden sollen, muß am Zeilenende ein Backslash [\] angefügt werden.

Fenster und Dialoge: Oberflächenkosmetik

Befehle und Prozeduren, Funktionen und Variablen geben gewissermaßen das Gerüst ab, das den Inhalt, den Aufbau und den Ablauf eines PROFAN-Programms bestimmt. Noch freilich deutet nichts darauf hin, daß aus diesem Gerüst ein Windows-Programm werden soll. Für die bloße Abarbeitung von Befehlen braucht man kein Windows. Was ein Programm erst zu einem typischen Windows-Programm macht, das sind die vielfältigen Interaktionen zwischen Anwender und Programm, die nicht durch Befehlseingabe, sondern intuitiv über die sichtbaren Kontrollelemente der grafischen Oberfläche gesteuert werden. Für die Programmierung bedeutet dies, daß die Gestaltung der Fenster und der Kontrollelemente von Anfang an in die Programmentwicklung einbezogen werden müssen. Klar, daß das Programm in einem eigenen Programmfenster ablaufen soll. Wie aber soll dieses Fenster aussehen? Welche Steuerungselemente werden benötigt?



Fortsetzung S.43

Sollen die Programmfunktionen über ein Menü und/oder über Schaltflächen und Icons abgerufen werden? Welche Benutzereingaben müssen eingeplant werden, und über welche Kontrollelemente sollen sie erfolgen?

Wenn Sie sich über die Grundstruktur im klaren sind, geht's an die Umsetzung und die Ausgestaltung. Wie lassen sich nützliche Befehlsfolgen in ein echtes Windows-Programm mit Programmfenster, Menüs und Dialogen verwandeln? Prinzipiell stehen zwei Möglichkeiten zur Disposition: Sie übernehmen die fertigen Dialogelemente, die *RGH-PROFAN* in Form von Funktionen bereitstellt, oder Sie entwickeln Ihre Dialoge selbst, was mit den entsprechenden Helfern zwar kein Hexenwerk ist, aber doch ein bißchen mehr Arbeitsaufwand und Programmierkenntnis erfordert.

Die Funktionen, mit denen fertige Objekte in den Quellcode des aktuellen Projekts eingebaut werden können, beziehen sich auf die folgenden Kontrollelemente:

- `MessageBox`
- `InputBox/EditBox`
- `ListBox`
- `LOAD-/SAVE-Dialoge`

Die Funktion `@MessageBox` gibt einfache Fragen und Meldungen auf dem Bildschirm aus und hat drei Parameter – den Text, die Überschrift der Titelzeile und die Kennzahl, die sich zusammensetzt aus den addierten Werten, die für die Spezifizierung von Buttons, Icon, Default und Fensterart eingesetzt werden. Die Funktionen `@Input` und `@EditBox` erfassen Benutzereingaben und werden durch drei Parameter definiert, nämlich die Eingabeaufforderung, den Dialogtitel und den Vorgabewert, der vom Anwender übernommen oder überschrieben werden kann. Der Funktion `@ListBox` ist ein eigenes String-Feld zugeordnet, das bis zu 10 000 Einträge zur Auswahl stellen kann. Die beiden Parameter dieser Funktion legen die Überschrift und das Schriftformat fest. Die Dialoge zum Laden und Speichern von Dateien schließlich werden mit den Funktionen `@LoadFile$` und `@SaveFile$` eingesetzt. Die dazugehörigen Parameter bezeichnen die Überschrift und die Datei, die geladen oder gespeichert werden soll.

In kleinen Windows-Applikationen reichen die fertigen *PROFAN*-Dialoge sicherlich aus, um einfache Bedienungs- und Steuerungselemente einzubauen. Wer die Interaktionsmöglichkeiten unter Windows freilich ausschöpfen möchte, wird nicht darauf verzichten können, eigene Dialoge zu entwickeln. Neben den Datenbankhelfern und dem Menü-Designer stellt das HELFER-Menü auch einen Fenster- und

einen Dialog-Designer bereit, mit denen die entsprechenden Fenster sehr viel schneller erstellt sind, als wenn sie mit den Funktionen `@CreateWindow` und `@CreateDialog` von Anfang bis Ende programmiert werden müssen.

Vom Aufbau und der Funktionsweise her sind der Fenster- und der Dialog-Designer quasi identisch – was sie unterscheidet, ist nur das Objekt: Der Fenster-Designer liefert das Gerüst für Programmfenster, der Dialog-Designer arrangiert Dialoge. Wenn Sie die Option *Fenster* oder *Dialog* gestalten im HELFER-Menü wählen, werden Sie aufgefordert, mit der Maus ein Feld für das Formularfenster aufzuziehen. Sowie Sie die Maustaste loslassen, erscheint ein gerastertes Formular nebst einer Objektpalette, aus der die Kontrollelemente, die die Interaktion zwischen Programm und Anwender steuern, ausgewählt und eingesetzt werden können.

Sowie ein Steuerungsobjekt in der gewünschten Größe und Position im Formular arrangiert ist, wird automatisch ein kleiner Dialog eingeblendet, in dem die *Objekteigenschaften* definiert werden müssen – dazu gehören der *Text*, d.h. der Beschriftung des Kontrollelements, und die zugeordnete *Variable*. Soll ein bereits eingefügtes Fensterobjekt wieder gelöscht oder abgeändert werden, wählen Sie die entsprechende Funktion des Objektménüs, das jedes Objekt nach einem Klick mit der rechten Maustaste bereithält. Was beinhaltet die Variable? Die Objektvariable ist nichts anderes als ein Platzhalter, der darauf hinweist, daß an dieser Stelle eine Benutzereingabe erfolgt. Welche Aktionen konkret zu erwarten sind und wie das Programm darauf reagieren soll, das legt die Prozedur fest, die im Editor an der entsprechenden Stelle des Quellcodes einzufügen ist – der Dialoghelfer generiert nur den Quellcode für das Gerüst, den Inhalt – die Prozeduren – müssen Sie natürlich selbst eingeben.

Ist das Dialogelement definiert und im Formular eingefügt, sollte es gespeichert werden: Mit der Option *Speichern* in der Objektpalette wird das erzeugte Objekt in einer eigenen Datei abgelegt, die so freilich noch nicht in den Programmcode eingebaut werden kann, sondern lediglich dazu dient, daß unvollendete Objekte nach einer Unterbrechung wieder geladen und fertiggestellt werden können. Um den Quellcode, der bei der Erstellung und Definition eines Kontrollelements automatisch erzeugt wird, in den Programmcode des aktuellen Projekts einzubinden und zu vervollständigen, muß das Objekt mit der Option *Quelltext* als PRF-Datei gespeichert werden.

Wenn schließlich alle Dialogkomponenten fixiert und als Werkdatei wie im Quelltext gespeichert sind, rufen Sie den Interpreter auf und schauen sich an, wie sich Ihr Programm im neuen Windows-Look auf dem Bildschirm präsentiert!



Alles paletti: Die Objektpalette der Fenster- und Dialog-Helfer stellt die typischen Kontrollelemente für die visuelle Programmsteuerung über die grafische Oberfläche bereit. Wichtig: Die Speicherungsoption *Quelltext*, die den automatisch erzeugten Quelltext in den Editor übernimmt.

Button
Text
Icon
Eingabe
RadioButton
CheckBox
GroupBox
ListBox
sort. ListBox
AuswahlBox
EditBox
HScroll
VScroll
Laden
Speichern
Quelltext
Editor
Ende



Praxis Der Sprung ins kalte Wasser

Wie das Programmieren mit *RGH-PROFAN* theoretisch vor sich geht, wissen Sie jetzt. Doch Theorie und Praxis sind nicht dasselbe. Wie es wirklich funktioniert, das zeigt sich erst, wenn der

berühmte Sprung ins kalte Wasser gewagt wird. Der folgende Praxisteil gibt Ihnen Gelegenheit, ein komplettes kleines Windows-Programm selbst zu entwickeln – Schritt für Schritt und Zeile für Zeile, so daß auch Einsteiger die einzelnen Etappen nachvollziehen können. Natürlich muß sich dieser erste Programmversuch auf die Grundlagen beschränken – die Entwicklung einer einfachen Anwendung mit einem Programmfenster, einem Menü und einer Dialogbox. In die tieferen Geheimnisse der Programmierung mit *RGH-PROFAN* werden Sie mit zunehmender Erfahrung von selbst vordringen!

Um die vielfältigen Möglichkeiten der Programmiersprache an einem kleinen Beispiel zu demonstrieren, werden wir eine kleine Anwendung programmieren, die dazu dient, einen zuvor eingegebenen Namen in kunstvoller typographischer Stilisierung im Hauptfenster anzuzeigen und auszudrucken. Dabei sind folgende Arbeitsetappen zurückzulegen:

- 1 Gestaltung des Programmfensters
- 2 Aufbau des Menüs
- 3 Definition des Dialogfensters
- 4 Zusammenführung der Quelltexte
- 5 Einbau der Events in der WHILE-Schleife
- 6 Programmierung der Ereignisprozeduren
- 7 Erstellung der fertigen EXE-Datei



In den ersten drei Etappen, die der Gestaltung der Programmoberfläche gewidmet sind, nutzen wir die komfortable Hilfe der Fenster-, Dialog- und Menü-Designer, die *RGH-PROFAN* im Menü *HELPER* bereithält. Dabei sollten Sie sich freilich von Anfang an darüber im klaren sein, daß Sie mit den Helfern lediglich den Code für das äußere Programmgerüst generieren – die Befehle, die Menüs oder Schaltflächen mit Benutzereingriffen – sogenannten *Events* – verknüpfen, müssen manuell eingegeben werden.

1 Die Gestaltung des Programmfensters

Beginnen wir gleich mit den Rahmenbedingungen, sprich: dem Programmfenster. Von diesem Hauptfenster, das beim Programmstart geöffnet wird, gehen alle weiteren Aktionen aus, und über dieses Fenster wird das Programm auch wieder ordnungsgemäß beendet. Das Design des Programmfensters wird mit den Befehlen *WINDOW*, *WINDOWSTYLE* und *WINDOWTITLE* und den dazugehörigen

Parametern festgelegt. Die Codeerzeugung können Sie freilich weitgehend dem Fenster-Designer überlassen. Sie selbst brauchen nur mit der Maus ein Feld in der gewünschten Fenstergröße aufzuziehen. Sowie Sie die Maustaste loslassen, wird neben dem Fenster die Objektpalette mit den windoweigenen Steuerungselementen eingeblendet. Da die Namensgebung im Hauptfenster unserer Beispielanwendung jedoch ausschließlich über ein Menü erfolgen soll, brauchen wir hier keine zusätzlichen Kontrollelemente einzufügen, sondern speichern das Programmfenster samt dem vom Fenster-Designer generierten Quellcode erst einmal ab.

Der bisher erzeugte Quellcode ist allerdings noch nicht vollständig: Noch fehlen die Befehle und die Funktionen, die in Form einer Prozedur oder eines Unterprogramms in den Programmcode eingebaut werden müssen. Damit der noch unvollständige Quellcode für das Hauptfenster funktionstüchtig in den Programmcode eingebaut werden kann, muß er im profaneigenen Codeformat *.prf gespeichert werden. Dies geschieht mit der Speicheroption *Quelltext* und nicht mit der Option *Speichern*, die nur dazu bestimmt ist, ein noch nicht fertiges Fenster für eine spätere Weiterbearbeitung mit dem Designer zu sichern.

Der Klick auf die Schaltfläche *Quelltext* in der Objektpalette bewirkt, daß

- die eingegebenen Variablen deklariert,
- die Prozedurbefehle *PROC* und *ENDPROC*, die das Fenster als eigenes Unterprogramm ausweisen, hinzugefügt werden
- die unerläßliche *WHILE*-Schleife mit dem *WAITKEY*- bzw. *WAITINPUT*-Befehl generiert wird, die das Fenster auf Anwendereingaben vorbereiten.

Um also das – zugegeben noch spartanische – Hauptfenster samt Quellcode in den Programmcode zu übernehmen, speichern Sie es mit der Option *Quelltext*. (Abb. 1 u. 2)

2 Der Aufbau des Menüs

Können Sie sich ein Windows-Programm ohne Menü vorstellen? Wohl kaum. Gerade weil wir im Programmfenster auf jegliche Schaltflächen verzichtet haben, ist die Steuerung über ein Menü erst recht unverzichtbar. Beim Entwurf des Menüs ist uns der Menü-Designer behilflich. Das Menü in unserem Programmfenster soll die Namensgebung und -ausgabe steuern. Also entwickeln wir ein PopUp-Menü *NAME* mit den Befehlen *Eingeben...*, *Bildschirm*, *Drucken* und *Ende* und dazu ein kleines *HILFE*-Menü, das freilich darauf beschränkt bleibt, ein *Info* in Form einer Messagebox einzublenden.

Die Definition der Menübefehle ist mit dem Menü-Designer wahrlich keine Hexerei: Geben Sie zunächst im Feld *Menütext* den Namen ein und stellen Sie dabei dem Buchstaben, der als *Hotkey* fungieren soll, das Sonderzeichen & voran. Damit beispielsweise der Menübefehl *Eingeben...* in unserer Anwendung mit der Taste [G]

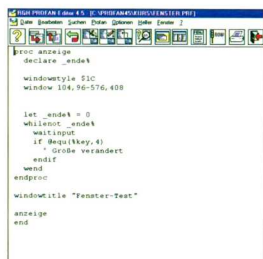


Abb. 1



Abb. 2

Der vom Fenster-Designer erzeugte Quellcode des Programmfensters sollte über die Option *Quelltext* abgespeichert werden, um die eingegebenen Variablen automatisch zu deklarieren und eine *WHILE*-Schleife zu eröffnen.

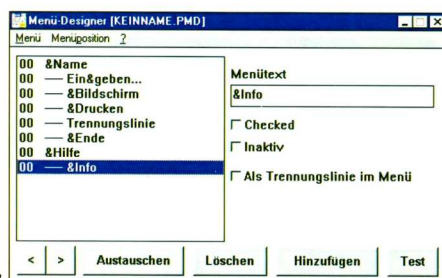


Abb. 3

aufgerufen werden kann, muß der Eintrag im Menütext „Ein&geben...“ lauten. Mit einem Klick auf *Hinzufügen* wird der Menüeintrag in die Liste der linken Fensterhälfte übernommen. Um einen Menüpunkt einem vorhergehenden Menübefehl unterzuordnen und einem eigenen PopUp-Menü zuzuweisen, wird er mit der Richtungstaste (>) um eine Position nach rechts verschoben – die hierarchische Ordnung der Menübefehle wird also schon bei der Programmierung optisch umgesetzt. Soll das Menü zusätzlich gegliedert werden, können Sie mit der Option *Als Trennlinie* im Menü und dem Befehl *Hinzufügen* eine Trennlinie einfügen – sinnvoll ist dies vor allem, wenn Menübefehle ganz unterschiedliche Aktionen ausführen. In unserem Beispiel soll der Befehl *Ende* – wie in den meisten Windowsprogrammen – durch eine Trennlinie von den übrigen Befehlen abgesetzt werden. Wie die fertigen Eingaben für unser Beispielménü aussehen sollten, sehen Sie in Abb. 3. Wer auf Nummer Sicher gehen möchte, sollte das fertige Menü mit einem Klick auf den Button *Test* in einem Testfenster überprüfen. Wie beim Programmfenster muß auch der vom Menü-Designer erzeugte Quelltext für das Menü mit dem Befehl *MENÜ/Quelltext* als *.prf-Datei gespeichert werden. (Abb. 3 u. 4)

3 Die Definition des Dialogfensters

Im Programmfenster unserer Anwendung wird der Benutzer aufgefordert, über den Menübefehl *NAME/Eingeben...* einen beliebigen Namen einzugeben. Die entsprechende Eingabe soll in einem eigenen kleinen Dialogfenster erfolgen, das Sie nun mit Hilfe des Dialog-Designers erstellen. Nachdem Sie das Fensterfeld aufgezogen haben, werfen Sie einen Blick auf die Objektpalette, um zu sehen, welche Kontrollelemente zur Verfügung stehen. Objekte werden in der Objektpalette ausgewählt und mit der Maus an der gewünschten Stelle im Formular platziert. Damit eingefügte Steuerungselemente in den Programmcode übernommen werden können, muß ihnen – mit Ausnahme des Icons – ein Text und ggf. eine Variable (ohne Suffix) zugewiesen werden. Eine Variable ist freilich nur erforderlich, wenn das Dialogobjekt auf eine Benutzereingabe reagieren soll.

Welche Kontrollelemente kommen für unseren Eingabe-Dialog in Frage? Zunächst ein Textfeld, das den Anwender auffordert, einen Namen einzutippen. Da dieses Textfeld nur angezeigt wird, aber keine Aktion vom Anwender auslöst, brauchen wir ihm keine Variable, sondern nur einen Text zuzuordnen – in unserem Beispiel: „Bitte Namen eingeben:“. Weiterhin benötigen wir noch ein Eingabefeld für den Namen und die Buttons „OK“ und „Abbrechen“, die das Dialogfenster schließen sollen. Während das Eingabefeld keinen Text benötigt (den gibt der Anwender selbst ein!), müssen die Buttons – warum nicht wie beim Menü auch mit Hotkey? – beschriftet werden. Da das Eingabefeld und die Buttons auf eine Aktion des Benutzers warten, weisen Sie ihnen auch gleich Variablen mit passendem Variablennamen zu – in unserem Fall z.B. *Name*, *ok* und *Ende*. Sind alle Objekte an ihrem Platz und deklariert, speichern Sie das Dialogfenster über die Schaltfläche *Quelltext* ab. (Abb. 5)

Hinweis: Alle Steuerungselemente mit Variablen müssen im Programmcode manuell mit einem Event verknüpft werden. Die einzige Ausnahme ist die Variable *Ende*, bei der *RGH-PROFAN* den Code in der *WHILE*-Schleife, der das Fenster schließt, automatisch erzeugt.

4 Die Zusammenführung der Quelltexte:

Noch liegen die Quelltexte von Hauptfenster, Menü und Dialogbox in separaten Dateien vor. Bevor wir die noch fehlenden Befehlszeilen im Programmcode ergänzen,

empfiehlt es sich, zunächst alle Quelltexte in einer Datei zusammenzuführen. Dazu legen Sie mit dem entsprechenden DATEI-Befehl eine neue Datei an und wählen dann den Befehl *BEARBEITEN/Einfügen aus Datei...*, um zunächst den Quellcode des Dialogfensters zu übernehmen. Warum beginnen wir ausgerechnet mit der Dialogbox? Ganz einfach weil der Quelltext dafür von *RGH-PROFAN* komplett als Prozedur (*PROC Dialog*) geschrieben wurde; Prozeduren aber sollten am Anfang des Programmcodes stehen, da sie vor ihrer ersten Verwendung definiert werden müssen. Ist der Quellcode des Dialogfensters komplett eingefügt, folgt der Quellcode des Hauptfensters, und zu guter Letzt bauen wir vor der *WHILE*-Schleife des Hauptfensters noch den Quellcode des Menüs ein. Nachdem die Quelltexte zusammengefügt sind, braucht der neue Quellcode nur noch unter neuem Namen gespeichert zu werden.

Nach diesen einfachen Kopiervorgängen läuft das Programm freilich noch nicht reibungslos ab. Dazu müssen noch diejenigen Programmzeilen entfernt bzw. geändert werden, mit denen die einzelnen Quelltexte bislang in eigenen (Test)Fenstern angezeigt wurden. Da sind zunächst die Programmzeilen

```
WINDOWTITLE „Dialog-Test“
CLS
dialog
WAITINPUT
END
```

am Ende der Dialog-Prozedur. Da sie ausschließlich für das Testfenster erzeugt wurden, können wir sie gestrost entfernen. Dasselbe Schicksal ereilt die Codezeilen

```
WINDOWSTYLE 31
WINDOW 10,10-600,400
WINDOWTITLE „Jetzt können Sie das Menü ausprobieren!“,
```

die sich nur auf das Testfenster für die Menüanzeige beziehen. Da wir die Umgebung für das Menü bereits mit dem Hauptfenster festgelegt haben und kein zweites Fenster erstellen wollen, sind diese Codezeilen überflüssig. Auch die Befehle *WAITINPUT* und *END* vor Beginn der *WHILE*-Schleife sind Relikte aus dem Testmodus und müssen gelöscht werden. Außerdem muß das Hauptfenster nun nicht mehr als Prozedur deklariert werden. Daher können wir die Zeile

PROC Anzeige
bzw. die Aufhebung mit *ENDPROC* und den Prozedurauftrag *Anzeige* nach der Ereignisschleife löschen. Und last but not least sollte die Titelzeile des Hauptfensters *WINDOWTITLE „Fenster-Test“* sinnvollerweise vom Ende des Programmcodes zu den anderen Fensterbefehlen *WINDOW* und *WINDOWSTYLE* gestellt werden – andernfalls erscheint als Titelzeile nur der Hinweis auf den *PROFAN*-Interpreter.

Wenn Sie die abgespeicherte Datei nach diesen Änderungen mit dem Interpreter ausführen, öffnet sich das Programmfenster mit dem

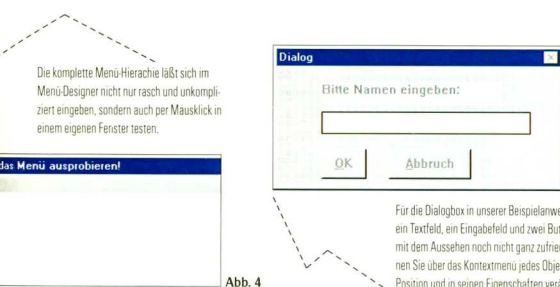


Abb. 5

Für die Dialogbox in unserer Beispielanwendung reichen ein Textfeld, ein Eingabefeld und zwei Buttons. Falls Sie mit dem Aussehen noch nicht ganz zufrieden sind, können Sie über das Kontextmenü jedes Objekt in seiner Position und in seinen Eigenschaften verändern.





Menü – und möglicherweise wird sich bei Ihnen schon der erste „Aha“-Effekt einstellen.

5 Der Einbau der Events in der WHILE-Schleife

Nachdem das äußere Grundgerüst mit dem Hauptfenster, dem Menü und der Dialogbox steht, kommt freilich noch der aufwendigere und auch schwierigere Teil der Programmierung – sozusagen die „Handarbeit“ – an die Reihe. Nun müssen Sie

noch einen in sich geschlossenen Programmcode erzeugen, indem Sie

- in den WHILE-Schleifen die Events für Menüs und Objekte eintragen,
- Events, die sich aus mehreren Befehlen zusammensetzen, in Prozeduren zusammenfassen,
- den Quelltext Zeile für Zeile durchgehen, um die letzten Korrekturen (Variablen deklarationen, Änderung der Fensterstile etc.) vorzunehmen.

Wenn Sie die bisherigen Quelltexte für Hauptfenster, Dialog und Menü testweise im PROFAN-Editor aufrufen und mit dem Interpreter ausführen, werden Sie einerseits feststellen, daß das „Layout“ stimmt, auf der anderen Seite aber auch bemerken, daß die Buttons bzw. Menübefehle beim Anklicken noch nicht die gewünschte Aktion auslösen. Kein Wunder – Sie haben ihnen die entsprechenden Events im Quellcode bislang auch noch vorenthalten! Dies muß nun nachgeholt werden.

Wir beginnen mit den WHILE- oder Ereignis-Schleifen. Wie bereits erwähnt, warten Windows-Programme in der Regel auf Benutzereingaben wie z.B. einen Tastendruck oder einen Mausklick, bevor eine Aktion erfolgt. Die entsprechenden Eingaben und die damit verknüpften Ereignisse werden in der WHILE-Schleife definiert. WHILE-Schleifen werden so lange durchlaufen, wie der Wert der Variablen, die die Schleife beendet (meist `_Ende%` oder `_Ende%`), Null ist. Daher wird der Variablen `_Ende%` vor Beginn der Schleife mit dem Befehl `LET` von RGH-PROFAN automatisch der Wert 0 zugewiesen:

```
let _Ende%=0
```

Da die Ereignisschleife, die mit dem Befehl `WHILE` oder `WHILENOT` (mit angefügter Variable) als Bedingungs Ausdruck eingeleitet wird, nur durch eine Eingabe des Anwenders beendet werden kann, fügt RGH-PROFAN automatisch den Befehl `WAITINPUT`

ein. Wie der erwartete Input aussehen soll, definieren Sie in den folgenden `IF`- und `ELSEIF`-Bedingungen selbst.

Betrachten wir noch einmal die WHILE-Schleife im Quelltext unseres Beispiel-Programms, und zwar zunächst die Schleife des Dialogfensters: Die erste `IF`-Bedingung, die die WHILE-Schleife unterbricht, hat RGH-PROFAN bereits eingetragen:

```
if @equ(%key,2)
  let _ende% = 1
```

Was bedeutet dies? Mit der Funktion `@equ`, die zwei Integer-Werte vergleicht, wird über die Systemvariable `%key` und den zugewiesenen Wert 2 bestimmt, daß sich das Fenster durch Anklicken des Kreuzes am rechten oberen Fensterrand schließt – bei Windows-Fenstern längst eine Marginalie.

Das Ende der Schleife wird ganz einfach dadurch erreicht, daß der Variablen `_Ende%` der Wert 1 zugewiesen ist, bei dem die Schleife automatisch abgebrochen wird.

Für die drei Dialogobjekte, denen wir im Dialog-Helfer eine Variable zugewiesen haben, sind bereits drei `ELSEIF`-Bedingungen reserviert – mit der Funktion `@getfocus`, die das Handle (die von Windows systemintern vergebene Verwaltungsnummer) überprüft. Die erste lautet

```
elseif @getfocus(Name%)
```

Hier verzweigt das Programm, wenn der Anwender im Eingabefeld ein Ereignis auslöst. Da der Anwender hier aber nur einen Namen eingeben soll, das Dialogfenster aber in jedem Fall geöffnet bleibt, können wir diese Bedingung ersatzlos streichen. Wichtiger ist die folgende Bedingung, die festlegt, was passiert, wenn der OK-Button (Variable `OK%`) angeklickt wird: Wenn dieses Ereignis eintritt, soll die Eingabe ausgelassen und der Dialog beendet werden. Dafür würde die Funktion `@gettext` mit der Variablen `Name%` als Parameter genügen. Da der eingegebene Text aber später auch noch in einer anderen Prozedur verwendet werden soll – bei der Anzeige im Programm mit dem Menübefehl `Bildschirm...`, ist es sinnvoll, eine neue Variable anzulegen. Wir nennen sie `Zeile%` und weisen ihr in der Codezeile

```
let Zeile% = @gettext(%Name%)
```

die Funktion `@gettext` zu. Somit brauchen wir später, wenn wir dem Menübefehl `Bildschirm` ein Ereignis zuweisen, nur noch den Namen der Variablen anzugeben. Anschließend setzen wir auch hier `_Ende%` auf 1, damit die Schleife verlassen wird. Natürlich dürfen wir später nicht vergessen, die Variable `Zeile%` im Hauptfenster zu deklarieren!

Die dritte Bedingung für den Abbruch des Dialogs hat RGH-PROFAN bereits formuliert: Es ist das uns schon bekannte Setzen der `_Ende%`-Variable auf den Wert 1. Darauf folgen die Befehle `ENDIF` und `WEND`, die die Bedingungen und die Definition der Schleife abschließen. Statt des `BASIC`-Befehls `WEND` läßt sich auch der Befehl `ENDWHILE` einsetzen, der in Analogie zu den Befehlspaaren `PROC`/`ENDPROC` und `IF`/`ENDIF` sicherlich logischer erscheint.

Wundern Sie sich über die angehängte Funktion `@destroywindow (%dlg)%`? Keine Sorge, es geht alles mit rechten Dingen zu: `@destroywindow` entfernt – je nach Integer-Variable – ein Fenster oder Objekt vom Bildschirm und aus dem Speicher, ähnlich dem Befehl `END`, der das Hauptfenster und damit das gesamte Programm beendet. Ohne diese Funktion würde die Dialogbox nach Beenden der WHILE-Schleife nicht geschlossen werden. Beachten Sie bitte auch, daß das gesamte Ereignis „Dialogfenster“ von RGH-PROFAN automatisch als Prozedur deklariert wurde, eingerahmt mit den Befehlen `PROC` (Dialog) und `ENDPROC`. Die vom Benutzer eingegebenen Variablen, die nur in dieser Prozedur erforderlich sind, werden als lokale Variablen direkt nach dem `PROC`-Befehl deklariert.

Nun zur Schleife des Programmfensters: Das Programmfenster soll sich nur über das Menü steuern lassen. Daher müssen wir in der Schleife die einzelnen Menübefehle als Bedingungen einfügen. Die ersten vier Zeilen der Schleife – vom Setzen der `_Ende%`-Variable bis zur Funktion `@equ` – sind bereits bekannt und können unverändert übernommen werden – sie tauchen, nur geringfügig verändert, bei fast jeder von den Helfern generierten Schleife wieder auf. Zu erklären ist hier allerdings noch der Wert 4 hinter der Systemvariablen `%key`: Er bestimmt, daß das Programmfenster in der Größe verändert werden kann.

Die einzelnen Menübefehle, die die Schleife unterbrechen sollen, werden wieder mit dem `ELSEIF`-Befehl angefügt, den Sie nun selbst eingeben müssen. Die Funktion,

```
Abb. 6 C:\PROFAN45\KURSNAME.PRF
waitinput
if @equ(%key,4)
elseif @menuitem(101)
  Dialog
elseif @menuitem(102)
  Ausgabe
elseif @menuitem(103)
  Drucken
elseif @menuitem(104)
  let _ende% = 1
elseif @menuitem(201)
  @messagebox ("(c)Trendverlag 1998", "INFO", 48)
endif
wend
```



Einfacher geht's nicht: Vordefinierten Dialogen wie z.B. der Funktion `@messagebox` brauchen Sie im Prinzip nur noch die Nachricht, die Titelzeile und den summarischen Wert der einzelnen Elemente einzutragen – alles in einer einzigen Befehlszeile!

die einen Menüpunkt mit einem Ereignis verknüpft, lautet

@MenuItem(N)

(N) steht dabei für die Menükennzahl, die dem Menüpunkt im Quellcode des Menü-Designers zugewiesen wurde. Wenn Sie also beispielsweise dem Menüpunkt „Eingeben“ ein Event zuordnen wollen, schauen Sie einfach im Menü-Quellcode nach und ergänzen

@MenuItem(101)

Was soll nun bei Anklicken des Menübefehls „Eingeben“ geschehen? Klar, der Eingabedialog soll eingeblendet werden. Da wir das gesamte Dialogfenster bereits als Prozedur festgelegt haben, brauchen wir nur noch den Namen der Prozedur anzufügen. Wir schreiben also

@MenuItem(101)

Dialog

Ähnlich gehen wir bei den Menübefehlen *Bildschirm* und *Drucken* vor: Da beide Befehle Aktionen aufrufen können, die mehr als einen Befehl oder eine Funktion zulassen, fügen wir einfach den (gleichlautenden) Namen für eine Prozedur hinzu, die wir freilich später noch definieren müssen. Beim Menüpunkt „Ende“, der das Programm sofort beendet, können Sie wahlweise den Befehl END einfügen oder die Variable *_Ende%* wieder auf 1 setzen.

Nun fehlt nur noch die Einblendung einer kleinen Nachrichtenbox über den Menübefehl „Info“ im Menü HILFE: Hier greifen wir auf einen vordefinierten Dialog zurück, der uns eine Menge Programmierarbeit abnimmt, nämlich die Funktion

@MessageBox(S1,S2,N)

Im Klammersausdruck geben Sie die einzublendende Nachricht, anschließend den Fenstertitel – beide in Anführungszeichen – und abschließend den Integer-Wert der Box ein, der sich aus der Addition der Werte für Buttons, Icon, Default und Fensterart zusammensetzt; die entsprechenden Werte finden Sie in der Referenz. In unserem Beispiel wählen wir den OK-Button (Wert 0), das Icon mit dem Ausrufungszeichen (Wert 48), den ersten – das einzigen – Button als Standard-Vorgabe (Wert 0) und die Anzeige in einem „normalen“ Fenster (Wert 0). Summengezählt ergibt dies eine Summe von 48, die wir im Klammersausdruck an letzter Stelle eintragen:

@MessageBox(„(c) Trendverlag 1998“, „INFO“, 48)

Damit haben wir auch der zweiten Ereignisschleife des Programms die noch notwendigen Einträge hinzugefügt. (Abb. 6)

6 Die Programmierung der Ereignisprozeduren

Nun müssen noch die Events, die von den Menübefehlen *Bildschirm* und *Drucken* eingeleitet werden, in einer eigenen Prozedur definiert werden. Zunächst zur Option *Bildschirm*: Was ist der Sinn und Zweck unseres Programms? Da die bloße Wiedergabe eines Namens nicht sonderlich interessant erscheint, lassen wir uns eine effektvolle Stilisierung einfallen. Wie wär's mit einer spiralförmigen Anordnung der Buchstaben, die zugleich immer größer werden? Probieren Sie es aus: Für beide Funktionen sollten wir gleich zu Beginn der Prozedur eine lokale Variable deklarieren. Wir positionieren den Cursor also nach der Dialog-Prozedur in eine neue Programmzeile und geben die ersten beiden Zeilen ein.

Proc Bildschirm

Declare Winkel%, Groesse%

Damit das Fenster vor der Textausgabe gelöscht wird, fügen wir noch in einer eigenen Zeile den Befehl CLS ein. Nun sollten wir noch entscheiden, ob und in welcher Farbe der Name angezeigt werden soll. Diese Aufgabe übernimmt der Befehl TEXT-COLOR; für die Farbspezifizierung fügen wir die nach dem Farbmodell benannte

Funktion @RGB mit den jeweiligen Farbwerten im Klammersausdruck an. Für einen schwarzen Schriftzug beispielsweise geben Sie einfach dreimal die Null ein. Der zweite Parameter von TEXTCOLOR bewirkt die Hintergrundfarbe. Eine -1 bedeutet „transparent“, d.h. es wird die Hinter-grundfarbe des Fensters übernommen. So heißt die nächste Programmzeile

TEXTCOLOR @RGB (0, 0, 0), -1

Als nächstes wird die Spirale gestaltet. Eine spiralförmige Anordnung bedeutet ja nichts anderes, als daß der Namenszug kreisförmig in regelmäßigen Winkelabständen arrangiert wird, bis die volle Umdrehung vollendet ist. Den Winkel der Textausgabe bestimmt der Befehl ORIENTATION, wobei unbedingt zu beachten ist, daß die Parameter in Zehntel-Grad angegeben werden. Wenn wir also mit der Zeile

ORIENTATION Winkel%

die Variable *Winkel%* ins Spiel bringen, müssen wir bei der Definition dieser Variablen dafür sorgen, daß diese den gewünschten Gradwert verzehnfacht. Konkret: Wenn wir der Variablen einen Winkel von 15° zuweisen, lautet die Definition, die natürlich vor dem Befehl ORIENTATION stehen muß,

Let Winkel% = 150

Nun müssen wir noch eine Ereignis-Schleife definieren, in der der Winkel, ausgehend von 0, solange um jeweils 15° hochgezählt wird, bis die 360° eines geschlossenen Kreises erreicht sind. Eingedenk der Zehntel-Grad-Angabe eröffnen wir die Schleife mit

WhileNot @Gt (Winkel%, 3600)

wobei die mathematische Funktion @Gt nichts anderes besagt, als daß die Schleife solange weiterläuft, bis der erste Wert in der Klammer den zweiten Wert erreicht.

Bislang fehlen noch immer drei Angaben: So müssen wir dem Programm mitteilen, daß der Schriftzug mit jeder Ausgabe größer werden soll. Dazu weisen wir der bereits deklarierten Variable *Groesse%* die Funktion @Div zu, die den ersten Wert im Klammersausdruck durch den zweiten Wert teilt. Wenn wir also die Zeile

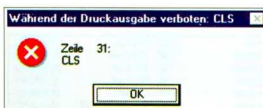
Let Groesse% = @Div (Winkel%, 100)

einfügen, bedeutet dies, daß die Schriftgröße von 0 bis auf 36 Punkt ansteigt. Allerdings müssen wir diesen steigenden Wert auch tatsächlich noch der Definition der Schriftart, der zweiten noch ausstehenden Angabe, hinzufügen. Die Schriftart stellen wir mit dem Befehl USEFONT ein. Die Online-Referenz klärt uns über die nötigen Parameter auf – Fontname, Schriftgröße, -breite, fett, kursiv, unterstrichen –, so daß wir definieren können

USEFONT „Times New Roman“, Groesse%, 0, 0, 0, 0

Nun müssen wir noch bestimmen, an welcher Bildschirmposition die Ausgabe des Namens einsetzen soll. Mit dem Befehl DRAWTEXT ist dies im Handumdrehen erledigt. Als Parameter geben wir zunächst die X-Koordinate, dann die Y-Koordinate und dann – wichtig! – den Text an, der angezeigt werden soll. Wie Sie sich erinnern,

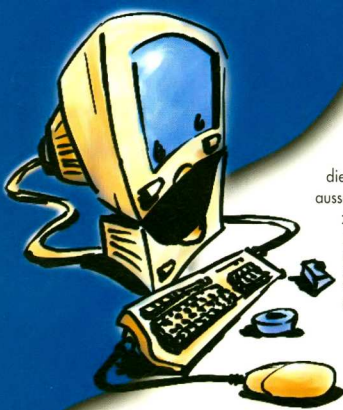
Experimentieren Sie ruhig mit den einzelnen Parametern der Schriftfarbe (TEXTCOLOR), Schriftart (USEFONT) und den Winkelwerten (Winkel%, Add Winkel%)! Korrekturen können in *RGH-PROFAN* in Sekundenschnelle abgespeichert oder wieder rückgängig gemacht werden, so daß Sie keinerlei Risiko eingehen.



Der Fehler steckt bekanntlich im Detail: Ein abschließender Probelauf mit dem Interpreter kann noch so manche Überraschung aufdecken. Bitte testen Sie Ihren Quellcode mehrmals und in allen Bereichen der Anwendung, um „Bugs“ auszuschließen.

Abb. 9





haben wir den eingegebenen Namen bereits in der Variablen `Zeile$` definiert, so daß die Programmzeile folgendermaßen aussehen kann:

```
DRAWTEXT 250, 180, Zeile$
! Ist die Schleife nun endlich fertig?
Fast – eine Kleinigkeit fehlt noch.
Zwar haben wir den Winkel mit 15°
definiert, aber noch fehlt der Befehl,
daß der Winkel nach jeder Namens-
ausgabe tatsächlich in einem
Abstand von 15° hochgezählt
werden soll. Diese Anweisung
erledigen wir mit dem Befehl
ADD Winkel%, 150
```

Eine abschließende Überprüfung des Listings unserer Bildschirm-Prozedur verrät, daß die Befehlszeile `ORIENTATION Winkel%` nach an der falschen Stelle steht: Die Winkelangabe muß natürlich in der `WHILE`-Schleife erfolgen, so daß wir die gesamte Zeile sinnvollerweise hinter der Definition der Variable `Groesse%` einfügen. Jetzt endlich können wir die Schleife und die Prozedur abschließen, so daß das Listing für unsere Namensausgabe auf dem Bildschirm folgendermaßen aussieht:

```
Proc Bildschirm
  Declare Winkel%, Groesse%
  Cls
  TextColor @RGB(0,0,0), -1
  Let Winkel% = 150
  WhileNot @Gt(Winkel%, 3600)
    Let Groesse% = @Div(Winkel%, 100)
    Orientation Winkel%
    UseFont „Times New Roman“, Groesse%, 0,0,0,0
    DrawText 250, 180, Zeile$
    Add Winkel%, 150
  Wend
EndProc
```

Aller (Programmier-)Anfang ist schwer, doch zum Glück ist die noch ausstehende Druck-Prozedur wesentlich einfacher. In `RGH-PROFAN 4.5` steht erstmals das Befehlspaar `STARTPRINT...ENDPRINT` zur Verfügung: Damit werden alle zwischen diese beiden Befehle eingefügten grafischen Bildschirmausgaben auf den angeschlossenen Drucker geschickt. Wir brauchen uns also nicht lange mit der Definition aufzuhalten, sondern schreiben einfach

```
PROC Drucken
  StartPrint
  Bildschirm
  EndPrint
```

ENDPROC

Haben wir nun endlich an alles gedacht? Der Testlauf mit dem Interpreter wird es uns zeigen. Leider müssen doch noch einige Korrekturen angebracht werden. So ist beispielsweise die Variable `Zeile$` noch nicht deklariert worden. Also fügen wir hinter der ersten globalen Variable `_Ende%` noch eine zweite ein:

```
Declare _Ende%, Zeile$
```

Doch dann meldet das Programm beim Testlauf noch einen gravierenderen Fehler: Der Bildschirm-Löschbefehl `CLS` wird beim Drucken nicht akzeptiert. So bleibt uns nichts anderes übrig, als nochmals eine weitere Prozedur für die Bildschirmanzeige anzulegen. Wir löschen also den `CLS`-Befehl aus der Bildschirm-Prozedur und ergänzen unseren Quellcode durch eine neue Prozedur:

PROC Ausgabe

```
CLS
Bildschirm
```

ENDPROC

Natürlich dürfen wir nicht vergessen, den Namen der Prozedur, die durch den Menübefehl „Bildschirm“ in der Schleife des Hauptfensters aufgerufen wird, entsprechend anzupassen, so daß diese `ELSEIF`-Bedingung nun korrekt lauten muß:

```
ELSEIF @MenuItem (102)
```

Ausgabe

Und da unser Programm natürlich nicht den Namen „Interpreter“ oder die Dialogbox nicht den Namen „Dialog“ haben soll, geben wir dem Kind in den Zeilen `WINDOWTITLE` bzw. im Handle der Funktion `@createdialog` noch einen vernünftigen Namen. Obwohl der abschließende Test im Interpreter fehlerfrei verläuft, ist es dennoch nicht verkehrt, einen allerletzten Kontrolldurchlauf im `TRACE`-Modus zu starten, um alle Eventualitäten auszuschließen... (Abb. 7, 8, 9)

7 Die Erstellung der fertigen EXE-Datei

Im letzten Schritt wird aus dem fertigen Quellcode ein eigenständiges Programm erzeugt, das als EXE-Datei beliebig weitergegeben werden kann und keine weiteren DLLs oder Runtime-Module benötigt.

Zunächst sollten wir sicherstellen, daß der Pfad zum Runtime-Modul korrekt eingestellt ist. Mit dem Befehl `OPTIONS/ Runtime-Modul des Projektes` rufen wir das Runtime-Modul auf – die Datei `Profan.exe` aus dem Verzeichnis `PROFAN 4.5`. Nun können wir mit dem Befehl `PROFAN/Kompilieren` – oder mit der Taste [F8] – das Programm kompilieren. Dabei wird ein Zwischencode erzeugt, in dem u.a. Befehls- und Prozedurnamen durch sogenannte Tokens und Prozeduraufrufe durch ihre Adressen ersetzt werden. Das macht den Code kompakter und – je nach Programm – bis zu 20mal schneller als im Interpreter-Modus. Als Ergebnis erhalten wir eine Datei im *.PRC-Format. Nun trennt uns nur noch ein Mausklick von unserer fertigen Anwendung: Linken Sie die PRC-Datei mit dem Befehl `PROFAN/Exe-Datei erstellen...` zu einer ausführbaren Programmdatei, indem Sie im Datei-Dialog die entsprechende PRC-Datei angeben. Kurz darauf meldet der Linker Vollzug. Herzlichen Glückwunsch – Ihr erstes Windowsprogramm ist fertig! Sie können die EXE-Datei separat starten, in einen anderen Ordner verschieben, an Freunde weitergeben – oder sich auch gleich an die Programmierung einer neuen Anwendung mit `RGH-PROFAN` machen... (Abb. 10, 11, 12, 13)

Der letzte Schritt: Mit dem Befehl `PROFAN/Exe-Datei erstellen` wird eine ausführbare und eigenständige Programmdatei erstellt. Dennoch sollten Sie natürlich den Quellcode im *.PRF-Format für eine eventuelle, spätere Bearbeitung gut aufbewahren. Abschließend steht einem ersten Rundgang in der fertigen Anwendung nichts mehr im Wege!

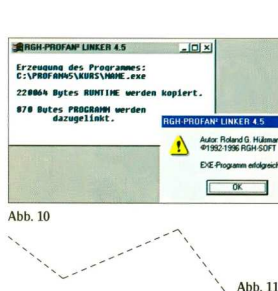


Abb. 10

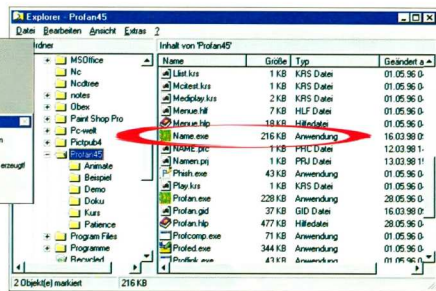


Abb. 11

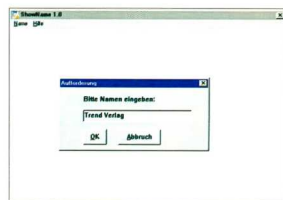


Abb. 12



Abb. 13

Das besondere Angebot für unsere Leser

RGH-Profan 6.0



Inkl. 16- und 32-Bit-Version!

Profan 6

Da bekommen Windows-Programmierer glänzende Augen: Das neue RGH-Profan 6.0 ist in einem attraktiven CD-ROM-Paket sowohl als 16-Bit-Version als auch als 32-Bit-Version enthalten. Hinzu kommt, daß ein und derselbe Quellcode ohne zusätzliche Änderungen sowohl unter Windows 3.1 als auch unter Windows 95 und Windows NT fehlerfrei läuft!

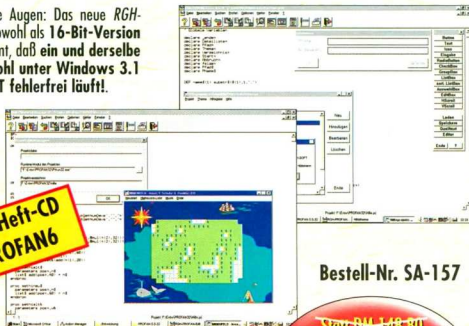
Neben der bewährten, übersichtlichen Entwicklungsumgebung hat RGH-Profan 6.0 vor allem den direkten Zugriff auf die Ressourcen von Windows verbessert: So können Sie Menüs und Dialog-Elemente aus DLLs und EXE-Dateien herauslesen und unter Windows 95 auf die Einträge in den Registry-Dateien zugreifen. Sicheres und professionelles Programmieren wird mit RGH-Profan 6.0 jetzt noch einfacher!

Überzeugen Sie sich von den leistungsstarken Features des neuen RGH-Profan:

- Voller, direkter Zugriff auf Windows-Ressourcen
- Hilfe-Generator zur Erzeugung von Windows-Hilfedateien
- Timer zur Programmierung zeitgesteuerter Ereignisse
- Sicheres Profan-Skript, eine auf PROFAN basierende Sprache zur Erstellung von Internetseiten
- Unterstützung dreidimensionaler Arrays
- Stark erweiterte Programmiersprache, die u.a. die TWAIN-Schnittstelle für Scanner und alle gängigen Bilddateiformate unterstützt
- Inkl. Bonuspack mit Tools, Beispieldateien und zahlreichen Quellcodes

Bei diesem Angebot sollten Sie zugreifen: Als lizenzierter Leser und Anwender dieses FAST GESCHENKT GOLD-Hefes erhalten Sie beim Kauf von RGH-Profan 6.0 den Heftpreis dieser Ausgabe voll angerechnet! Sie bezahlen somit für die deutsche Original-Version auf CD-ROM statt 148,80 DM nur noch den Update-Vorzugspreis von 128,80 DM.

Mehr Infos auf der Heft-CD im Verzeichnis PROFAN6



Bestell-Nr. SA-157

Statt DM 148,80
nur DM
128,80

OHS-Visualis Standard

Die professionelle Entwicklungsumgebung für RGH-Profan, die Ihnen Hunderte Stunden an Entwicklungsarbeit erspart! In Verbindung mit Ihrer RGH-Profan-Version erschließen sich Ihnen hiermit folgende neue Features:

- Workbench
- Editor mit Projektverwaltung
- Optimizer für schnelleren, kompakteren Code
- Riesige Profan-Quellcode-Bibliothek

- Applikationsgenerator
- Menuwizard
- Listingdruck
- Zeichentabelle
- Farbinfo
- Preprozessor
- Bigdat

Demoversion auf der Heft-CD!

OHS-Visualis Standard erhalten Sie als deutsche Originalversion auf CD-ROM inkl. deutschem Handbuch für besonders günstige 68,80 DM

Bestell-Nr. PK-155

Systemvoraussetzungen:
Windows-PC mit bereits installiertem RGH-Profan.

nur DM
68,80

WICHTIG! Bitte verwenden Sie für Ihre Bestellung den untenstehenden Bestellschein und füllen Sie den umseitigen Registrierungsschein als Update-Berechtigungsnachweis ebenfalls mit aus.

Bestellschein

Ja, ich möchte Ihr Angebot nutzen und bestelle hiermit:

LIEFERN SIE MIR GEGEN

(Versandkosten in Klammern)

- ☐ Bankinzug (+ DM 6,90)
- ☐ Scheck liegt bei (+ DM 7,90)
- ☐ Nachnahme (+ DM 9,90)
- ☐ Rechnung (+ DM 11,90)

▲ (Nur Großfirmen / öffentl. Institutionen mit offiz. Bestellung)

Bitte geben Sie hier Ihre Bankverbindung an!

BLZ

Kto.



PEARL Agency Allgemeine Vermittlungsgesellschaft mbH

PEARL-Straße 1

D-79426 Buggingen

Persönliche Bestellannahme: 0180-55582

Rund um die Uhr * An allen Tagen
Telefax: (076 31) 360-444 BTX *pearl# CompuServe: GO PEARL

Auf Bestellungen unter einem Auftragswert von DM 30,- erheben wir einen Mindermengenzuschlag von DM 4,- für Druckfehler übernehmen wir keine Haftung. Geringfügige Änderungen des Lieferumfangs oder des Produkt Designs behalten wir uns vor!

Kunden-Nr. (falls vorhanden)

Anzahl	Produkt	Best.-Nr.	je DM
	RGH-Profan 6.0	SA-157	128,80
	OHS-Visualis Standard	PK-155	68,80

Absender

Vorname Nachname

Straße / Hausnummer

Land / neue PLZ Ort

Datum Unterschrift

PEARL Schweiz GmbH
Freilagerstrasse 1
CH-4023 Basel
Tel: 084-888 77 88
Fax: +41-61-333 11 44

PEARL Österreich
Hauptstraße 6
A-3441 Baumgarten
Tel: 0640-5214
Fax: 02274-737 15

LIZENZURKUNDE

Hiermit wird bestätigt, daß Sie mit dem Kauf der Zeitschrift „Fast Geschenk Gold“ und der darin enthaltenen CD berechtigt sind, das Programm

RGH Profan 4.5

als Lizenzinhaber und damit rechtmäßiger Anwender in vollem Umfang und zeitlich unbegrenzt zu nutzen.

Als kommerzielles Lizenzprodukt der Firma RGH-Soft unterliegt das Programm RGH Profan 4.5 – im Unterschied zur Shareware – rechtlichen Bedingungen für die Programmnutzung, wie sie bei jedem kommerziellen Programm üblich sind.

Als lizenzierter Anwender verpflichten Sie sich, die folgenden Bestimmungen einzuhalten:

- 1 Geltung der Lizenz:** Der Lizenznehmer ist berechtigt, das registrierte Programm uneingeschränkt auf einem Arbeitsplatz-Rechner zu nutzen. Für den Einsatz auf verschiedenen Rechnern sind weitere Lizenzen erforderlich.
- 2 Schutz der Software:** Der Anwender ist berechtigt, eine Archivkopie zum Zwecke der Datensicherung zu erstellen. Weitere Kopien und insbesondere die Weitergabe von Kopien an Dritte verletzen das Urheberrecht und sind untersagt. Der Verleih sowie jegliche Abänderung oder Modifizierung des Programms sind nicht gestattet. Alle Urheber- und Warenzeichenrechte verbleiben bei der Firma RGH-Soft.
- 3 Haftung:** Für den Fall, daß die Programm-CD einen Materialfehler aufweist, erhalten Sie kostenlosen Ersatz. Senden Sie den defekten Datenträger zusammen mit dem Registrierungsschein sowie einem frankierten (DM 3,-) und adressierten Rückumschlag an die umseitige Adresse. Weitergehende Garantien und Haftungsansprüche sind ausgeschlossen.
- 4 Registrierung:** Der Eingang des ausgefüllten Registrierungsscheins ist die Voraussetzung dafür, daß Sie berechtigter Lizenzinhaber des Programms RGH Profan 4.5 werden und damit auch das Anrecht auf vergünstigte Updates erwerben.

Bitte tragen Sie hier als Lizenznachweis das Datum der Absendung Ihres Registrierungsscheins ein.

Abgesandt am:

Unterschrift

Bitte beachten Sie, daß die Heft-CD über RGH Profan 4.5 hinaus Shareware- und Freewareprogramme enthält, für die eigene lizenzrechtliche Bestimmungen gelten (siehe jeweilige Readme- und Hilfedateien).

REGISTRIERUNGSSCHEIN

fast geschenkt GOLD 9

Ich habe eine Lizenzversion des Programmes RGH Profan 4.5 mit dem Kauf der Zeitschrift „Fast Geschenk Gold 9“ erworben und die geltenden Lizenzbestimmungen für die rechtmäßige Nutzung zur Kenntnis genommen. Mit der Absendung dieses Registrierungsscheins werde ich lizenzierter Anwender und habe damit auch Anspruch auf vergünstigte Update-Angebote.

Ich wünsche künftig unverbindlich über Updates und Produktneuheiten informiert zu werden: ☐ ja ☐ nein

Vor- und Zuname

Straße

PLZ/Ort

Datum und Unterschrift

Bitte senden Sie diesen Registrierungsschein an die umseitige Adresse:

PEARL

(Passend für Fensterbriefumschlag)

**Nichts wird mehr so
sein, wie es einmal war!**

NEU!



**ab 9. April '98
im Zeitschriftenhandel**

RGH-PROFAN 4.5

IHR EINSTIEG IN DIE WINDOWS-PROGRAMMIERUNG

Programmieren unter Windows – **jetzt einfacher denn je** mit **RGH-PROFAN**: Die ebenso komfortable wie anwenderfreundliche Programmiersprache für Windows setzt **kein Insider-Wissen** und **keine besonderen Programmierkenntnisse** voraus. **RGH-PROFAN** kombiniert die Vorzüge einer prozeduralen Programmierung mit den Möglichkeiten einer grafisch orientierten

Entwicklungsumgebung: Im **PROFAN**-Editor geben Sie den Quellcode ein, und mit den Dialog-Designern entwerfen Sie die Kontrollelemente für die interaktive Programmsteuerung unter Windows. Das Ergebnis ist ein **voll funktionsfähiges Windows-Programm**, das mit dem internen **Interpreter** ausgeführt und mit dem **Compiler** und **Linker** zum Laufen gebracht werden kann.

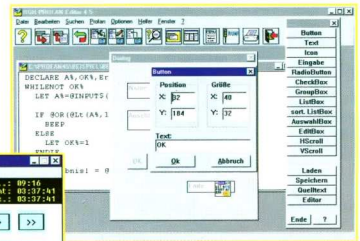
RGH-PROFAN ist nicht nur die ideale Programmiersprache für den **Einstieg und Umstieg von DOS zu Windows**, sondern hat auch erfahrenen Windows-Entwicklern manches zu bieten:

- ✗ Die **einfache und verständliche Befehlssyntax** – in Anlehnung an **BASIC** – erleichtert den Umstieg und Einstieg in die Windows-Programmierung
- ✗ **Programmbefehle** können im **Text- und Grafikmodus** eingegeben werden
- ✗ Die **mächtigen Grafik-Befehle**, die direkt auf den Grafik-Kern von Windows zugreifen, nutzen alle Vorzüge der grafischen Benutzeroberfläche
- ✗ **Fertige Dialoge und Dialog-Designer** minimieren den Programmieraufwand bei der Oberflächen-gestaltung
- ✗ Mit den **umfangreichen Datenbankfunktionen** können komplette **dBASE™ III-**kompatible Datenbank-Applikationen entwickelt werden
- ✗ Über die **MCI-Schnittstelle** lassen sich alle in Windows integrierbaren **Multimedia-Player** ansteuern. Die Untermalung mit **Sound** in Applikationen und Spielen ist ebenso einfach zu realisieren wie das Abspielen von Musik-CDs und WAV-Dateien
- ✗ Mit dem **Interpreter** werden die fertigen Programm schrittweise ausgeführt und getestet, **Compiler** und **Linker** erzeugen daraus **eigenständige EXE-Files**.

für Windows



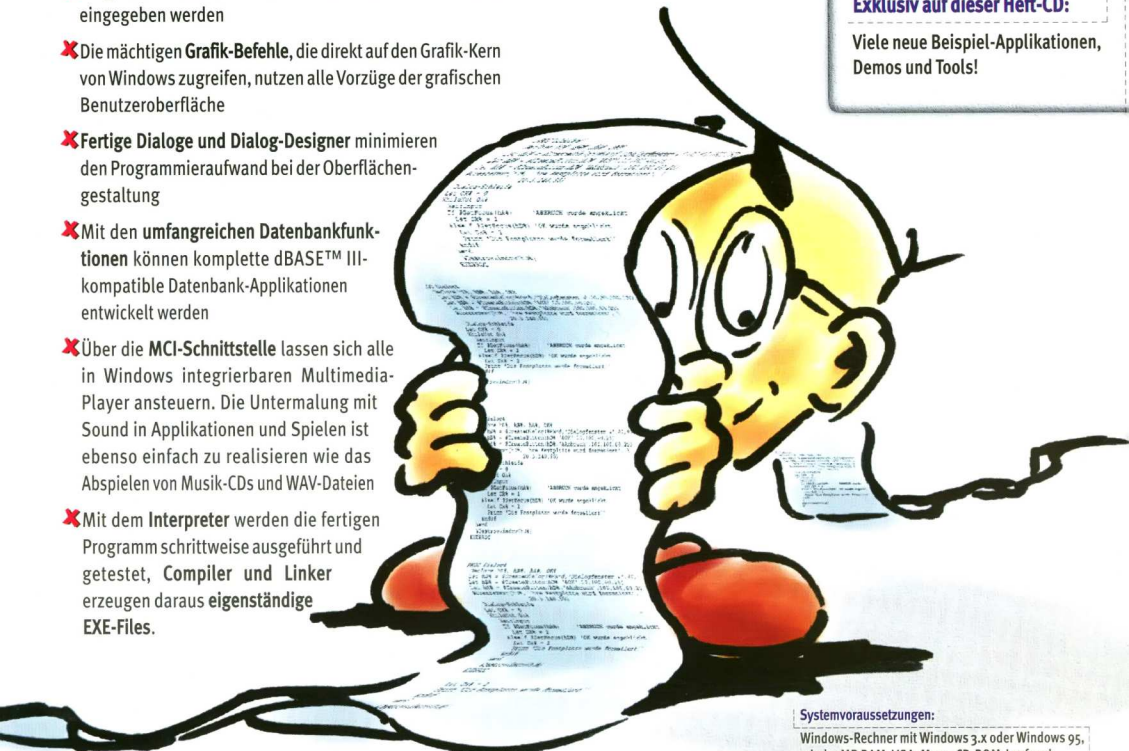
Lizenziertes
deutsches Originalprogramm
von RGH-Soft



Extra-Bonus

Exklusiv auf dieser Heft-CD:

Viele neue Beispiel-Applikationen,
Demos und Tools!



Systemvoraussetzungen:

Windows-Rechner mit Windows 3.x oder Windows 95,
mind. 4MB RAM, VGA, Maus, CD-ROM-Laufwerk.