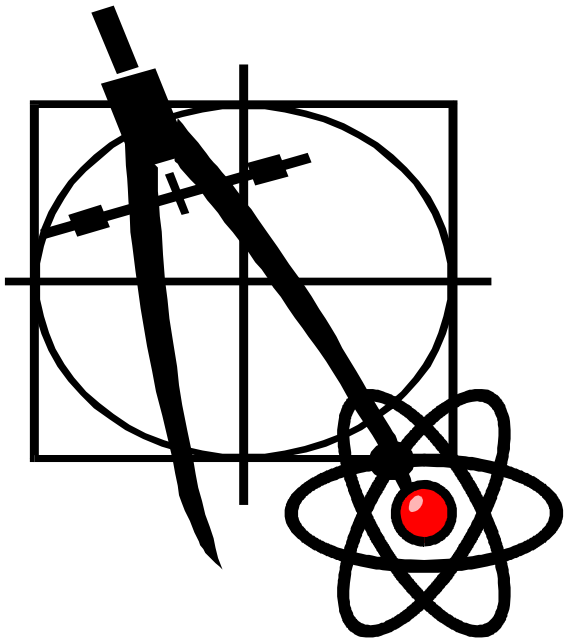
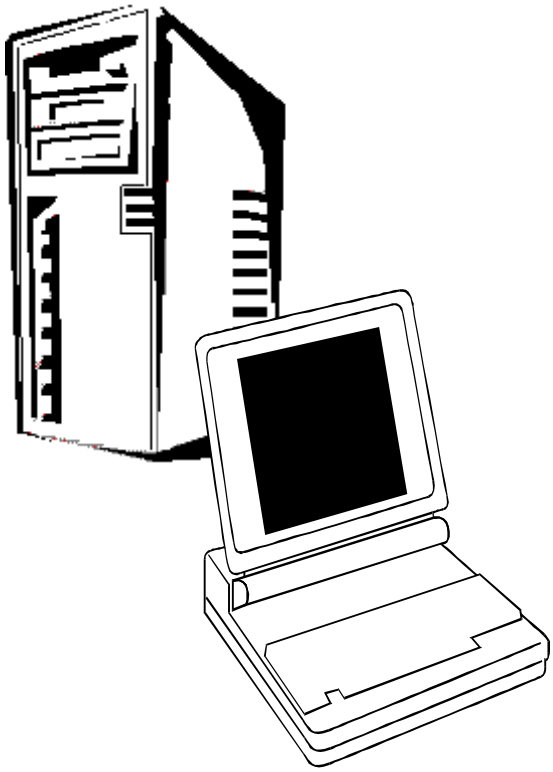
VIERTE DIMENSION

**für Wissenschaft und Technik, für kommerzielle EDV,
für MSR-Technik, für den interessierten Hobbyisten.**





In dieser Ausgabe:

15. Jahrgang - **Ausg. 3/’99** - 10,- DM

Organ der FORTH-Gesellschaft e.V.

Dienstleistungen und Produkte fördernder Mitglieder des Vereins

FORTH - Shirt

Räumungsverkauf

T - Shirt: hellgrau / grün
in Größe M-L-XL **15 DM**
Sweat-Shirt: grau / grün
in Größe M-L-XL **25 DM**
(+ Porto)

ForthWORKS
Ulrike Schnitter
Nelkenstr. 52
85716 Unterschleißheim
fon/fax 089-310 33 85

Hier könnte IHRE Anzeige stehen

Setzen Sie sich doch einfach einmal mit dem Büro der Forthgesellschaft e.V. in Verbindung.

Dipl.-Ing. Arndt Klingelnberg

Tel.: ++32 +87 -63 09 89 (Fax: -63 09 88)
Waldring 23, B-4730 Hauset, Belgien
akg@aachen.forth-ev.de

Computergestützte Meßtechnik und Qualitätskontrolle, Fuzzy, Datalogger, Elektroakustik (HiFi), MusiCassette HighSpeedDuplicating, Tonband, (engl.) Dokumentationen und Bedienungsanleitungen

Forth Engineering
Dr. Wolf Wejgaard

Tel.: +41 41 377 3774 - Fax: +41 41 377 4774
Neuhöflirain 10
CH-6045 Meggen http://holonforth.com

Wir konzentrieren uns auf Forschung und Weiterentwicklung des Forth-Prinzips und offerieren HolonForth, ein interaktives Forth Cross-Entwicklungssystem mit ungewöhnlichen Eigenschaften. HolonForth ist erhältlich für 80x86, 68HC11 und 68300 Zielprozessoren.

KIMA Echtzeitsysteme GmbH

Tel.: 02461/690-380
Fax: 02461/690-387 oder -100
Karl-Heinz-Beckurtz-Str. 13
52428 Jülich

Automatisierungstechnik: Fortgeschrittene Steuerungen für die Verfahrenstechnik, Schaltanlagenbau, Projektierung, Sensorik, Maschinenüberwachungen. Echtzeitrechnersysteme: für Werkzeug- und Sondermaschinen, Fuzzy Logic

FORTecH Software
Entwicklungsbüro Dr.-Ing. Egmont Woitzel

Joachim-Jungius-Straße 9 D-18059 Rostock
Tel.: (0381) 405 94 72 Fax: (0381) 405 94 71

PC-basierte Forth-Entwicklungswerkzeuge, comFORTH für Windows und eingebettete und verteilte Systeme. Softwareentwicklung für Windows und Mikrocontroller mit Forth, C/C++, Delphi und Basic. Entwicklung von Gerätetreibern und Kommunikationssoftware für Windows 3.1, Windows95 und WindowsNT. Beratung zu Software-/Systementwurf. Mehr als 15 Jahre Erfahrung.

Ingenieurbüro
Dipl.-Ing. Wolfgang Allinger

Tel.: (+Fax) 0+212-66811
Brander Weg 6
D-42699 Solingen

Entwicklung von µC, HW+SW, Embedded Controller, Echtzeitsysteme 1-60 Computer, Forth+Assembler PC / 8031 / 80C166 / RTX 2000 / Z80 ... für extreme Einsatzbedingungen in Walzwerken, KKW, Medizin, Verkehr / >20 Jahre Erfahrung.

Ingenieurbüro
Klaus Kohl

Tel.: 08233-30 524 Fax: —9971
Postfach 1173
D-86404 Mering

FORTH-Software (volksFORTH, KKFORTH und viele PD-Versionen). FORTH-Hardware (z.B. Super8) und -Literaturservice. Professionelle Entwicklung für Steuerungs- und Meßtechnik.

Inhalt

In der nächsten Ausgabe finden Sie voraussichtlich:

- den ersten Teil eines Aufsatzes über das ‘Hashing’; von Friederich Prinz
- einen Artikel CFA2NAME; von Wolfgang Allinger
- und hoffentlich wieder viele Leserbriefe

Editorial

IMPRESSUM

Name der Zeitschrift

Vierte Dimension

Herausgeberin

Forth-Gesellschaft e.V.
Postfach 16 12 04
D-18025 Rostock
Tel.: 0381-400 78 28
E-Mail:
SECRETARY@FORTH-EV.DE
DIREKTORIUM@FORTH-EV.DE

Bankverbindung: Postbank Hamburg
BLZ 200 100 20
Kto 563 211 208

Redaktion & Layout

Friederich Prinz
Hombergerstraße 335
47443 Moers
Tel./Fax:: 02841-58 3 98
E-Mail:
VD@FORTH-EV.DE
FRIEDERICH.PRINZ@T-ONLINE.DE

Anzeigenverwaltung

Büro der Herausgeberin

Redaktionsschluß 1999

März, Juni, September, Dezember
jeweils in der dritten Woche

Erscheinungsweise

1 Ausgabe / Quartal

Einzelpreis

DM 10,- zzgl. Porto u. Verp.

Manuskripte und Rechte

Berücksichtigt werden alle eingesandten Manuskripte. Leserbriefe können ohne Rücksprache gekürzt wiedergegeben werden. Für die mit dem namen des Verfassers gekennzeichneten Beiträge übernimmt die Redaktion lediglich die presserechtliche Verantwortung. Die in diesem Magazin veröffentlichten Beiträge sind urheberrechtlich geschützt. Übersetzung, Vervielfältigung, Nachdruck sowie Speicherung auf beliebigen Medien ist auszugsweise nur mit genauer Quellenangabe erlaubt. Die eingereichten Beiträge müssen frei von Ansprüchen Dritter sein. Veröffentlichte Programme gehen - soweit nichts anderes vermerkt ist - in die Public Domain über. Für Fehler im Text, in Schaltbildern, Aufbauskizzen u.ä., die zum Nichtfunktionieren oder eventuellem Schadhaftwerden von Bauelementen oder Geräten führen, kann keine Haftung übernommen werden. Sämtliche Veröffentlichungen erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Liebe Leser,

die vorliegende Ausgabe der VD wird die allermeisten von Ihnen mitten in der Ferien- und Urlaubszeit erreichen. Einige von Ihnen werden ihren Urlaub bereits hinter sich haben - und sich hoffentlich über die VD freuen, die nach der verdienten Erholung neue Anregungen und Informationen rund um Forth und aus der Forthgesellschaft bietet. Andere werden die VD vielleicht mit an ihren Urlaubsort nehmen, um sie dort in Ruhe zu ‘studieren’ und, so hoffen wir, sofort den einen oder anderen Gedanken zu Papier und auf den Weg in unsere Redaktion zu bringen.

Aber wann, wo und wie auch immer Sie diese VD lesen - wir wünschen Ihnen einen erholsamen Urlaub und schöne Ferien (gehabt zu haben)!

Ein ‘kleiner Urlaub’ war bereits die diesjährige Jahrestagung der Forthgesellschaft. Die einhellige Meinung zu der ganz hervorragend organisiert gewesenen Veranstaltung ist: “es kann kaum schöner werden”! Selbstverständlich stand Forth nicht nur auf dem Papier im Vordergrund, aber Ulrike und Heinz Schnitter haben bestens besorgt, daß kulinarische, kulturelle und ‘sportliche’ Genüsse einen gebührenden Platz im Programm hatten. Mit besonderer Freude auf der Seite der Organisatoren läßt sich hierzu vermelden, daß das Angebot des ‘zusätzlichen Tages’ verstärkt angenommen wird. Es scheint den Mitgliedern der FG ein wachsendes Bedürfnis zu sein, sich ‘einfach so’ in geselliger Runde zu treffen und sich auch über Forth hinaus auszutauschen.

Bezüglich des ‘forthigen’ Austausches waren in Oberammergau Stimmen zu hören, die zwar nicht den absehbaren Untergang von C++ berichten konnten, wohl aber darüber, daß sich international ein gesteigerter Unmut und entsprechendes Nachdenken über die Nachteile breit macht, denen sich Entwickler bei der Nutzung dieser ‘Sprache’ verstärkt ausgesetzt sehen. Die hierzu am Rande der Tagung geführten Diskussionen bestärken die im Editorial der letzten Ausgabe zum Ausdruck gebrachten Hoffnungen auf eine (notwendige) Rückkehr zu angemessener Einfachheit. Es geht vorwärts !

Friederich Prinz

VD im Internet

Die VD auch im Internet wiederfinden zu können, wird in den Netzwerkforen, in E-Mails an die Redaktion und auch in persönlichen Gesprächen immer wieder als Wunsch geäußert. Unsere Zeitschrift ist nicht nur für die Mitglieder der Forthgesellschaft interessant ! Es spricht auch wenig dagegen, die VD - mit entsprechender, zeitlicher Verzögerung - ‘allgemein zugänglich’ zu machen. Woran es scheitert ist einmal mehr der Wille zur Tat, sprich: Wir brauchen Jemanden, der uns 4 Mal im Jahr die dazu notwendige Arbeit machen will !

fep

Quelltext Service

Die Quelltexte in der VD müssen Sie nicht abtippen. Sie können diese Texte auch direkt bei uns anfordern und sich zum Beispiel per E-Mail schicken lassen. Schreiben Sie dazu einfach eine E-Mail an die Redaktionsadresse.

fep

Neues aus der FIG SV

Friederich, Danke für die Ostergrüße, und wir wünschen auch Dir das Beste zu Ostern!

(Nun ja, dieser Brief erreicht die Leser der VD etwas spät. Aber als dieser Brief mich erreichte, war die VD 2/99 b e - reits in der Vervielfältigung - fep)

Etwa vor einer Woche fand ein SVFIG-Treffen statt, zu dessen Beginn ich mich sehr wichtig fühlte, insofern daß ich zu Beginn 10% der Teilnehmer bildete. Am Ende des Tages war ich allerdings unten bei 5% angelangt!

Wie ich schon früher gesagt habe, gehe ich, obwohl Computer-Wissenschaften nicht mein Beruf sind, zu den Forth-Treffen hauptsächlich wegen der Unterhaltung und der Weiterbildung -- meiner Meinung nach, Weiterbildung zu den günstigsten Kosten, die man in diesem Land finden kann.

Ich habe festgestellt, daß ich üblicherweise zum SVFIG-Treffen mehr PhDs (Doktoren) in einem Raum treffe, als in all meinen Klassen in einem Tag an der Universität. Um die Mittagszeit am letzten Samstag waren, ich bin ziemlich sicher, die Hälfte der Anwesenden Doktoren der einen oder anderen Art. Nun, zumindest kann ich sagen, daß keiner von Ihnen drei alte Volvos besitzt, so wie ich!

So, was haben wir gelernt?

Der immer anwesende, immer wissende, niemals ermüdende Dr. Ting gab uns eine neue wertvolle Lektion aus der Welt der Mikrocontroller. Insbesondere behandelte er den populärsten, wie er sagte, den Motorola 6805 Mikroprozessor. Er hatte gehört, daß die Chinesen diesen Mikroprozessor nachbauen und große Marketingpläne für ihn haben. Somit, dachte er, sollte eForth auf dem 6805 zum Laufen gebracht werden, wie es schon mit dem 8051 der Fall ist. Jedoch nach einer ganzen Menge Arbeit hat Dr. Ting es aufgegeben, insbesondere da der "verkrüppelte" Stack des 6805 einfach keine anständige Implementation von Forth erlaubt.

Dr. Ting setzte mit einer erfreulicheren Bemerkung fort: Er teilte uns mit, daß er einige Worte zu seinem 8086 eForth (wie z.B. READ und LOAD) hinzugefügt hat, um seine neue Version gut genug zu machen, damit man wirklich etwas damit anfangen kann.

In der Nachmittagssitzung gab uns Dr. Everett (Skip) Carter eine Zusammenfassung über die Arbeiten und den momentanen Status der Forth Scientific Library (abgekürzt FSL). Wie Ihr vielleicht wißt, wurde sie von Skip begonnen (auf der Rochester Konferenz 1994). Man kann mehr darüber
bei einem Abstecher auf Skip's Homepage erfahren --- **www.taygeta.com**. Die Bibliothek hat einige Verbindungen zu FORTRAN, 51 Beiträge wurden in den fünf Jahren angenommen, und 60 warten auf eine Durchsicht. Es gibt da Zufallszahlen-Generatoren, Algorithmen zur Wurzelsuche, für Transformationen usw. Die wissenschaftliche Gemeinschaft beginnt die FSL in ihren Publikationen anzunehmen. Die Hauptbeiträge kommen, außer von Skip selbst, von Hendrix, Noble und Montgomery. Noch mehr Freiwillige werden benötigt, sowohl für neue Beiträge als auch zur Durchsicht.

Dr. Bob Smith (der Autor von FPC's Gleitkommapaket und anderer Dinge) bemerkte das Tom Zimmer wieder Zeit hat, um am Win32Forth zu arbeiten und Updates zu verbreiten, und das auch tut. Die aktuelle Version ist wohl 4.0. Ich bin nicht sicher, ob Bob das Folgende mit Bezug auf Win32Forth gesagt hat, aber da es auch auf viele andere Tools zutrifft, die wir erhalten haben, möchte ich es zitieren: "Es kann mehr, als was immer man damit machen möchte, und man kann es nicht dazu bringen, weniger zu tun."
("It does more than you ever want to do with it, and you can't get it to do less.")

Wie ich zu Dr. Haydon bemerkte (sofern ich weiß der einzige Doktor in unserer Gruppe, der Leuten mit medizinischen Problemen helfen kann), sagt Ted Nelson, daß derartige Tools einen Mangel, "Featuritis" genannt, haben. Und, ich glaube, daß Dr. Ting, der Mathematik an der San Francisco State University lehrt, zustimmen würde. Er befreit gern alle Dinge von überflüssigen Features unter Verwendung seines Occam'schen Rasiermessers.

Nun, wenn Dr. Beierlein all das oben gesagte in ein verständliches Deutsch übersetzen kann, werde ich Ihn sicher auf meiner Liste von Doktoren behalten, für den Fall, daß ich irgendwann einen Bedarf an einer Herzdruckmassage habe.

(Henry, I tried my best to get on your list. But the last medical slang (percutaneous myocardial reperfusion) was a very strong attack to the borders of my knowledge of the english language. I hope I got it right ;-).

-- Thomas)

Glückauf,

Henry.

übersetzt von Thomas Beierlein

Embedded 3 ist erschienen.

Embedded ist eine unregelmäßig erscheinende Zeitung, deren Inhalte sich vorwiegend auf Microcontroller und deren Programmierung konzentrieren. In dieser Ausgabe werden folgende Themen behandelt:

CORDIC (= Coordinate Rotation Digital Computer) ist ein Verfahren zur Berechnung mathematischer Funktionen. Da bei diesem Verfahren nur einfache Befehle nötig sind, kann es auf Controllern zum Einsatz kommen. Das Verfahren stellt verschiedene trigonometrische und mathematische Funktionen zur Verfügung. Als Anwendungen schlägt der Autor die Bestimmung von Körpern im Raum vor, wie sie z.B. von Werkzeugmaschinen benötigt wird.

Gray Code ist das Thema eines weiteren Artikels. Der Gray Code eignet sich für Anwendungen in der Dekodierung von Winkelcodierern besonders, weil sich bei zwei unmittelbar aufeinanderfolgenden Winkelstellungen genau ein Bit ändert,

bei der Änderung von mehr oder weniger Bits muß also ein Fehler vorliegen. Ein Auftreten von Schrittfehlern ist also ausgeschlossen.

Multitasking ist auf Controllersystemen schon lange ein Thema. Der Autor greift mit seinem Artikel über Round-Robin-Multitasker ein Problem auf, welches sicher ein Dauerbrenner ist. Allerdings sind die Probleme bzw. deren Lösungen ausschließlich auf einer Hardware verfügbar, deren Bezugsquelle verborgen bleibt, auch das verwendete Forth ist mir nicht bekannt. Im Heft 1 der embedded sind jedoch Hinweise zur Anpassung beschrieben.

Der nächste Artikel befasst sich mit CRC-Systemen. Dieses Thema ist bereits in der VD behandelt worden, allerdings halte ich die Art der Darstellung in embedded für leichter verständlich.

Weitere Artikel befassen sich mit Ergänzungen zu Schrittmotoren, zu PID-Reglern und zu Emulatorsystemen. Außerdem wird noch eine theoretische Darstellung der V24-Schnittstelle beschrieben.

Fazit. Für Anwendungen auf Microcontrollern enthält diese Zeitung außerordentlich viel Material. Sie kann jedem empfohlen werden, der sich mir dieser Thematik auseinandersetzen will.

Robert Freitag

Aus dem Netz gefischt

Von: *Hier ist uns der Autor 'abhanden gekommen'. Vielleicht ist Jemand unter den Lesern so freundlich, den Namen nachzureichen ?*

Betreff: Stacking and Structures

Using the stack is an art and a science. There are several things that I have learned which make my usage of the stack easier. One thing which I have not seen mentioned in the literature is the following technique.

Don't pass a bunch of arguments, pass a single pointer to a structure with your arguments in it. Then you can use words to access each element in that structure.

Here is an example.

```
variable position
4 constant cell-size  ( assumes cell sizes of 4 bytes
                        in your system )

: start    0 position ! ;
: finish  ;

: :pointer
    create position @ ,
    cell-size position +!
    does>
    @ +
;

( and then you do this to create your pointers )

start
  :pointer fred
  :pointer wilma
  :pointer pebbles
  :pointer barney
  :pointer betty
  :pointer bambam
finish

( and then you can create your table which will hold your Flintstones in it like this )

create bedrock   cell-size 6 * allot

( and in your code do something like this, assuming that the action words are already defined of course )

: episode   ( city  --  )
    dup fred shovels
    dup wilma sweeps
    dup pebbles plays
    dup barney wisecracks
    dup betty chats
    dup bambam destroys
    drop
;

( and you will never have to worry about using ROLL or PICK to get at BAMBAM again )

This is particulary nice when you can't use direct variables but the characters or whatever actually MUST be passed into the word. There are times when this is necessary. For example, supposed we wanted to clone or rip off the Flintstones to create a new cartoon which will tie into a new breakfast cereal and childrens stationary line and stuffed animal line and such. Lets call our spinoff the Munchkins. In this show we recycle many of the old scripts while changing the characters.

create munchkin-land   6 cell-size * allot

( and then we could go )

munchkin-land episode

( and only we need be aware that what was old is now new again )
```

Dieter Staas: **StarBasic-Programmierung**
(Hanser-Programmier-Praxis)
München; Wien; 1999
Carl Hanser-Verlag

ISBN 3-446-19585-8
DM 79.- ÖS 577,-

Um es vorwegzunehmen: Ich halte dieses Buch für ein Gutes!

Es wendet sich an Menschen, die sich bisher mit den Funktionen der verschiedenen Programme des StarOffice-Paketes der Version 5.xx beschäftigt haben und die Möglichkeiten, die StarOffice bietet, voll ausschöpfen möchten, also fortgeschrittene Beginner und Power-User.

Ich selbst gehöre nicht unbedingt zu diesen Menschen, stoße aber doch immer wieder an Grenzen von z.B. StarWriter oder StarCalc, die mit selbstgeschriebenen Makros (manchmal) zu überwinden sind.

Diese Herangehensweise verfolgt auch Dieter Staas mit seinem didaktischen Ansatz. Gleich zu Beginn des Buches wird gezeigt, wie man mit den Makro-Funktionen der StarOffice-Umgebung kleine Makros schreibt. Dies geschieht kleinschrittig und stets nachvollziehbar. Im Text verstreute Tips zur Handhabung des IDE (Integrated Development Enviroment) zeugen von der Kompetenz Dieter Staas's einerseits und seinem Willen, auch wenig erfahrenen Anwendern zu helfen. Drohende Klippen und Inkonsistenzen von StarBasic werden deutlich benannt. Dies gilt durchgängig für alle Kapitel.

Über die Verwaltung solcher Makros, ihre Zuordnung zu den einzelnen StarOffice-Modulen, geht es weiter zu der Programmierung von Subroutinen und Funktionen (die immer als Teil von Makros geschrieben werden). Programmieren wird hier gezeigt als ein Erweitern der Makros um Fähigkeiten, die sich nur durch Mausklicken etc. nicht formulieren lassen. Es ist Teil der StarOffice-Technik, daß StarBasic-Programme ohne StarOffice nicht laufen können, Stand-Alone-Programme sind also nicht möglich!

Ein 'Grundkurs' zur eigentlichen Programmiersprache Basic (Typendeklaration, Schleifen und Verzweigungen) des StarOffice-Paketes umfaßt mehr als einhundertfünfzig Seiten. Für die Qualität spricht dabei, dass Dieter Staas auf Möglichkeiten zur Geschwindigkeitsoptimierung und zur vorbeugenden Fehlerbeseitigung eingeht.

Das OO-Modell der Star-Programmierer wird erläutert. Man erfährt, was Klassen, Methoden, Eigenschaften und Referenzen sind, wie man sie ausliest und evtl. ändert.

Es wird gezeigt, wie in Programme (Makrofolgen) Schaltflächen und Dialoge eingebaut werden bzw. wie solche Dialoge wiederum mit Makros belegt werden können. Dialoge können sich hier aus all den Elementen zusammensetzen, die die Star-Programmierer selbst in ihrem Office-Paket verwendet haben (StarOffice ist zu einem guten Teil in StarBasic geschrieben). Sie sind in Funktion und Aussehen weitgehend mit den Dialogelementen der verschieden GUIs (Windows, OS2, Unix/Linux) identisch. Zu ihrem Einsatz in plattformübergreifenden Projekten liefert Dieter Staas einige Tips und Hinweise, beschränkt sich aber in Toto auf die Windowsversion. Trotzdem: hätten die Autoren meines Browsers diese Tips beherzigt - ich könnte manchen Dialog unter Linux/KDE wohl besser lesen!

Zu den einzelnen Modulen der StarOffice-Umgebung (StarWriter, StarCalc, StarBase) werden spezielle Programmtechniken und Befehle vorgeführt, deutlich sagt Dieter Staas, bei welchen (seltenen) Gelegenheiten z.B. in StarCalc die 'normalen' Programmfunktionen überlegen sind!

Einige (leider wenige) feine 'Programme', die auch auf eine Diskette gepaßt hätten, werden als 'Projekte' von der Idee bis zum fertigen Programm vorgestellt. Sie bieten gute Ausgangspunkte für eigene Ideen und Pläne. Besonders gelungen erscheint mir die Programmierung einer (Beispiel) Literaturdatenbank mit Eingabe-und Ausgabemasken.

Es ist nicht das Ziel Dieter Staas's ein Referenzbuch vorzulegen, dazu verweist er auf die Hilfefunktion von StarOffice, deren Schwächen er nicht nur deutlich benennt; er zeigt gleichzeitig auch, wie sie dennoch genutzt werden kann. Wer eine ausführliche Referenz braucht, muss sich die leider nur online erhältliche 800-seitige StarBasic-Referenz im PDF Format downloaden.

Den Abschluß des Buches bildet ein Anhang zur Fehlerbehandlung, Debuggingtechniken (in gewohnter Qualität), Shellfunktionen, Dateiaufrufen auf Betriebssystemebene. Das Stichwortverzeichnis ist ausreichend ausführlich.

mb

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

VIJGEBLAADJE
der HCC Forth-gebruikersgroep, Niederlande

Nr. 14, April 1999

Das gesamte Vijgeblaadje wird diesmal von Willem Ouwerkerk, dem Vorsitzenden der HCC-Forth-gebruikersgroep, bestritten. Bewundernswert, wieviel Energie ein einzelner Mensch aufbringen kann; schade, daß sich das Vijgeblaadje immer noch nicht wieder zum altehrwürdigen, etwas umfangreicheren Vijgeblad zurückverwandelt hat. An sachkundigen Forthlern mangelt es der Forth-gg ganz bestimmt nicht.

De comparator in de AT89C2051
Willem Ouwerkerk

Einer der Artikel, bei denen auch das Herz desjenigen höher schlägt, der sich nicht scheut, ab und zu eine saubere Lötstel-

le hinzulegen, der aber nicht soviel Tatkraft aufbringt, sich gleich eine ganze Schaltung auszudenken. Forth ist ein wunderbares Vehikel, Hard- und Software-Interessen zusammenzubringen. Der AT89C2051 wird im Igel-Projekt der holländischen Forth-Freunde (wir berichteteten darüber) verwendet. Es geht hier um den Aufbau eines Dämmerungsschalters. Natürlich in Forth programmiert, in ByteForth.

Machine Forth
Willem Ouwerkerk

Willem berichtet über eine Diskussion im Internet über Machine Forth von Charles Moore, das auf den 27 Opcodes des MuP21 aufbaut. - Demnächst soll über Color Forth, auch von Charles Moore, berichtet werden, in dem zur Kennzeichnung des Compilierzustands eines Forth-Wortes verschiedene Farben verwendet werden.

Verbeteringen in ByteForth 1.70pc
Willem Ouwerkerk

Es geht weiter mit dem Projekt ByteForth, jetzt für folgende Chips: AT89C1051(U), AT89C2051, AT89C4051, via SPI; AT89S8252, AT89S53, AT89C51, AT89C52, AT89C55. Und was kommt dann? Unter anderem wird der Einbau der CREATE-DOES>-Struktur genannt. "Viele meinen, ohne diese Struktur sei Forth kein Forth. ByteForth hat bisher das Gegenteil bewiesen", sagt der Autor.

FORTHWRITE
der FIG UK, Großbritannien

Nr. 101, April 1999

2 Forth News

Aus dem Internet: Forth für Psion - Preissturz für ProForth - SwiftForth für Schulen und Universitäten jetzt kostenlos - 4OS, ein komplettes Forth-Betriebssystem fürs Netz von iTVc, 500 KB (siehe www.itvc.com) - Delta Forth 0.8 für Java - Forth-ähnliche Java-Abart FIJI - Win32Forth v4.1 von Tom Zimmer - 486ASM für Win32Forth von Jim Schneider (jschneid@fire.he.net) - ForthMacs für StrongARM - Quartus Forth für Palm Pilot - Forth für Lego-Roboter - PowerMops 3.4 für Apple Mac - Forth-Editor BMW - MusikMANX frei mit iForth von Marcel Hendrix - Win32APIs - TechTips für MaxForth - Julia v4.0 - Benchmarks für SwiftX Forth - In England (Southampton) werden einige Forth-Programmierer gesucht.

7 FIG UK Hardware Project
Jeremy Fowell

Die englische Forth-Gesellschaft hat mehrere Gemeinschaftsprojekte zu laufen. Eines davon hat das Ziel, Frank Sergeants (FIGUK-Mitglied) Pygmy Forth auf einem Einplatinencomputer mit dem 68HC11-E1 zu implementieren. Der Autor berichtet über den Stand der Dinge und Chris Jakeman fügt ein vorläufiges Schaltbild (das nachzubauen auch ich mich ohne weiteres trauen würde) hinzu.

9 Getting stuck in(to) Win32Forth
Dave Pochin

Dave hat bisher mit Tom Zimmers 16-Bit-Versionen gearbeitet. Beim Sprung zu Win32Forth gedachte er, keine großen Schwierigkeiten anzutreffen. Pustekuchen! Spätestens bei den eingebauten OOP-Mechanismen merkt man den Riesenunterschied. Dave gibt seine Erfahrungen weiter, um die Lernkurven anderer abzukürzen.

15 Nominations for the FIG UK Awards

Zwei Preise werden ausgelobt: (1) für den besten Forth-Beitrag, (2) für den besten Beitrag in Forthwrite. Die Mitglieder werden aufgerufen, Kandidaten vorzuschlagen.

16 Vierte Dimension
Alan Wenham

Kommentierte Übersetzung der Titel der Artikel aus unserer VD 1/99.

17 Deutsche Forth-Gesellschaft

Eine Freude, unsere FG-Anzeige wieder zu sehen. Ich werde mich darum bemühen, daß das nächste Mal unsere neue Web-Adresse eingesetzt wird.

18 Forth for the Z88
Garry Lancaster

Der Autor berichtet über sein Forth für den Z88, angelehnt an CamelForth. Auf seiner Webseite liegen auch einige Z88-Emulatoren für den PC (http://www.menaxus.demon.co.uk)

19 Z88 CamelForth
Garry Lancaster

Ein Auszug aus Garrys Webseite.

23 From the 'Net - Turnkey Apps and Documentation
Ray Allwright

Ray bespricht ein paar "Threads" aus den Newsgroups. "Nicht immer leicht", sagt Ray, "den Faden wirklich zu finden - und zu halten". Einerseits viel leichtes Geplauder ohne großen Zusammenhang, andererseits profunde Debatten zwischen Experten über Probleme, die sie gerade eben erst geschaffen haben. Ein paar Beiträge darüber, wie am besten man Anwendungen fabriziert, die von der Kommandozeile aus direkt als Programme aufgerufen werden - ohne Dazwi-

schenfunken eines Forth-Interpreters. (Bisher bin ich, der Rezensent, einfach nur immer hingegangen und habe diese Dinge "gemacht", ohne dabei ein Problem zu erkennen, Parameter über das PSP gelenkt usw.) Dann noch Beiträge über den Umstand, daß manche "Operanden" eines Wortes ganz Forth-unlike HINTER das Wort geschrieben werden müssen. Wie sollte man das im Stack-Kommentar beschreiben?

29 The 'Egel Coursebook - Integrating Hardware and Software with the Atmel 8051 Family
Leendert van den Heuvel

Von Dick Willemse ins Englische übersetzt. So ungefähr das, was ich (der Rezensent) kürzlich vom holländischen Original für die VD ins Deutsche übersetzt habe. Die holländischen Forth-Freunde sind mit ihrem Hardware-Projekt für die Atmel-8051-Familie fleißig zugange. Wir freuen uns über ihre Berichte. Vielleicht können auch wir in Bälde von unserem (FG) neuen Hardware-Gemeinschaftsprojekt (auf der Forth-Tagung 1999 ins Leben gerufen) dem Rest der Welt Kunde tun (?)

32 Letters

4 Briefe. Der wichtigste davon von Graeme Dunbar. Es geht um Verbreitung von Lehrmaterial per E-Mail oder IRC.

38 Forthwrite Subject Index 1990-1998

Algorithmen (9), Anwendungen (8), Arithmetik (13), Arrays (2), Assembler (1), Block-Tools (3), Interessante Worte (11), Parallel-Verarbeitung (2), Steuerfluß (8), Datenbank (2), Entwürfe (14), Dynamische Daten (2), Editor-Tools (4), Editorials (11), Verschlüsselung (1), Ausnahmebehandlung (3), FANSI-Projekt (17), Dateiverarbeitung (8), Bruchzahlen (4), Zukünftige Entwicklungen (3), Graphik (6), Hardware (5), Geschichtliches (4), Humorvolles (3), Interfaces (5), Systembezogenes (13), Interpreter (2), Forth-Bibliothek (3), MCFAs (1), OOF (3), Leistungsmessung (1), Permutationen (3), Darstellung (8), Wahrscheinlichkeit (3), Veröffentlichungen (1), Rätsel (5), Zufallszahlen (4), Besprechungen (11), Wurzeln (6), Suchverfahren (6), Mengen (1), Sortierverfahren (3), Stacks (7), Standard-Forth (1), Endliche Zustandsautomaten (3), Strings (14), Strukturen (1), Systeme (11), Tools (19), Tutorials (17), Vektorisierung (6),

Die FIG UK unterhält einen Bibliotheksdienst. An sich nur für Mitglieder gedacht. Aber vielleicht rücken die englischen Forth-Freunde auch mal eine Kopie eines Artikels für Nicht-Mitglieder heraus. Notfalls wende man sich einfach an mich, den Rezensenten.

Fred Behringer

Verschiedenes

Direktorentreffen in Schierke, Hochharz - vom 18. bis 20. Juni haben sich die Direktoren der FG in Begleitung ihrer Familien in Schierke im Hochharz getroffen, um Angelegenheiten der FG zu besprechen. Neben den Themen der FG - die nächste Jahrestagung, eine englischsprachige Sonderausgabe der VD und der WEB-Seiten der FG - standen auch Besuche der sehr reizvollen Umgebung auf dem Programm; unter anderem eine Wanderung auf den Brocken. Dabei konnten sich die Direktoren unserer Gesellschaft davon überzeugen, daß FORTH als Entwicklungsumgebung für moderne Steuerungssoftware der HSB (Harzer Schmalspur Bahnen) nicht benötigt wird. Die Lokomotiven dieser Bahnen werden noch mit Dampf betrieben.

LOGO für die FG - die Einsendungen zu einem LOGO für die Forthgesellschaft halten sich ‘in engen Grenzen’. Zur Zeit liegen drei Vorschläge auf dem Tisch der Redaktion. Zwei dieser Vorschläge sind von ein und dem selben Mitglied. Es darf (muß) ‘noch etwas mehr kommen’.

Argumente für Forth - ‘Warum Projektverantwortliche sich für den Einsatz von Forth entscheiden sollten’, erklärt J. Reillhofer in den Web-Seiten der FG. Schauen Sie dort hinein, wenn es Ihnen gelegentlich an Argumenten mangelt. J. Reillhofer’s Argumenten kann sich auch ein Kaufmann nicht verschließen.

Tagungsbände ‘99 - wird es voraussichtlich nicht geben. Ulrike und Heinz Schnitter, die den Teilnehmern eine Tagung organisiert haben, die neue Maßstäbe setzt, haben **lange auf die zugesagten Unterlagen einzelner Referenten gewartet**. Zur ‘Drucklegung’ dieser Ausgabe der VD waren leider noch immer nicht alle Unterlagen in Unterschleißheim angekommen. Die Vorträge der Referenten der Tagung ‘99 sollen deshalb in den nächsten Ausgaben der VD veröffentlicht werden. Den Anfang macht - lobenswerter Weise - Bernd Paysan in der vorliegenden Ausgabe.

Verbunden mit dieser Information ist die Aufforderung an alle Referenten, ihre Unterlagen - möglichst kurzfristig - an die Redaktion der VD zu senden !

Die Jahrestagung 2000 - wird die Mitglieder der FG voraussichtlich **in Hamburg** versammeln. Klaus Schleisiek will diese Tagung organisieren und ausrichten. Wir freuen uns auf Hamburg !

C++ bekommt vermehrt Akzeptanzprobleme - könnte der Tenor vieler Stimmen lauten, die nicht nur in Oberammergau zu hören waren. Um es mit Goethe zu sagen:

Ich stehe mit Vergnügen da
und freue mich diesen,
von C++ da kann ich ja
auf Forth und an’dre schließen.

(frei nach Faust, Intermezzo)

Die Forth-Tagung ‘99

Forth-Tagung '99 in Oberammergau

Dieses Jahr fand die Forth-Tagung ganz im Süden von Deutschland statt, in Oberammergau. Trotz der malerischen Landschaft und der Märchenschlösser in der Umgebung, und obwohl schon Gerüchte im Umlauf waren und Zusagen gegeben wurden, kam Chuck Moore nicht zur Tagung.

Die Tagung begann am Donnerstag mit einem “Forth Day”, bei dem zunächst der Besuch der Klosterkirche (runder, barrocker Kuppelbau) und eine Führung durch die Klosterdestille auf dem Programm stand. Der zuständige Kräutermönch, der wegen seiner Abneigung dem Alkohol gegenüber in der Destillerie arbeitet (erhöht die Ausbeute), war ausgesprochen gut drauf.

So erzählte er über ein Gegengift gegen den Rittersporn, eine der giftigsten Pflanzen hierzulande -- schon ein Teelöffel Extrakt wirkt innerhalb kurzer Zeit und ist schon 2 Stunden nach dem Tod nicht mehr nachzuweisen. Das ,,Gegengift" bestehe laut einem alten Folianten aus folgenden Zutaten:

* Man nehme eine Maus, die vom Rittersporn gegessen hat.
* Von dieser Maus braucht man nur den rechten Schnei dezahn, den man pulverisiert,
* und in der Milch einer Frau, die einen Knaben säugt, aufkocht.

Kurz gesagt: Entweder fehlt die Maus, oder die Milch, jedenfalls, man wird mit dem Gegengift nicht rechtzeitig fertig -- zumal wohl noch einige andere ähnlich seltene Zutaten benötigt werden. Natürlich konnten sich auch die anwesenden Forther (insbesondere Wolfgang Allinger) nicht zurückhalten,
eigene Witze beizusteuern.

Die für den Abend angesagte Diskussion “Force of Forth” lief wegen Übermüdung einiger Teilnehmer weitgehend ergebnislos aus.

Freitag vormittag, bei wunderschönem Wetter (wie bestellt) bestiegen die meisten der schon anwesenden Forther den Kofel. Der Berg ist mit etwa 500 Metern Höhenunterschied leicht zu bezwingen, die felsigen letzten paar Meter verlangen aber etwas Schwindelfreiheit. Oben konnte man die herrliche Aussicht genießen.

Am Freitag nachmittag ging es dann mit den Vorträgen los.

Jens Wilke
präsentierte den PSC1000, ein Forth-Prozessor, der auf den ShBoom von Chuck Moore zurückgeht, und inzwischen von der Patriot Scientific Corporation weiterentwickelt wird. Die verkaufen den Prozessor als “Java-Processor”, obwohl der Prozessor nur etwa 20% der JavaVM-Befehle direkt ausführen kann. Da sieht man, daß die Ähnlichkeit von JavaVM und Forth nur auf die grundlegenden Befehle beschränkt ist. Jedenfalls kann man auf dem PSC1000 auch ein richtiges Forth implementieren. Allerdings ist das erstaunlicherweise schwieriger als ein threaded-code Forth auf irgendeinem beliebigen Prozessor, da Jens erst mal dem Cross-Compiler von Gforth beibringen mußte, wie man Bytecode erzeugt.

Ewald Rieger
zeigte seine erste MINOS-Applikation, eine Visualisierung eines Gaschromatographens. Angezeigt werden ein Spektrum, eine Tabelle der verwendeten Stoffe, Regeln, nachdem die Stoffe farblich gekennzeichnet werden, und eine Voransicht der Spektren, in denen man Spektrum und Stoff mit der Maus anwählen kann. Die Anwendung läuft zur Beruhigung des Chefs auch unter Windows NT, und darf deshalb ohne Probleme weiter unter Linux entwickelt und verwendet werden.

Am Abend war eigentlich ein IRC-Chat (über Egmont Woitzels Handy) geplant. Da aber im IRC niemand auf dem angekündigten Channel war, und ohnehin eine Fackelwanderung auf dem Programm stand, fiel das flach.

Klaus Kohl
leitete die Vorträge am Samstag mit einer Präsentation mehrerer Microkontroller ein, die alle natürlich auch Forth können. Der unmittelbare Bezug zu Forth wurde aber nicht hergestellt.

Dieter Peter
ging in dieselbe Richtung mit einem Vortrag über die neue PICmikro-Architektur, die 18er PICs. Die haben nun ein paar neue Features, die es leichter machen, ein Forth darauf zu implementieren; so ist der Hardware-Returnstack erstmals zugreifbar.

Egmont Woitzel
brachte mit ,,Eine Forth-Machine für Hitachie H8-300", den obligatorischen NEXT-Vortrag. Benchmarks gab's nicht nur zu NEXT-Varianten, sondern insbesondere zur Anzahl gecacheter Stackelemente (TOS, TOS&NOS, keins). Dabei zeigte sich, daß nur TOS zumindest beim H8 das Optimum darstellt.

Bernd Paysan
stellte nach der Kaffeepause seine 3D-Turtle-Grafik alias “Dragon Graphics” anhand zweier Beispiele vor. Anders als 2D-Turtles werden hier richtige Körper definiert, indem man ausgehend von der Turtle Schnitte durch den darzustellenden Körper beschreibt. Als noch amtierender Drachenträger hatte er den Swap-Drachen als 3D-Animation implementiert, mit Wellenbewegungen im Schwanz und flatternden Flügeln. Als zweites Beispiel diente ein typisches Turtle-Graphics-Objekt, ein Baum. Nur daß hier Stamm und Äste dreidimensional dargestellt wurden. Die 3D-Turtle-Grafik verwendet OpenGL als Basis.

Alexander Burger
präsentierte mit “Tea Time” seine neuste Version von Lifo, eine Mischung aus Forth (Syntax), Lisp (Strukturen) und Java (Objekte).
Auch hier stand eine Anwendung im Vordergrund, ein Bestellmanagement.

Tea Time bietet die Datenbank, Java die Benutzeroberfläche (Swing), die von Tea Time wieder abstrahiert wird.

Louis Schmitt
kam nach dem Mittagessen mit seiner Version vom Forth-GUI zum Zuge. Basis ist hier ein 16-Bit-Forth für den PC, Turbo-Forth, und eine DOS-Grafik-Schnittstelle (*Fastgraf, die red.*). Die einzelnen Komponenten sind als Overlays realisiert, werden also nacheinander von der Platte geladen (oder aus dem Cache kopiert). Trotz aller Unterschiede sehen die einzelnen Zeilen für ein GUI-Objekt irgendwie Tea Time und MINOS ähnlich, wenn auch hier kein Layout-Manager wirkt (also absolute Koordinaten eingesetzt werden müssen). Louis setzt das System an einer Ingenieur-Schule als Lehrsystem ein, und läßt Maschinenbauingenieure damit in zwei Semestern in die Informatik hineinschnuppern. Als wichtigste Lehre kann man hier ziehen, daß es offensichtlich nicht schwer ist, Forth zumindest soweit zu lernen, daß man etwas damit anfangen kann, und daß der motivierende Feedback vom Computer sehr schnell kommt. Voraussetzung: Eine bereits vorhandene Library von Komponenten.

Johannes Reilhofer
zeigte nach der Kaffeepause, daß in Sachen Vortragstechnik noch eine Steigerung möglich ist. Obwohl sein Vortrag das Thema Forth schnell verließ (“Wir machen das alles in Forth, und das ist das letzte Mal, daß Sie etwas von Forth hören”), wurden die Einzelheiten zum Thema “Dynamischer Getriebeversuch” sehr anschaulich und interessant präsentiert.

Hauptidee ist eine Spektralanalyse der Getriebeschwingungen, und ein Vergleich mit Daten, die das Programm zu Beginn des Versuchs gelernt hat (da war das Getriebe noch ganz). Auch aus den Frequenzen der Störsignale kann man bestimmen, welches Zahnrad wohl defekt ist.

Klaus Schleisiek
stellte einen neuen Forth-Prozessor für FPGAs vor -- allerdings noch ein Papiertiger. Der Prozessor hat zwar Ähnlichkeiten mit Chuck Moores neueren Chips, verwendet aber die Transputertechnik, um ein Literal zusammenzubauen: Jeder Literal-Befehl schiebt einige Bits in das Literal-Register hinein. Klaus will den Prozessor unter einer Art OpenSource-Lizenz entwickeln lassen, hat aber bereits ein Patent angemeldet und würde am liebsten die ganze Versammlung ein NDA unterscheiben lassen. Das mit der “gift culture” hat er offenbar nicht richtig verstanden. Auch das Literal-Register stößt auf Kritik, man solle Literals doch besser auf dem TOS zusammenbasteln.

Nach dem Abendessen war noch ein Workshop zu diesem Prozessor geplant. Gerade als Klaus beginnen wollte, stürmten die Begleiterinnen der Tagungsteilnehmer den Saal und wollten den Swap-Drachen sehen. Es bildete sich spontan ein zweiter Workshop “wir verbessern den Swap-Drachen”, der den Tagungsraum und den LCD-Projektor in Beschlag nahm. Der Forth-Prozessor-Workshop verzog sich in ein anderes Zimmer.

Den Swap-Drachen selbst (den aus Bronze) bekam dieses Jahr Fred Behringer für seine Arbeit für die VD -- schließlich rezensiert er nicht nur eine fremdsprachige Forth-Zeitschrift, sondern gleich deren drei.

Das Foto, das Heinz Schnitter auf der Forml gemacht hatte, wurde Johannes Reilhofer für seinen besonders lebendigen Vortrag verliehen.

Als Resumé der Tagung kann man sagen, daß etwa die Hälfte der Vortragenden tatsächlich Programme in Forth schreiben -- und daß diese Programme durchaus zeitgemäß aussehen. Die andere Hälfte portiert nach wie vor Forth-Systeme auf irgendwelche obskure Prozessoren, oder entwickelt gar eigene solche (zumindest auf dem Papier). Obwohl der Standard ja in die zweite Runde geht, hat niemand einen Vortrag darüber gehalten -- das mag auch daran liegen, daß Ullrich Hoffmann nicht zur Tagung gekommen ist.

Bernd Paysan

Fred Behringer - beherbergt den SWAP in diesem Jahr in München.

Wir gratulieren !

Wieviel Windows braucht der Mensch ?
Früchte zählen ‚klassisch'

Friederich Prinz
Hombergerstraße 335
47443 Moers
Friederich.Prinz@T-Online.DE

Der nachfolgende Aufsatz ist in einer englischen Übersetzung
von Alan Wenham erstmals erschienen in der Forthwrite, Ausg. 100

Stichworte

Windows; OOP; Datenstrukturen, CREATE...DOES>, Vererbung, verkettete Listen

Vorwort

Objektorientiertes Programmieren kann auch dem Forth-Programmierer das Arbeiten ganz wesentlich erleichtern. Das hat Egmont Woitzel [1] in seiner Artikelserie zur objektorientierten Programmierung mit comForth 4 aufgezeigt. Selbst ein ‚Mini-OOF', dessen Möglichkeiten Chris Jakeman [2] beschreibt, ist dazu geeignet, einzelne Definitionen effizienter zu erstellen, zu testen und immer wieder in Ableitungen zu verwenden.

Die Fähigkeiten zu nutzen, die das System quasi ‚fertig' anbietet, bedeutet für den Programmierer, daß er alle seine Fertigkeiten auf die Lösung seiner eigentlichen Aufgabe fokussieren kann. Das COM (Component Object Model), und mittlerweile auch das DCOM (D = distributed) bietet eine Vielzahl von Schnittstellen zu ‚Funktionalitäten', die gerade von kommerziellen Programmierern einfach ‚nur genutzt' werden. Es spielt nur eine untergeordnete Rolle, welches Objekt-Modell genutzt wird – die Beschränkung auf die bloße Nutzung der Systemangebote bleibt – leider – das tragende Element jeglicher Programmierung in einer objektorientierten Umgebung.

Dabei ist der Begriff der Beschränkung durchaus negativ zu sehen. Die, nach Auffassung des Autors, viel zu hoch gelobte Arbeitsteilung zwischen System- und Applikationsprogrammierern führt neben den offensichtlichen Vorteilen auch zu verfahrensbedingten, einschneidenden Nachteilen. So machen sich die wenigsten Menschen wirklich deutlich, daß die Übernahme der fertigen Arbeit eines Anderen automatisch auch die Übernahme der Verantwortung für die Fehler in der Arbeit des Anderen bedeutet. In vielen Aufgabenstellungen mag es heute trotz dieses Nachteils ‚gar nicht anders gehen'. Das gilt ganz sicher für Applikationen, die in der Welt der bunten Fenster, des Drag&Drop und anderer ‚visueller Features' arbeiten sollen. So bald aber diese besonderen Anforderungen nicht im Vordergrund einer Applikation stehen, erscheint die Frage angebracht, wieviel Windows man den wirklich braucht. Sehr oft lassen sich einfachere und angemessenere Lösungen finden.

Vorarbeiten

1992 habe ich, gemeinsam mit Michael Major, in einer Projektarbeit ‚Fenster' entwickelt, die über alle wesentlichen Eigenschaften der Fenster von WINDOWS verfügten. Die Dateninhalte der Fensterobjekte waren gegen reguläre Zugriffe geschützt, weil sie als ‚interne Bestandteile' der Objekte definiert wurden. Neue Fenster konnten aus bestehenden Fenstern abgeleitet werden und dabei alle Eigenschaften (Daten) des Vorgängers erben. Jedes Fenster kannte sowohl seine Vorgänger als auch seine Nachfolger. Der Anwender konnte über den Stack und den Eingabestrom mit den Fenstern kommunizieren. Alle Fensterobjekte haben auf den gleichen Code zugegriffen. Redundante Definitionen waren nicht notwendig – auch nicht innerhalb der Objekte selbst.

Die Fensterobjekte haben den Eingabestrom an einen separaten Parser übergeben. Dieser hat, entsprechend den , Anweisungen' im Eingabestrom, Funktionen wie "SetzeGroesse" aufgerufen. Auf Polymorphismus haben wir damals ebenso wenig Wert gelegt, wie auf das ‚Late-Bindung' oder die Möglichkeit, einmal definierte und getestete Codes mehrfach zu nutzen. Das war für eine solche, relativ einfache Applikation nicht notwendig. All diese Dinge ließen sich aber relativ problemlos und ohne großen Aufwand mit einem ‚klassischen' Forth nachbilden.

Die Objekte selbst waren ‚Ableitungen' eines CREATE ... DOES> Konstruktes. CREATE ... DOES> taucht in der Literatur stets als Werkzeug zur Definition von Datenstrukturen auf. Tatsächlich waren die von uns definierten Fensterobjekte nur wenig mehr. Aber auch ‚Windows-Objekte' sind wenig mehr als Datenstrukturen. Man muß sich dies nur verdeutlichen, um zu sehen, daß Forth uns ‚schon lange vor Windows' alle Instrumente an die Hand gegeben hat, die zum ‚Handling' modernster Anforderungen an Software notwendig sind. Wenn man sich dies aber erst einmal bewußt gemacht hat, gibt es keinen Grund, Forth nicht selbst arbeiten zu lassen.

Wiederholung: CREATE ... DOES>

Den versierten Forthern sind die Zusammenhänge um CREATE...DOES> natürlich bekannt. Weil es mich aber immer wieder wundert, daß dieses Werkzeug so selten genutzt wird, fasse ich die wesentlichsten Aussagen zu diesem Konstrukt noch einmal kurz zusammen.

Das einfachste ‚Objekt', das jedes Forth quasi ‚von Haus aus' kennt, ist in meinen Augen die CONSTANT. Hier sind unter einem Namen Daten und Codes zusammengefaßt. Der Code von CONSTANT sagt, was mit den Daten der CONSTANT zu geschehen hat. CONSTANT ist in jedem Forth ein wenig anders implementiert. Seine Definition folgt aber im Prinzip in allen Implementierungen diesem Beispiel:
Das ‚Verfahrensprinzip' hierbei ist die Trennung zwischen Compile- und Laufzeit innerhalb der Definitionsarbeit, die

```
: CONSTANT ( n -<name>- )
    CREATE      \ erzeuge das Wort <name>
    ,           \ kompiliere den TOS
    DOES> @     \ gib den kompilierten Dateninhalt zur
                \ Laufzeit auf den TOS
;
```

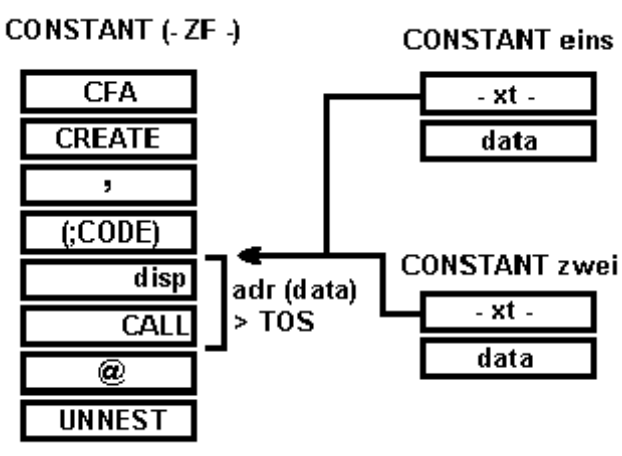
der Programmierer vornimmt. Bis zum DOES> definiert der Programmierer für die Compilezeit eines Wortes. Deutlicher: definiert wird nicht für die Compilezeit von CONSTANT, sondern für die Compilezeit der Worte, die mit CONSTANT definiert werden sollen !

Nach dem DOES> folgen die Statements, die für die Laufzeit der Definition verantwortlich sind. Auch hier gilt: Laufzeit meint die Laufzeit der Worte, die mit CONSTANT definiert werden.

Das CREATE legt lediglich einen Namen im Wörterbuch des Forth an. Auf das CREATE können (fast) beliebige Definitionen folgen. Das gilt für die Kompilation von Datenstrukturen ebenso wie für Anweisungen, die ‚sofort' – also noch während der Compilezeit – mit den Daten arbeiten sollen.

Das DOES> beschreibt den Start des Codes, der mit den zuvor angelegten Daten zur Laufzeit des Wortes verknüpft sein soll. Zur Laufzeit legt DOES> die Adresse der Datenstruktur auf den TOS. Die nachfolgenden Definitionen können deren Inhalte beliebig manipulieren.

In dem sehr einfachen Beispiel von CONSTANT braucht die damit erzeugte Konstante nur noch den Inhalt, der ihr angebotenen, eigenen Datenadresse auf den TOS zu ‚fetchen'.
Bemerkenswert ist hierbei, daß ein so erzeugtes Wort de facto über keinen eigenen Code verfügt. Der nach dem DOES> definierte Code liegt ausschließlich in dem Wort CONSTANT. Jede mit diesem Wort erzeugte Konstante verfügt lediglich über zwei 16-Bit Zellen. In der ersten Zelle ist ein Zeiger auf den Code hinter dem DOES> in CONSTANT hinterlegt ! Die zweite Zelle enthält die Daten der Konstanten. Das macht so definierte Worte angenehm klein !



Die Dateninhalte in solchen Strukturen sind gegen Zugriffe , von außen' geschützt. Regulär kann nur das neu definierte Wort (Die CONSTANT MeineKonstante) auf diese Dateninhalte zugreifen. Natürlich, das zeigt die Diskussion um VALUES und seine Operatoren, ist es dem Forther trotzdem jederzeit möglich, diese Inhalte zu manipulieren. Die Daten liegen im Speicher. Sie haben eine Adresse, die sich zumindest als Offset auf die CFA der CONSTANT beschreiben läßt. Die CFA ist jederzeit über das Wörterbuch des Systems abfragbar. Forther kennen ‚ihr' System, das ihnen meist sehr viel mehr als nur ‚Sprache' ist, in der Regel exakt genug, um solche Manipulationen jederzeit zu realisieren.

Aber – gilt das nicht ebenso für alle Objekte ‚unter Windows' ? Auch diese Objekte liegen im Speicher, haben eine eigene, bekannte Adresse – und sind somit manipulierbar. Alles was man zu solchen Manipulationen benötigt, ist die Kenntnis der Adresse und der darunter definierten Strukturen.

Datenstrukturen

Die jeweils am einfachsten zu handhabenden Strukturen sind immer diejenigen, die man selbst definiert hat. Zum Glück lassen CREATE ... DOES> Konstrukte die Definition von beliebigen Datenstrukturen zu. Das Grundgerüst für die be-

```
: ErzeugeFenster ( - name - )
    CREATE
    <was immer Du brauchst> ALLOT
    DOES> DUP >Parser
          DROP
;
```

reits angesprochenen Fenster ist nur wenig komplexer:
‚was immer Du brauchst' könnte die linke, obere Position des Fensters sein, seine Breite und Höhe, Farbe und jede denkbare, andere Information. Die Breite der einzelnen Daten – Byte, Wort, DWort... – richtet sich in meinen Applikationen stets an den jeweiligen Notwendigkeiten aus.
Das DUP vor dem Aufruf des Parsers sorgt dafür, daß die Adresse der Datenstruktur auch nach der Arbeit des Parsers zur Verfügung steht. Hier wird sie allerdings ‚gedropt', weil zunächst keine weitere Verwendung vorgesehen ist.

Können Sie sehen, welche Möglichkeiten sich hier eröffnen ?

```
DEFER >Parser      \ ...am besten: gleich austauschbar


: _Parser1 ( adr — )
    BL WORD COUNT    \ Eingabestrom nach HERE
                     \ auswerten ...
    2DUP “Aktion1” SEARCH IF Aktion1
                          THEN
                     \ ... und entsprechende Worte
  ...                \ aufrufen...
    2DROP
;

, _Parser1 IS >Parser
```

Stellen Sie sich Folgendes vor:
Mit einer solchen, noch sehr einfachen Definition läßt sich ein großer Teil aller Aufgaben, die von Fenstern erledigt werden sollen, flexibilisieren. Die Einfachheit ist sicher vielen Aufgabenstellungen angemessen.

Früchte zählen, die Aufgabe

Egmont Woitzel hat in seiner Artikelserie dazu aufgefordert, das Zählen von Früchten ‚mit anderen Forth-Systemen‘ darzustellen. Dieser Aufforderung soll der folgenden Ansatz nachkommen. Zur Verdeutlichung soll die – selbstgewählte – Aufgabenstellung beschrieben werden.

- EIN Wort definiert beliebige Früchte
- Jeder Frucht können beliebige Eigenschaften zugeordnet werden
- Einzelne Früchte können auch während eines Zählvorganges hinzu definiert werden
- Früchte definieren ihre Eigenschaften selbst (‚Ableitung...‘)
- Die Eigenschaften können auch während einer Zählung um weitere Eigenschaften ergänzt werden
- NEUE Früchte können WAHLWEISE die Eigenschaften beliebiger, zuvor definierter Früchte erben
- Zählungen werden primär auf die Eigenschaften vorgenommen
- Zählungen auf eine Eigenschaft werden gleichzeitig auf die zugehörige Frucht ausgeweitet
- Eine AUSKUNFT über die ANZAHL einzelner Früchte und Eigenschaften ist jederzeit möglich

Vorüberlegungen

Selbstverständlich soll die Lösung auf CREATE...DOES> aufbauen. Nach dem DOES> können beliebige Codes definiert sein – natürlich auch ein weiteres CREATE...DOES>.

```
: NeueFrucht ( -<name>-)
     CREATE
     <Datenstruktur>
     <Codes>
     DOES> ( -<name>-) CREATE
                  <Datenstruktur>
                  <Codes>
                  DOES>
                  <Codes>
;
```

Folglich läßt sich die Lösung mit dem groben Rahmen beschreiben:

Die Arbeit sähe dann etwa so aus:

```
NeueFrucht  Aepfel
            Aepfel rot
            Aepfel gruen
NeueFrucht  Bananen
            Bananen gruen
            Bananen gelb
            Aepfel reif
            ...
```

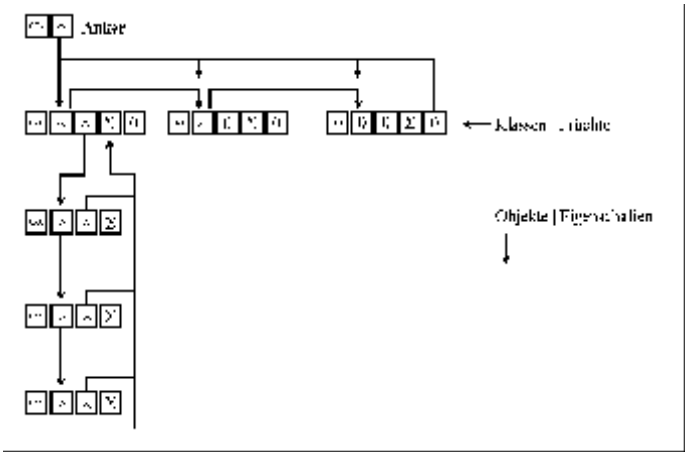
Die Forderung nach jederzeit hinzu definierbaren Früchten und Eigenschaften ruft nach einer ‚dynamisierten‘ Lösung. Wir wissen, daß Forth jedes neue Wort dort im freien Speicher ablegt, wo ab HERE gerade Platz ist. Ein Arbeitsgang des Zählens von Früchten wird in natürlicher Form kaum dergestalt ablaufen, daß der Zähler zunächst eine Frucht und deren Eigenschaften definiert und anschließend die nächste Frucht und deren Eigenschaften. Vielmehr ist anzunehmen, daß Früchte und Eigenschaften im Wörterbuch ungeordnet definiert sein werden.

Wenn eine Frucht ihre Eigenschaften selbst ableiten soll, dann muß sie ‚wissen‘, wo ihre bisherigen Eigenschaften im Wörterbuch aufzufinden sind und wie diese heißen. Und selbstverständlich müssen Eigenschaften exakt ‚wissen‘, zu welcher Frucht sie gehören. Anders ließe sich die Addition einer Eigenschaftenzählung nicht auf die zugehörige Frucht ausweiten.

Solche Beziehungen lassen sich am einfachsten mit einer einfach verketteten Liste darstellen. Es ist leicht vorstellbar, daß eine Frucht in ihren Daten die Adressen aller ihrer Eigenschaften festhält. Wenn dies in der Art einer Tabelle geschieht, dann muß diese entweder sehr groß angelegt sein, oder an einem – meist recht frühen – Zeitpunkt ist es nicht mehr möglich, eine weitere Eigenschaft hinzu zu fügen. Wenn aber die Frucht nur die Adresse ihrer ersten Eigenschaft festhält, und diese Eigenschaft wieder die Adresse der nächsten Eigenschaft usw., dann kann diese Liste aus Eigenschaften so lange wachsen ‚bis der Speicher platzt‘.

Die Früchte werden im Wörterbuch nichts anderes sein, als

alle anderen Worte des Systems auch. Wenn die Aufgabenstellung verlangt, alle Früchte und die bisherige Zählarbeit aufzuzeigen, dann müssen die Früchte eindeutig von den , anderen' Worten im System unterscheidbar sein. Bevor man an dieser Stelle darüber nachzudenken beginnt, wie sich dies über die Namensgebung realisieren ließe, sollte man in die Datenstruktur der Früchte einen weiteren Zeiger einplanen; einen Zeiger auf die jeweils nächste Frucht. Die dabei zu-



nächst in den Gedanken entstehende Liste nimmt sofort eine endgültige Form an, wenn man ein drittes Datenfeld dazugibt: die Summe.
Für die Aufgabe des Früchtezählens habe ich die Struktur der Abbildung gewählt. Ausgehend von einer ‚Ankerzelle', die entweder eine NULL enthält – wenn noch keine Frucht definiert wurde – oder die Adresse der ersten Frucht, wächst die Früchteliste dynamisch und ‚nach Bedarf'. In der jeweils ersten Datenzelle einer Frucht wird die Adresse der nächsten Frucht abgelegt, oder eine NULL, wenn es noch keine nächste Frucht gibt. Die zweite Datenzelle der Frucht enthält die Adresse der ersten Eigenschaft dieser Frucht. In der dritten Datenzelle legen die Eigenschaften die Zählwerte ab. Die vierte Zelle wird schließlich temporär benötigt, ausschließlich während der Compilezeit, um den noch zu beschreibenden Vererbungsmechanismus zu steuern.

Der ausschließliche Gebrauch dieser vierten Zelle während der Compilezeit, macht es denkbar, die Summenzelle, die erst zur Laufzeit benötigt wird, mit einer doppelten Funktion zu belegen und auf die vierte Zelle zu verzichten. Darauf habe ich hier verzichtet, weil mir im Augenblick die Übersichtlichkeit der Struktur und des zugehörigen Quellcodes wichtiger ist als die hier verschwendeten Bytes.

Die Eigenschaften benötigen in jedem Fall nur drei Zellen. Wie bei den Früchten, "zeigt" auch hier die erste Zelle auf die jeweils nachfolgende Eigenschaft; sofern eine solche definiert wurde. Anderenfalls enthält diese Zelle eine NULL. Die zweite Zelle enthält die Adresse der Summenzelle der Frucht! Damit ist es der Eigenschaft sehr bequem möglich, den ihr übergebenen Zählwert nicht nur auf den Inhalt der eigenen Summenzelle zu addieren, sondern auch gleich auf den Inhalt der Fruchtzelle.

Die Realisierung

Die abgebildete Struktur sieht schön ordentlich und leicht verständlich aus. Ebenso leicht ist sie auch zu implementieren. Wie ich zuvor beschrieben habe, lassen sich sowohl vor als auch nach dem DOES> Codes ‚ablegen' – mit dem Unterschied, daß die einen zur Compilezeit und die anderen zur Laufzeit der Definition arbeiten. VOR dem DOES> lassen

```
: NeueFrucht
   CREATE
   <verankere mich im Vorgänger>
   0 , 0 , 0 ,
   <Adresse von Erblasser oder NULL>
   ,
   DOES> CREATE
        <verankere mich im Vorgänger>
        0 , 0 , 0 .
        DOES>  < wenn ein Zählwert auf dem TOS
                 liegt>
               < dann Zählwert auf eigene Summe
                 addieren>
               < Zählwert auf Früchtesumme
                 addieren> ;
```

sich also alle Verknüpfungen in den gerade erst angelegten Datenbereichen vornehmen, die zur Erstellung und zur Dynamisierung der Liste notwendig sind.

Mit Hilfe dieses Gerüstes läßt sich alles Wesentliche der eigentlichen Aufgabe definieren. Eine Frucht, die neu angelegt werden soll, hangelt sich ab der Ankerzelle solange durch die Liste, bis sie in dem entsprechenden Zeigerfeld der zuletzt definierten Frucht eine NULL findet. Dort trägt sich die neue Frucht ein. Fertig. Ganz ähnlich verfahren Eigenschaften. Sie müssen lediglich an der richtigen Stelle - in ihrer Frucht (!) - den Abzweig in den Teil-Ast der Eigenschaften ihrer Frucht finden. Auch das macht dem Forth-Programmierer keine Kopfschmerzen. Jedes "WORDS" tut nichts anderes !

Eindeutige Namensgebung

Es bleibt ein kleines, aber nicht triviales Problem ! Alle Früchte und Eigenschaften sind ‚ganz normale' Forthworte. Die Eigenschaft GELB kann nicht gleichzeitig für Äpfel, Birnen und Kirschen existieren ! Eine eindeutige Namensgebung ist Pflicht ! Ich habe das Problem für mich wie folgt gelöst:

Zunächst macht schon das Wort NeueFrucht etwas Ungewohntes. Der Aufruf NeueFrucht Aepfel legt nicht etwa das erwartete Wort AEPFEL im Wörterbuch an, sondern das Wort DEF.AEPFEL. Den Prefix "DEF." habe ich in Anlehnung an das Wort DEFINE gewählt. Mit dem Wort DEF. AEPFEL sollen schließlich die Eigenschaften der Aepfel definiert werden.

Wieviel Windows braucht der Mensch ?

Dementsprechend erzeugt der Aufruf DEF.AEPFEL gelb auch nicht das Wort GELB, sondern AEPFEL.GELB. DEF. BIRNEN gelb würde natürlich BIRNEN.GELB erzeugen. Ich habe das Problem der Namensgebung mit minimalen , Erweiterungen‘ von CREATE umgangen, bzw. mit kleinen Manipulationen des Eingabestromes. Selbstverständlich erledigen die Worte diese Arbeit ‚nebenher‘ und automatisch.

Die Auswertung des Eingabestromes ist auch die Basis für den Vererbungsmechanismus. Der Aufruf NeueFrucht Kirschen DEF.AEPFEL erzeugt das Wort DEF.KIRSCHEN. Noch vor seinem Abschluß wertet DEF.KIRSCHEN den Eingabestrom aus und erzeugt sich selbst eine Liste aller Eigenschaften, über die DEF.AEPFEL zu diesem Zeitpunkt verfügt. Selbstverständlich werden diese Eigenschaften dabei mit dem Prefix “KIRSCHEN.” versehen.

Zur Kontrolle der Definitionsarbeit – beim Anlegen neuer Früchte und Eigenschaften – habe ich der ‚Applikation‘ das Wort “.INFO” mitgegeben. Auch dieses Wort hangelt sich ab dem Anker durch die Liste und ihre Teil-Äste. Dabei gibt . INFO die Namen aller Früchte, deren Zählerstände, alle zugehörigen Eigenschaften und deren Zählerstände aus. Und selbstverständlich werden die Namen dabei ‚automatisch‘ von ihrem jeweiligen Prefix ‚befreit‘.

```
.INFO
  AEPFEL      12
    GELB       4
    ROT        3
    GROSS      3
    KLEIN      2
  BIRNEN      30
    REIF      20
    UNREIF    10
  KIRSCHEN    10
    ROT       10
```

Ich betrachte die Aufgabe damit als gelöst.

Wichtiger als das Lösen dieser Aufgabe ist mir aber ein nachdrückliches Plädoyer für Forth und vor allem für CREATE... DOES>, bzw. für eine angemessene EINFACHHEIT. Ich stelle die Frage noch einmal. **Wieviel Windows braucht man eigentlich** ? Zum ‘Früchtezählen’ braucht man Windows und seine vorgefertigten Klassen und Objekte und deren Mechanismen ganz sicher nicht ! Polymorphismus wird hier ebenso wenig benötigt wie Late-Bindung. Beides ließe sich realisieren, wie Chris Jakeman zeigen wird. Und die Dateninhalte der ‚Klassen‘ und ‚Objekte‘ sind hier gegen reguläre Zugriffe durchaus geschützt. Wenn Sie die Daten eines EigenschaftObjektes mutwillig ‚von außen‘ verändern, dann sind dafür ebenso selbst verantwortlich, wie sie es für die Schraube sind, die sind in ein Getriebe werfen !

‚Klassisches Forth‘, ein wenig Vordenken – Nachdenken kommt zu spät und macht Arbeit – und eine einfache, nachvollziehbare und in jedem Teil durchschaubare (!) Lösung ist in vielen Fällen angebrachter, als sich auf die Arbeit Anderer zu verlassen !

Friederich Prinz

[1] Egmont Woitzel, OOP in comForth 4,
VD 3 und 4 in 1998, sowie VD 1 in 1999

[2] Chris Jakeman, OO-Forth, klein und fein,
VD 2 in 1999
Original: Forthwrite, Issue 99, Nov. ‘98


```
\ 2. Ansatz - [ Nach Diskussion mit Michael ] fep - 04/12.98

  EMPTY

\ --- Orientierungsworte ----------------------------------------------

  COMMENT:
```

Ich 'vergeude' Laufzeit für den Vorteil der besseren Lesbarkeit meiner Codes. Dazu habe ich die nachstehenden 'Orientierungsworte' eingeführt.
Die Struktur 'in' der Ankerliste geht auf Einwände von Michael Major zurück, dessen Aufmerksamkeit mich bei der Diskussion meines ersten Ansatzes vor einer größeren Menge unnötiger Arbeit bewahrt hat.

Der 'Anwender' arbeitet mit den bereitgestellten Worten wie folgt:

NeueFrucht <Name> erzeugt eine neue 'Fruchtklasse'. Ohne Dazutun des Anwenders wird der Fruchtname um den Prefix "DEF." erweitert, z.B:

```
NeueFrucht Aepfel ==> DEF.AEPFEL
```

Die so definierten Früchte werden um Eigenschaften ergänzt:

```
DEF.AEPFEL gross ==>  AEPFEL.GROSS
```

Auch hier wird die Zeichenkette des zu erzeugenden Wortes ohne das Dazutun des Anwenders verändert. Der Prefix "DEF." wird entfernt, und der 'reine' Fruchtname wird um einen Punkt und den Namen der Eigenschaft erweitert.
Der Aufruf der 'Klasse' ohne nachfolgende Eigenschaft hat keine Auswirkungen.

Diesen 'SubClasses' können Werte übergeben werden:

```
3 AEPFEL.GROSS
```

Das Wort AEPFEL.GROSS addiert den Wert auf dem Stack auf den Inhalt seiner eigenen Summe, sowie auf die Gesamtsumme der Frucht 'AEPFEL'.
Der Aufruf von AEPFEL.GROSS ohne einen Parameter auf dem Stack, hat keine Auswirkungen.

Neue Früchte können - auf Wunsch - die Eigenschaften einer bereits definierten Frucht 'erben'.

```
NeueFrucht Birnen def.aepfel
```

erzeugt die Klasse DEF.BIRNEN, die alle Eigenschaften bekommt, die in der Klasse DEF.AEPFEL zu dieser Zeit vorhanden sind.

```
  COMMENT;

  : >CFA     ( adr -- CFA )  BODY> ; \
  : >SuccPtr ( CFA -- adr )  2+ ;       \ zum :   Zeiger auf
                                        \         Nachfolger
```

```
  : >EignPtr ( CFA -- adr )  4 + ;       \          Zeiger auf
                                         \          Eigenschaften
  : >Summe( CFA -- adr )  6 + ;          \          Summenfeld
  : >PrntPtr  ( CFA -- adr )  8 + ;      \          Zeiger auf
                                         \          Erblasser (Data)
  : >SummPtr ( CFA -- adr )  4 + ; \     Zeiger auf
                                   \     Summenfeld (Klasse)

\ ------------------------------------------------------------------------

  Variable Anker            \ Start der Liste mit Klassen- und
       0 Anker !            \ Eigenschaften / Definitionen

\ --- Werkzeuge ----------------------------------------------------------

  : GetMyCFA ( -- CFA )\ CFA der jüngsten Definition
     LAST @ NAME> ;

  : @succ^ ( # adr -- # adr' )     \ # ist 'dummy'
     NIP DUP @ DUP ;

  : Me>Prev! ( Ptr # Ptr' -- )     \ # ist 'dummy'
     BEGIN @succ^
     0= UNTIL DROP ! ;

  : Add$ ( $1 Cnt1 $2 Cnt2 -- $1+$2 )
     2SWAP
     DUP 3 PICK + PAD C!       \ CountSumme gespeichert
     PAD 1+ SWAP 2DUP + >R CMOVE
                               \ 1. String ab PAD+1 abgelegt
     R> SWAP CMOVE             \ 2. String ab PAD+1+Cnt1
                               \                    abgelegt
     PAD ;

  : AbortCreate ( -- )
     FIND IF ABORT" bereits definiert !"
       ELSE COUNT 1- + C@ ASCII . =   \ Eigenschaften-
                                      \ name
                                      \ nicht angegeben
            IF ABORT
       ELSE HERE "CREATE
       THEN
     THEN ;

  : Create+Prefix ( $1 Cnt1 -<name>- )  \ dem neuen Wort
                                        \ wird
                                        \ ein beliebiger
     BL WORD COUNT Add$                 \ String als Prefix
                                        \ vorangesetzt
     COUNT 1+ >R               \ Neuen Namen nach HERE
                               \ kopieren
     1- HERE R> CMOVE          \ und auf "CREATE vorbereiten
     HERE SUFIX.BL
     ?UPPERCASE
     AbortCreate ;

  : GetClass$ ( xt -- $adr cnt )  \ Prefix aus Klassennamen
                                  \ bilden
     >NAME DUP 1+ SWAP
     YC@ 31 AND                \ 'korrigierter' Count
     DUP PAD 100 + C!          \ Count sichern
     YSEG @ -ROT
     ?CS: SWAP PAD 101 + SWAP
     CMOVEL                    \ Gesamtnamen aus YSEG
                               \ kopiert
     PAD 100 + DUP C@ + DUP C@
                               \ letztes Zeichen muss noch
     127 AND SWAP C!           \ korrigiert werden (siehe TJZ)
     PAD 100 + COUNT
     ASCII . SCAN 1-           \ Prefix bis "." abschneiden
     SWAP TUCK C! COUNT ;

  : DefSub$ ( CFA -- adr cnt )       \ "<ClassName>."
     GetClass$ " ." Add$ Count ;

  : Parent! ( -- )
     DEFINED                   \ Parent im Eingabestrom
                               \ genannt ?
     DUP 0= SWAP 1 = OR        \ 0 = undefinierter Parent
     IF DROP 0 ,               \ 1 = leerer TIB
    ELSE 0 Anker
       BEGIN @succ^ >CFA       \ Parent nur sichern,
           3 PICK = IF 2DROP , EXIT
                THEN           \ wenn in Ankerliste
           DUP 0= IF 2DROP DROP 0 , EXIT
                THEN           \ enthalten
       AGAIN
    THEN ;

  : $>Here ( adr cnt -- adr )
     1+ SWAP 1- SWAP HERE SWAP CMOVE
     HERE ;

  : $>Tib ( adr )
    COUNT >R TIB R@ CMOVE R@ SPAN ! R> #TIB ! >IN
OFF ;

  : ExecuteMe ( -- )
     " DEF." GetMyCFA GetClass$ Add$ $>Tib ;

\ --- zur Kontrolle ... -------------------------------------------------

  VARIABLE Info
       Info OFF

  : .Info ( -- )       \ Ausgabe der Früchte und ihrer Zählsummen,
                       \sowie
    DARK CR            \ Ausgabe der Eigenschaften und ihrer
                       \ Zählsummen
    0 Anker BEGIN @succ^
          0<> WHILE
          >CFA DUP GetClass$ TYPE 20 #OUT !
              DUP >Summe @ 5 .R CR
              DUP >EignPtr 0 SWAP
                 BEGIN @succ^ 0<>
                   WHILE 2 SPACES
                   >CFA DUP GetClass$ TYPE 18 #OUT !
                       DUP >Summe @ 5 .R CR
                   >SuccPtr
                  REPEAT 2DROP
            >SuccPtr
          REPEAT 2DROP ;

\ --- Neuefrüchte -------------------------------------------------------

  : Inherit ( CFA -- )
    DUP >PrntPtr @ DUP 0<>\ CFA.selbst; CFA.parent
    IF >EignPtr 0 SWAP
      BEGIN @succ^ 0<>
        WHILE                  \ solange es etwas zum
```

```
                                \ Erben gibt...
        >CFA 2 PICK GetClass$ \ String aus 'Erbmasse' und
                                \ eigenem
            " ." Add$ COUNT     \ Klassennamen
                                \ zusammenbauen
            2 PICK GetClass$
               Add$ COUNT
            $>Here "CREATE      \ Wort erzeugen
            DUP @ GetMyCFA !    \ Inhalt der CFA kopieren
            GetMyCFA >SuccPtr   \ Liste der ererbten
                                \ Eigenschaften
            3 PICK >EignPtr     \ erstellen
            0 SWAP Me>Prev!
            0 ,                 \ Zeiger auf Nachfolger
            2 PICK >Summe ,     \ Zeiger auf das
                                \ Summenfeld Klasse
            0 ,
          >SuccPtr
       REPEAT 2DROP
       >PrntPtr 0 SWAP !
  ELSE 2DROP
  THEN ;

: Neuefrucht ( -<name>- )
    " DEF." Create+Prefix
    GetMyCFA >succPtr
    0 Anker Me>Prev!    \ Mich selbst im Vorgänger
                        \ verankern
    0 ,                 \ Zeiger auf Nachfolger
    0 ,                 \ Zeiger auf Eigenschaften
    0 ,                 \ Summenfeld der Klasse
    Parent!             \ CFA von Erblasser oder
                        \ NULL kompilieren
\ --- Die Datenstruktur ist bereits fertig, der Code ist längst
\     definiert. Da kann das neue Wort auch schon aufgerufen
\     > ExecuteMe < werden, damit das mit dem Erben
\     wirklich klappt.

    ExecuteMe

    DOES> >CFA          \ Datenadresse in CFA umwandeln
        DUP Inherit     \ erben, wenn möglich
        DUP DefSub$ Create+Prefix
                        \ <ClassName.SubName>
        DUP >R GetMyCFA >succPtr
           0 R> >EignPtr Me>Prev!
                        \ Mich selbst im Vorgänger verankern
           0 ,          \ Zeiger auf Nachfolger
        DUP >Summe ,    \ Zeiger auf Summenfeld ( Klasse )
           0 ,          \ Summenfeld der Eigenschaft
        DROP ( CFA )    \ Erweiterungen sind möglich...
        DOES> DEPTH 1 >
            IF >CFA
              2DUP >Summe +!
                 >SummPtr @ +!
```

Soeben erfahren:

Die **Jahrestagung 2000** in Hamburg wird voraussichtlich vom **14.-16./04.** stattfinden. Dies meldet der Organisator der Tagung, Klaus Schleisiek heute per E-Mail an unsere Redaktion.

fep

Wordinfo
Ein nützliches, kleines Tool für die Arbeit mit
WIN32FOR

Die Stackbilanz von DUP haben wir alle parat. Bei Worten die weniger häufig genutzt werden, sieht das meist anders aus. Das Nachsehen via VIEW im Quellfile ist mir zu umständlich. Man verliert dabei zu schnell den gedanklichen Faden der Arbeit, die man eigentlich gerade leisten möchte. Hier hilft mir Wordinfo...

...Laden; auf der Console Forth Worte eingeben; dann mit der rechten Maustaste auf ein Forth-Wort klicken; (Es erscheint ein Popup) danach mit der linken Maustaste irgendwo auf die Console klicken **UND** die Maustate **WEITERHIN GEDRÜCKT** halten. Wenn man keine Lust mehr hat - Maustaste lösen! Sehen Sie selbst...

```
\ Wordinfo, eine Hilfe beim editieren

: cfa>source    ( a1 -- line )
            get-viewfile 0= abort" Undefined word!"
            over >view @ 0<
            if      ." CONSOLE" 2drop 0 -1
            else    count "path-file
                    abort" File not found!"
                    cur-file place
                    >view @             \ Zeilennummer holen
            then    ;

Create Linebuffer 256 allot
0 Value wordinfofile

: read#line ( n -- )
  0 swap                \ dummy
  0
  ?DO linebuffer 1 + 255 wordinfofile fread-line
     abort" error reading Wordinfofile!"
     drop swap drop
  LOOP linebuffer c! ;

Internal also
: .comment ( x y -- )
\  linebuffer swap Ascii ( scan type ;
  linebuffer
  conhndl get-mouse-xy
  start: InfoWindow ;

: "showinfo ( c-adr -- )
  caps-find
  if      cfa>source
          cur-file count r/o fopen-file
          abort" Sourcefile not found!"
          to wordinfofile
          read#line
          .comment
          wordinfofile fclose-file drop 0 to wordinfofile
```

```
  else    c@ abort
          cur-line @ cur-file
  then    ;

: .wordinfo     ( -<name>-  )
  bl word "showinfo ;

: info-aus close: InfoWindow ;

Create wort_puffer 80 allot

: mouse-info ( -- )
  [ hidden ] word@mouse" wort_puffer place [ forth ]
  wort_puffer
  "showinfo ;

: NewRightMouseClick ( -- )                 \ Handle a right
                                            \ mouse click
            info-aus
             mouseflags 3 and 2 <> ?EXIT \ exit if not right
                                            \ mouse clicked
             mouse-info ;


mouse-chain chain-add NewRightMouseClick
\ mouse-chain chain-add info-aus

\S wichtige worte hcst (highlight columnstart)
 word@mouse"
```

Martin Bitter

Le .. Le .. Lee-Ef..f..ffekt

"Noch mal - schön langsam!"

Erfahrungen mit Windows und Win32For

von Martin Bitter
Mehrhoog

Der Kongress

Am 27. und 28. Mai 1999 fand der Kongress "10 Jahre Gemeinsamer Unterricht im Kreis Wesel - es ist normal anders zu sein" statt.
"Gemeinsamer Unterricht" ist das Bestreben, behinderte Schüler und Schülerinnen zusammen mit nichtbehinderten SchülerInnen zu unterrichten. Der ein oder andere mag inzwischen wissen, daß ich als Sonderschullehrer innerhalb eines solchen Schulversuches an einer nordrheinwestfälischen Hauptschule tätig bin. So rutschte ich (eher widerstrebend) in das Organisationsteam hinein.
Da zur Zielgruppe besonders Menschen gehörten, die (noch) nicht mit gemeinsamen Unterricht beschäftigt waren und die ganze Veranstaltung nicht nur Information vermitteln, sondern auch Spaß machen sollte, wurde als "Eisbrecher" ein Block mit Selbsterfahrungsexperimenten geplant.
Auf verschiedene Art und Weise konnten die Besucher 20 Minuten lang ausprobieren, wie es ist, wenn der Gleichgewichtssinn gestört ist, die Hände nicht das tun, was man will, die Welt auf dem Kopf steht, das Auge alles verzerrt sieht, man auch laute Sprache einfach nicht versteht, Tische und Stühle einem urplötzlich in den Weg springen.
Und, das war uns das Wichtigste: Wie man in einer solchen Welt zurechtkommt, wenn die (erwachsene) Umwelt unerfüllbare Anforderungen stellt: "Nun reiß dich mal zusammen!" oder überbehütend reagiert: "Ach der Arme, der kann das ja gar nicht!"
Während der Vorbereitungen bot ich mich an, ein Programm zum Lee-Effekt so zu erstellen, daß es ebenfalls als Selbsterfahrungsexperiment angeboten werden könnte. Ich erhielt den nachdrücklichen Auftrag, das zu machen.
Da saß ich nun mit knapp acht Wochen Zeit!

Lee-Effekt?
Jeder Mensch kontrolliert sein Aussprache mehr oder weniger bewußt. Dabei gibt es mehrere Regelkreise: einen inneren Regelkreis, bei dem der beabsichtigte Laut über die Stellung der Artikulationsorgane (Mundraum, Kieferwinkel, Lippenstellung und –öffnung, Zungenstellung. Stellung des Gaumensegels und einiges mehr) kontrolliert wird, und einen äußeren Regelkreis, bei dem das Gehörte Ergebnis mit dem beabsichtigten Laut verglichen wird. Gegebenenfalls wird die Stellung der Artikulationsorgane nachgeregelt, bis das Gehörte mit dem Beabsichtigten genau genug übereinstimmt.
Während des Spracherwerbs in der frühen Kindheit erlernen Menschen das "richtige" Artikulieren in einem hohen Maße über den äußeren Regelkreis. Mit zunehmendem Alter läßt diese Kontrolle nach, man spricht sozusagen auswendig. Das ist einer der Gründe, weshalb "alte" Kinder und Erwachsene, nach einem Wohnortwechsel noch viele Jahre lang durch eine nun fremde Lautung ('falscher' Dialekt) auffallen.
Dennoch spielt das Selbsthören weiterhin eine wichtige Rolle. Kinder und Erwachsene, die schwerhörig werden, zeigen oft eine verwaschene Aussprache. (Ein Tip für's Leben: Kennen Sie jemanden, der plötzlich seine Aussprache ändert, lassen sie das Gehör untersuchen! Typisch Lehrer? Sie werden sich wundern, wieviel unerkannte Schwerhörigkeiten und schlecht ausgeheilte Mittelohrentzündungen ich an der Sprache erkannt habe! Und jedesmal war ich schrecklich sauer!!!)
Hier greift nun der **Lee-Effekt** ein: **durch gezieltes Stören des auditiven Regelkreises werden normalsprechende Menschen zum Stottern gebracht!**
Interessant ist die Geschichte des Lee-Effektes: Zur Zeit des Vietnamkrieges gab es einige Leute, die eine Taubheit simulierten, um als wehrunfähig zu gelten. Das war damals nicht widerlegbar und so wurde Mr. Lee beauftragt, eine Methode zu finden, die solche Simulanten ans Messer, Pardon: an die Front, lieferte.
Mr Lee fand heraus, daß hörende Menschen stotterten, wenn ihnen ihre eigene Sprache leicht verzögert (ca. 100-200 ms) über Kopfhörer eingespielt wurde. Wer dann stotterte, konnte

hören, war also ein Simulant! Im anglo-amerikanischen Bereich heißt der Lee-Effekt auch DAF (delayed auditory feedback).
Als Nebeneffekt stellte sich heraus, daß einige Stotterer unter dem Lee-Effekt flüssig sprachen! Es folgte eine euphorische Phase unter vielen Sprachtherapeuten, eine Menge elektromechanischer Lee-Geräte wurde verkauft. Leider haben sich die Erwartungen in die therapeutische Wirkung dieses Effektes in der Folgezeit nicht erfüllt.

Ein Demoprogramm
Vor Jahren hatte ich ein Lee-Effektprogramm geschrieben, das auf einem ATARI ST 500+ (8MHZ, 1024 KB, keine Soundkarte!) lief und zu Ausbildungszwecken benutzt wurde.

Nun waren die Vorgaben andere: auf mehreren Rechnern sollte das Programm laufen (damit schied das antike Rechnergerät aus: ich habe nur einen ATARI), es sollte ohne weitere Konfigurationsorgien auf verschiedenen Rechnern arbeiten, es sollte auf eine Diskette passen (zum Weitergeben), es sollte von (geübten) Computerlaien zu bedienen sein.
All dies "gipfelte" in der Entscheidung: Win32for und Windows95/98.

Warum Win32For?
Ich kenne zwar einige Forth Versionen, die unter Windows 95/98 laufen (comForth, Swift-Forth), aber am vertrautesten bin ich nun mal mit Win32For, was nicht bedeutet, daß ich es gut kenne! Plattform und Sprache standen also fest.

Zwei Schwerpunkte waren gesetzt: die technische Realisierung und die visuelle Gestaltung. Für beides sollte die Windows API herhalten. Für die Technik mit der Funktionsfamilie aus der Multimedia Library, hier die basalen Routinen mit dem einleitenden Begriff wave: waveOutOpen, waveInOpen etc. Nachzulesen mit mm.hlp, das man bei Inprise (vormals Borland) "downloaden" kann: *http://www.inprise.com/searchsite/index.html* (der genaue Standort wechselt, ab hier selbständig weiter suchen).
Für die Gestaltung der Oberfläche wurde kein Construktion-Set benutzt, sondern "nur" die in Win32For eingebauten Worte und der mitgelieferte Resourcecompiler von Michael Schroeder.
Etwas Stöbern im Internet brachte mich auf die Homepage von Jeff Kelm *http://www.concentric.net/~jkelm/win32for/classidx/index.htm* der verschiedene "Controls" aus der comctr32.dll für Win32For nutzbar gemacht hat.

Sich ein Bild machen
Relativ schnell war die Oberfläche entworfen: ein waagerechter Schieber für die Stärke des Lee-Effektes (Verzögerung in ms), ein senkrechter Schieber für die Lautstärke, ein weiterer waagerechter Schieber für die Balance und ein Textfenster mit Rollbalken für evtl. Lesetexte oder Hilfen und Erklärungen. Ich versuchte durch eingescannte Bilder, die als "StaticBitmap" eingebunden wurden, diese Oberfläche selbsterklärend zu machen.

Ein Problem tauchte auf und kostete recht viel Zeit: Die Oberfläche ließ sich unter Win98 starten, stürzte aber mit einer Fehlermeldung "Non existent Window!" ab. Dies aber nur in der Turnkey-Version! In Win32For geladen, lief das Programm. Lange Zeit vermutete ich den Fehler bei Win32For. Irgendwann fiel mir auf, dass das "geturnkeyte" Programm sich starten ließ, wenn vorher eine Fileselector-Box geöffnet wurde. Das war ein Hinweis auf eine fehlende Initialisierung, die durch das Starten von Win32For bzw. durch den Aufruf der Fileselectorbox automatisch geschah. Eine solche Initialisierung war aber in Win32For nicht zu finden! -- Das Ende vom langen Lied: es gibt tatsächlich eine Windowsroutine die InitCommonControls heißt und die Comcnt32.dll für Win95 erst nutzbar macht (Dank an Tom Zimmer!). Von Win32For wird sie implizit aufgerufen, unter Win98 ist sie nicht notwendig!
Mit dieser Kentniss startete die Oberfläche unter beiden Windows-Versionen ohne murren.

Die Stimme seines Herrn
Ebenso "erfolgreich" verlief die Programmierung der technischen Seite: schon beim ersten Versuch lief der Lee-Effekt! :-)
Es wurde (zu Testzwecken) ein sehr großer Puffer für einkommende Wavedaten geöffnet, alle gesampelten Daten aus der Soundkarte dort gespeichert und im Gleichtakt mit einem gehörigen Abstand zu der jeweiligen Schreibposition ausgelesen.
Aber!!! sobald die Leseposition sich der Schreibposition zu sehr näherte, überschlug sich alles: die Wiedergabe wurde zerhackt, unverständlich. :-(
Langes Experimentieren ergab: Egal welches Verfahren ich anwendete, der Abstand zwischen der aktuellen Schreibversion und der aktuellen Leseposition durfte nicht enger sein (in Sampleblocks), als es einem zeitlichen Abstand von 170 ms entsprach!
Das paßte nicht für den Lee-Effekt! Anscheinend lesen (und schreiben) die Windowsfunktionen die gesampelten Daten nicht Sampleblock für Sampleblock (d.h. bei 16 Bit Stereo hat ein Sampleblock 2 Byte), sondern in größeren zusammenhängenden Blöcken. (vgl. Abbildung 1)

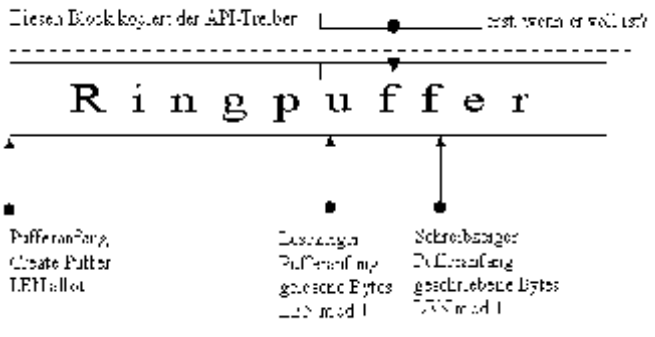


Abb.: 1

... und bist du nicht willig ...
Ich mußte eine Weg finden, Windows zu zwingen, kleinere Portionen aus den Paketen zu machen.
Den fand ich auch: Es ist möglich, den Puffer, der bei Win-

dows für eine Aufnahme angemeldet wird, sehr klein zu machen. Indem nun ein solcher Aufnahmepuffer gerade so groß gehalten wird, wie es dem (zeitlichen) Abstand zwischen Aufnahme und Wiedergabe entspricht, ist das API gezwungen, diesen Aufnahmepuffer ganz zu füllen, bevor er zur Wiedergabe ausgelesen wird.
Im einzelnen funktioniert es so: Insgesamt werden vier gleichlange Puffer (Windows –> Waveblocks) benutzt. Zwei für die Aufnahme (waveInOpen) und zwei für die Wiedergabe (waveOutOpen). Die Puffer sind so berechnet, daß ein einzelner Block der halben Länge des Abstandes zwischen Aufnahme und Wiedergabe entspricht.
Zur Darstellung numeriere ich die Blöcke durch: Block 0 und Block 1 dienen der Aufnahme, Block 2 und Block 3 der Wiedergabe. Zu Beginn werden alle vier Blöcke mittels der API-Funktionen angemeldet.

Sowohl bei der Aufnahme, als auch bei der Wiedergabe benutzt Windows die Blöcke in der Reihenfolge ihrer Anmeldung!

Zwar gibt es bei abspielenden Funktionen die Möglichkeit, Windows einen Block (oder eine Reihe von Blöcken) 'endlos' abspielen zu lassen - aber bei der Aufnahme fehlt eine entsprechende Funktion. Das Aufnehmen und Abspielen in Schleifen wird also im Lee-Effekt-Programm von Win32For geregelt.

Die innere Programmschleife ist so gegliedert:

Block 0 und Block 1 (in dieser Reihenfolge) zur Aufnahme anmelden.

Block 2 und Block 3 (in dieser Reihenfolge) zur Wiedergabe anmelden.

Aufnahme starten und (durch Pollen) überprüfen, ob Block 0 gefüllt ist.

Sobald Block 0 gefüllt ist, seinen Inhalt nach Block 2 kopieren und das Abspielen von Block 2 starten.

Dann Block 0 abmelden und sofort wieder zur Aufnahme anmelden. Windows bearbeitet im Moment noch (in der Aufnahme) Block 1 'merkt' sich aber, daß es danach bei Block 0 weitermacht.

Nun überprüfen (wieder durch Pollen), ob Block 1 mit Daten gefüllt ist

- falls ja, den Inhalt von Block 1 nach Block 3 kopieren.

Wiedergabe bei Block 3 starten. Block 1 abmelden und sofort wieder anmelden.

Wir haben jetzt folgende Situation:
Block 0: wird bespielt
Block 1: ist bespielt (mit alten Daten)
Block 2: ist gefüllt (mit alten Daten)
Block 3: ist noch (wird abgespielt)

(vgl. Abbildung 2)

Das Ganze wird solange wiederholt, bis ein Abbruch (meist eine neue Einstellung der Verzögerung = des Abstandes) gewünscht wird.
Für einen Lee-Effekt mit variabler Verzögerung muß diese innere Schleife nur noch in eine äußere Schleife gepackt werden und: fertig!

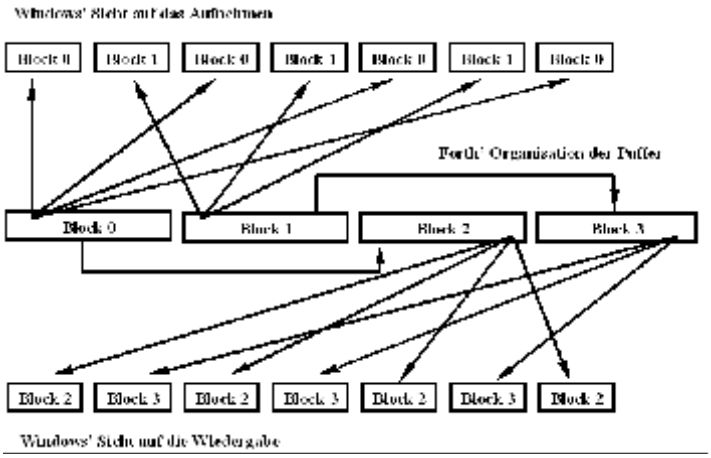


Abb.: 2

Das Hauptwort sieht dann so aus:

```
: leeprg ( -- )
    Call InitCommonControls DROP
           \ comcnt32.dll dem System anmelden
    karte? \ erfüllt die Soundkarte
           \ die Anforderungen
    IF Start: LeeWindow
           \ Ja? Oberfläche starten
    BEGIN  \ äußere 'endlos'-Schleife
      lee_open    \ Initialisierungen;
                  \ benötigte Puffer 'öffnen'
      aufnahme_start   \ Aufnahme starten
      BEGIN            \ innere 'endlos'-
                       \ Schleife beginnen
        0 header# fertig? IF 0->2copy
                          THEN
        \ evtl. Block 0 nach Block 2 kopieren
        1 header# fertig? IF 1->3copy
                          THEN
        \ evtl. Block 1 nach Block 3 kopieren
        ekey? drop \ gib WINDOWS eine Chance
        delay_changed: LeeWindow
        \ wurde im Programmfenster etwas
        \ geändert?
      UNTIL \ evtl. innere 'endlos'-Schleife
            \ wiederholen
      FALSE to delay_flag \ ansonsten ein
                          \ flag setzen
      aufnahme_ganz_aus   \ Aufnahme und ...
      wiedergabe_ganz_aus \ ... Wiedergabe
                          \ stoppen
     AGAIN \ äußere Schleife wiederholen, der
           \ Abbruch erfolgt gegebenenfalls
           \ durch ein BYE im Fenster
ELSE bye   \ keine passende Soundkarte
           \ gefunden--> Tschüss
THEN ;     \ fertig
```

Selbst mit Pollen bleibt zwischen den einzelnen Kopierfunktionen noch enorm viel Zeit für Anderes (96.000 mal DO LOOP !) Hier ist noch Raum für entsprechende Manipulationen, Verzerrungen u.ä.

Der große Tag!

Die Durchführung des Kongresses gestaltete sich mindestens so anstrengend, wie erfolgreich! Meine Schüler führten ein Theaterstück auf; zu dem Workshop den ich (mit) leitete, er-

hielten wir positive Rückmeldungen und die Selbsterfahrungsexperimente waren ein 'Renner'.

Das Programm Lee-Effekt lief acht Stunden am Stück - ohne Absturz! (Puh ;-) Die Rechner standen unbetreut in einer ruhigen, etwas abseits gelegenen Ecke. Ab und zu konnte ich einen Blick auf kleine Gruppen von Kindern und Erwachsenen, die sich gegenseitig abhörten, erhaschen. Es gab viel Spaß dabei. Mehr Menschen als ich dachte, waren absolute Computerlaien (“Das ist doch so ein Diktierprogramm? Mein Sohn hat jetzt auch so was.”) kamen aber trotzdem mit der Oberfläche zurecht. Manche (Kinder) spürten wohl, daß sie beim Sprechen gestört wurden, merkten aber nicht, daß sie stotterten. Das verwunderte wiederum die Umstehenden. Einige Kinder gaben ihren eigenen Kommentar ab: “Voll das Verarschungsprogramm, ey!” Aus der Sicht der Veranstalter hat das Programm seinen Zweck voll erfüllt. Ein Lehrer war so begeistert, das er 'auf jeden Fall' eine Kopie haben will. Er fragte auch nach der verwendeten Sprache (ach Forth?). Ein Hörgeräteakustiker (zuständig für Experimente zu Hörstörungen) bot sogar Geld! Er hatte während seiner Ausbildung elektromechanische Lee-Effektgeräte gebaut (diese waren sehr anfällig).

Fragen ...

Ich habe von Windows-API wenig Kenntnisse! Deshalb bleiben viele Fragen offen.

- Wie erfragt man, ob ein Mikrofon am Line-in-Eingang einer Soundkarte angeschlossen ist?
- Bei mehr als einem Line-in-Eingang: wie erfährt man, an welchem evtl. ein Mikrophon angeschlossen ist?

Im Moment muß all dies der Benutzer selbst über die Mixer-Funktionen, die der Treiber der Soundkarte mit sich bringt, selbst einstellen.

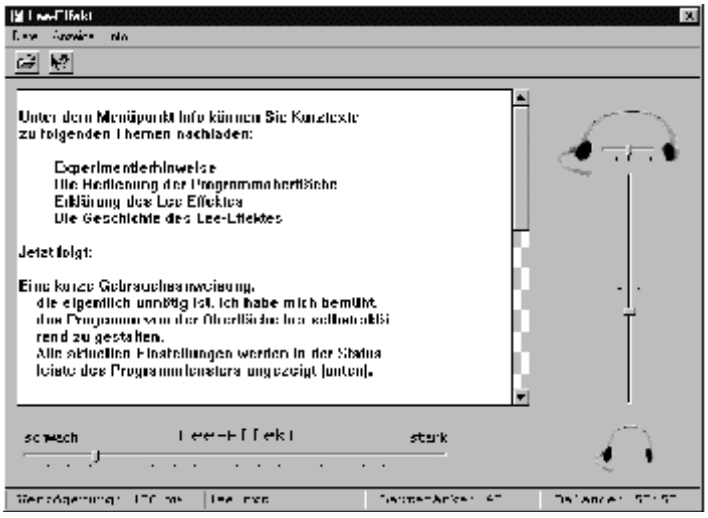
- Wäre der Lee-Effekt auch mit den Filter-Funktionen der mm.dll zu realisieren gewesen (durch die Beschreibungen habe ich nicht durchgeblickt)?
- Gibt es wirklich keine Möglichkeit mit WINDOWS-API feiner als 1 ms aufzulösen (nur in Verbindung mit AVI-Funktionen)?

Meine Erfahrungen?

Diese sind ganz persönlicher Art. Sobald man auf eine API angewiesen ist, ist man (bin ich) den Programmierern der API gnadenlos ausgeliefert. Wenn eine Funktion, die eine Soundkarte (oder irgendeine andere Hardware) unterstützt, nicht über die API anzusprechen ist, gucke ich so :-| .
Auf der einen Seite werde ich von der Funktionsvielfalt erschlagen, auf der anderen Seite fällt es schwer, die richtige Funktion zu finden. Trotz Hilfedateien komme ich oft nicht weiter (kaum Einträge zu microphone oder headset). Manchmal fällt es sehr schwer, den Fehlerbereich einzugrenzen. Ist es WINDOWS, ist es FORTH? (vgl. InitCommonControls) Die Portabilität zwischen Rechnern, wegen der ja unter WINDOWS programmiert wurde, ist nicht 100%ig, es gibt (kleine?) Unterschiede zwischen WIN95 und WIN98 Rechnern.
Andererseits: Die Verwendung von WINDOWS war unbedingt notwendig, da nur WINDOWS verbreitet genug ist (in unserem Fall).
Einige Male war ich kurz davor, mit einem Dissassembler in den API-Code einzusteigen, dieser Kelch ging glücklicherweise an mir vorbei.
Der Quellcode zum Lee-Effekt steht zur Verfügung: mit zwei Aber!
Aber nur weil Fritz Prinz drauf gedrängt hat. Aber nur wenn jeder, der ihn anguckt, vorher dreimal sagt: ich wundere mich über nichts!
Das Programm wurde unter Zeitdruck geschrieben, ist schlecht dokumentiert, und 'historisch' gewachsen, das heißt in Vielem inkonsistent und unlogisch. Es 'schön' zu machen, dazu fehlt mir jetzt der Antrieb (vielleicht bei entsprechender Nachfrage?)

PS. (Achtung: Moral!)
Das Thema 'meines' Workshops behandelte die Frage des Übergangs in die Berufswelt für behinderte Schüler aus dem “Gemeinsamen Unterricht”. Viele dieser Schüler haben sich in der 'normalen' Schule behaupten können. (Damit kein Irrtum aufkommt: sie sind behindert!) Und sie wünschen sich einen Beruf in der 'freien' Wirtschaft. Leider gibt es bei so manchem Arbeitgeber (noch) immer falsche Informationen z. B. über den Kündigungsschutz, die ihn davon abhalten, behinderte Menschen einzustellen.
Sollte der ein oder andere Leser oder Leserin selbständig sein, oder sonst Einfluß auf Personalfragen haben - warum nicht mal benachteiligten Menschen ein Chance geben. Papierkörbe kann man leeren, ohne studiert zu haben! Nähere (überregionale) Informationen gibt es bei der Bundesarbeitsgemeinschaft für Unterstützte Beschäftigung *http://www.bag-ub.de/* oder *http://bidok.uibk.ac.at/impulse/index.html#963*

Martin Bitter



Nutzeroberfläche LEE-Effekt

Betreff: VD-Leserbrief
Datum: Wed, 09 Jun 1999 12:18:04 +0200
Von: Fred Behringer Rückantwort: behringe@mathematik.tu-muenchen.de

Tagung ausgezeichnet - Leserzuschriften könnten mehr werden

Und wieder ist eine Tagung hervorragend gelungen. An sich kein Wunder: Heinz und Ulrike Schnitter sind anerkannte Experten im Organisieren. Und trotzdem darf man es ihnen ruhig nochmal bescheinigen: Das war eine Meisterleistung. Das Hotel und die Umgebung waren enorm, und eine so gelöste Stimmung - das setzt Maßstäbe.

- Ich selbst möchte mich herzlich für den mir anvertrauten Drachen bedanken. Das spornt ungemein an. Lob ist immer gut. Das gilt aber auch für die diversen Autoren, die, irgendeinem undefinierbaren Drang folgend, Artikel für die VD schreiben. Ohne jeden einzelnen von ihnen hätte unser Verein keinen dauerhaften Bestand. Und für den übergroßen "Rest" gilt: Rühren Sie sich! Rührt Euch! Nichts ist für einen Autor schlimmer als keine Reaktion. Dann schon lieber harsche Kritik. Am liebsten aber Zustimmung oder konstruktive Alternativvorschläge. Manchmal schlägt man sich als Autor (auch als VD-Autor) lange Zeit mit einer Idee herum und gießt das Ganze dann in eine leidlich brauchbare Form. Manchmal "überkommt" es einen aber auch ganz einfach. Wie bei Goethe. In Schüben. Dann fallen einem ganz verschiedene Gedanken parallel zur gleichen Zeit ein. In der U-Bahn, im Restaurant oder an anderen unmöglichen Orten. Und die Hand kann gar nicht so schnell schreiben, wie das Gehirn schon vorgearbeitet hat. Dann sitzt jedes Wort und man möchte nichts mehr verändern.
Für solche Fälle wäre ein Selektionsmechanismus der Form "Turbo-Forth - interessiert keinen; ZF - schon eher" recht nützlich. Man könnte seine Energien dann in meistgewünschte Richtungen hin kanalisieren. Ein (für mich) aktuelles konkretes Beispiel: Ich zerpflücke gerade ZF. Das haben schon viele vor mir getan: Friederich Prinz, Martin Bitter, Thomas Beierlein - um nur einige zu nennen. Natürlich vergleiche ich mit Turbo-Forth, meinem Lieblingssystem, und es fallen mir viele Dinge gleichzeitig ein. Wie schön wäre es, wenn ich wüßte, wieviele VD-Leser eigentlich jemals ZF angekurbelt haben. Ich könnte ja, bei zu geringer Zahl, meine Gedanken leicht auch für Turbo-Forth umformulieren. Woher
soll ich aber die gewünschten Informationen bekommen, wenn keine Rückmeldungen eingehen?
Ganz konkret: Wer beschäftigt sich mit ZF oder hat sich jemals mit ZF beschäftigt? Wenn auch nur spaßeshalber erbeten, um anstrengungslos Bewegung ins Vereinsleben zu bringen: E-Mail oder Brief an mich. "Ich. Gruß xyz" genügt. Ich werde die Reaktionen dann irgendwie verarbeiten. Die ersten drei, die sich melden, werden von mir auf jeden Fall irgendwann irgendwo lobend erwähnt.

Fred Behringer

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

Forth Dimensions der Forth Interest Group, USA

November/Dezember 1998

Lange hat es gedauert und schwer scheint es zu sein, nach der offensichtlich gelungenen FORML-Konferenz, für die ja Marlin Ouverson, der Redakteur, ebenfalls zuständig zeichnete, den zeitlichen Anschluß für die alle zwei Monate erscheinende Forth Dimensions wiederzufinden und zu halten. Das jetzt vorliegende neueste Heft (vom November/Dezember 1998) hat es dafür aber auch wirklich wieder in sich. Weiter so!

4 Preparing for the Future
Marlin Ouversons Editorial

Diesmal ein flammender Aufruf, die Techniken und Errungenschaften anderer Sprachen zu übernehmen, um Forth zu dem zu machen, was es schon immer sein wollte, aber nicht immer geschafft hat zu sein: Ein weithin akzeptiertes Mittel, mit dem man das, was in anderen Sprachen getan werden kann, zumindest auch tun kann, und manchmal bis meistens sogar besser. Mit der Betonung auf "weithin akzeptiert". Elektronisches Zurverfügungstellen der Artikel und Programme der Forth Dimensions im Internet ist dabei eines der Stichworte, die Marlin aufzählt.

5 DynOOF-style Objects for the i21 microprocessor
András Zsótér

Der Autor gibt 3 frühere Arbeiten an, in denen er sich schon mit OOP in Forth beschäftigt hat. "Forth hat mit CREATE ... DOES> schon immer die wesentlichen Bestandteile des OOP enthalten. Moderne OOP-Techniken erfordern aber etwas mehr. Trotzdem, in Forth läßt sich das alles sofort, schnell und effizient verwirklichen. Der i21 ist eine Stack-Maschine, die dem Programmierer möglichst wenig abverlangt. Auf den ersten Blick scheint er kein guter Kandidat für OOP in Forth zu sein. Hier soll gezeigt werden, daß es geht und daß sich ein solches Vorhaben auch für den kommerziellen Bedarf eignet." - Dr. Zsótér ist bei iTV beschäftigt. Wer sich für IRC interessiert, wird auf den zugehörigen Internet-Seiten unweigerlich auch auf iTV (preiswerter Internetzugang) stoßen. Übrigens wurde der i21 unter der Leitung von Charles Moore entwickelt.

8 Cross-compilers and Embedded Systems
Elizabeth D. Rather

Was von Charles Moores Mitstreiterin der ersten Tage in Sachen Forth kommt, ist unbesehen gut. Jeder, der mal ein

Forth-System für einen anderen Prozessor auf einem PC hochgezogen hat, kennt das Problem mit dem Cross-Compiler: Ähnlich und doch verschieden. Eigentlich fängt das schon beim Cross-Metacompiler an. Bei Stand-Alone-Anwendungen für Embedded Systems sind es die gleichen Probleme. Hier die Worte der Autorin: "Ein wichtiger Diskussionspunkt bei den ANS-Forth-Bemühungen waren Embedded Systems und Cross-Compiler. Diese Dinge bilden einen wesentlichen Bestandteil der Forth-Anstrengungen. Forth Inc. (Elizabeth Rathers Firma) und MPE (MicroProcessor Engineering Ltd.) haben 1996 gemeinsam einen Satz von Standard-Empfehlungen für solche Systeme ausgearbeitet. Diese wurden anschließend in kommerziellen Systemen erprobt und inzwischen liegen eine ganze Reihe von guten Erfahrungswerten vor, auf denen man Zusätze zum ANS-Standard aufbauen kann - Ich (die Autorin) schlage vor, für die Cross-Compilation einen zusäzlich wählbaren Wortsatz einzuführen ..."

11 FORML Conference #20, 1998
Richard Astle

Zweieinhalb Seiten Bericht über die Jubiläumskonferenz (20 ist die FORML geworden). Dr. Ting war wieder aktiv, Glen Haydon trieb die Forth-Philosophie ein weiteres Stück voran, Elizabeth Rather ist für noch mehr Standardisierung und Chuck Moore scheint dauernd mit neuen, überraschenden Vorschlägen aufzuwarten und alles in Frage zu stellen. Standardisierung und die damit eventuell verbundene Einschränkung der kreativen Bewegungsfreiheit scheint ihm überaus suspekt. Warum hört eigentlich niemand auf den Forth-Erfinder? Der muß es doch wissen! "There were new faces from Germany and Australia", berichtet der Berichterstatter. Eines dieser "neuen Gesichter", Heinz Schnitter und seinen Beitrag, erwähnt er ausdrücklich. Heinz bekam aber auch den Preis für den längsten Vortrag. Der Preis für die längste Anreise ging an den Teilnehmer aus Australien. Dieser Bericht ist für mich, den Rezensenten, ein guter Gradmesser dafür, wieweit ich mit den Namen auf dem internationalen (nicht nur amerikanischen) Forth-Parkett und deren Ansichten und Meinungen inzwischen schon vertraut bin. Ein Neuhinzukommender könnte aber von soviel Insider-Information leicht erdrückt werden. Aber so geht es ja wohl mit jedem Verein. Besonders bei solchen mit Sendungsbewußtsein. Was uns noch interessieren könnte, nur so zum Vergleich: Die FIG-Mitgliederzahl ist auf 670 geschrumpft. Das weltweite Interesse bleibt aber groß: Die Zahl der Zugriffe auf die FIG-Webseite beläuft sich auf 500 pro Tag.

14 Simple Object-Oriented Programming
Wil Baden

Prima Artikel des "Stretching-Forth"-Autors über objektorientierte Programmierung in Forth. Ziel: Leicht zu lesen, leicht zu schreiben und schnell. 2 Seiten Text, 2 Seiten Programm. E-Mail an wilbaden@netcom.com mit der Bitte um Stretching Forth # 23: SOOP. Geht zurück auf einen comp.lang.forth-Beitrag von Helge Horch. Ist inzwischen ergänzt worden von Rick VanNorman (mir als Autor von OS/2-Forth bekannt). Rick wird in der nächsten Ausgabe der Forth Dimensions einen Bericht geben. Der Rezensent: Am Anfang des Artikels steht ein ganz einfaches Beispiel (5 kurze Zeilen) in Forth. Jetzt weiß ich endgültig, was eine Klasse ist: Forth konnte es schon immer, wie ich stets vermutete, aber nur rudimentär (CREATE...DOES>). Die Exemplarisierung (Instanzierung?) scheint das Wesentliche zu sein, nicht die Abkapselung (oder Einkapselung oder Kapselung schlechthin). (Hat das nicht Ähnlichkeit mit den Dokumentvorlagen bei WORD?) Glänzende Präsentation des Themas! Das Abstrahieren von diesem Beispiel kann ruhig mir, dem Leser, überlassen werden! Das prägt sich gut in mein Gehirn ein. So sollte man schreiben! Vom einfachen Beispiel zum komplizierten Sachverhalt hinleiten. Ich werde mir ein Beispiel an diesem Beispiel nehmen. Es mangelt dem Leser nicht an Kraft, etwas zu kapieren, es kommt auf die Bereitschaft an, etwas aufzunehmen. Meine Bereitschaft ist mit diesem kurzen Artikel gestiegen.

18 How and Why to Use Multitasking
Frank Sergeant

Frank ist als Autor von Pygmy Forth (mit inzwischen einigen Varianten) bekannt. Er exemplarisiert seine Gedanken in diesem Artikel anhand von Pygmy Forth, versichert aber, daß sie Allgemeingültigkeit haben. Die Schwerpunkte in Franks momentanen Überlegungen liegen in (1) "Wie baut man zuverlässige Software?" und (2) "Wie schafft man das am schnellsten?". Frank versucht, klar zu machen, daß beide Punkte über Multitasking erledigt werden können. "Falls Sie Multitasking bisher noch nicht verwendet haben, hoffe ich, Ihnen mit diesem Artikel einen Anstoß zu geben und Ihnen den Weg zu ebnen."

23 Forth and Functional MRI
Ronald T. Kneusel

Der Autor ist Programmierer am Medical College of Wisconsin und Teilzeit-Student der Biophysik in höherem Semester. (Interessant zu sehen, daß in den heutigen Forth-Umgebungen immer noch jeder seine Chance hat.) Magnetic Resonance Imaging (MRI) gibt es seit 20 Jahren. Im Gegensatz zu Untersuchungen mit Röntgenstrahlen oder radioaktiven Isotopen wirkt die MRI zerstörungs- und belastungsfrei über magnetische Resonanz von Wassermolekülen und Aussendung meßbarer elektromagnetischer Strahlung im Radiofrequenzbereich (für den Körper ungefährlich). Seit 8 Jahren untersucht man (oder versucht man zu untersuchen) mittels MRI auch Vorgänge im Gehirn. Hierzu reichen keine statischen MRI-Bilder. Eine schnelle Abfolge von Bildern, eine Dia-Show der Gehirntätigkeit, eben Functional MRI, ist gefragt. Ronald ist überzeugt davon, daß er der erste ist, der für diesen Zweck Forth einsetzt. Forth eignet sich gut dazu, sagt er. Er arbeitet mit einem Apple-Macintosh und verwendet Pocket Forth (Freeware) von Chris Heilman. Es sagt (hier)

nichts über das eigentliche Programm. Darüber hat er aber auch schon in Forth Dimensions XIX.3 berichtet.

26 The Problem with Buffers
Hugh Aguilar

Problem: Ein Datenpuffer, so groß wie nur irgendmöglich, der in gewissen Zeitabständen einen Schwall von Daten schnell aufnimmt, um diese dann Stück für Stück mit geringerer Geschwindigkeit an das Programm abzugeben. Die Daten werden seriell eingelesen und es bietet sich ein "wrap-arounding" (Ringpuffer wie beim IBM-Tastaturpuffer) an. Das eigentliche Problem möge darin bestehen, daß die Daten nur dann vernünftig verarbeitet werden können, wenn sie zusammenhängen (nicht der eine Teil vorn und der andere hinten, um die Wrap-Around-Ecke herum). Das Stichwort lautet Datenrotation innerhalb des Puffers. Der Autor verwendet einen "algorithm which is the author's own invention". Einen Algorithmus? Es ist von lauter "Pointern" die Rede. Zeitmangel und Ungeduld hinderten den Rezensenten daran, das Wesentliche zu erfassen. Zwei Zeilen Erklärung vorneweg wären hilfreich gewesen.

30 Reed-Solomon Error Correction
Glenn Dixon

Vorwärts-Fehler-Korrektur bei CDs, Satelliten, Übertragungskanälen ganz allgemein, besonders aber auch bei solch erstaunlichen Festplatten-Disketten-Zwitterwesen wie den Iomega-Produkten ZIP 100 und ZIP 250. (Der Rezensent: Ich schwöre auf das ZIP-100-Laufwerk. Da war ich wieder mal einer der ersten, die darauf angesprungen sind, auch wenn ich bei Windows 98 und anderem zögere.) Glenn ist bei Iomega beschäftigt und verwendet Forth "für praktisch alles, was seine Firma ihm erlaubt". Bewundernswert! Wir diskutieren über die Akzeptanz von Forth, andere gehen einfach hin und tun es. Der äußerst interessante Artikel ist zweigeteilt. Das nächste Mal mehr. Grundlage ist die Arithmetik in endlichen Körpern. (Der uns allen geläufige Körper der reellen Zahlen ist nicht von solcher Art.) Erstaunliche Eigenschaften: Die Subtraktion ist kommutativ und unterscheidet sich in nichts von der Addition! Und alles wird mit XOR und Verschiebungen erzeugt und erschlagen! In Forth alles ganz schnell machbar: 3 Seiten Listing. Programm anfordern bei Dixong@iomega.com . Im nächsten Teil die eigentliche vorausschauende Fehler-Korrektur: Blockweiser Einbau redundanter Elemente, die bei Bedarf eine Korrektur gestatten, ohne die Daten ein zweites Mal einlesen zu müssen. (Jetzt begreife ich, wie die bei Iomega das mit den 100, 250, ja sogar 2000 Megabyte geschafft haben.)

VIJGEBLAADJE der HCC Forth-gebruikersgroep, Niederlande

Nr. 15, Juni 1999

De 'Egel hoek
Willem Ouwerkerk

Beschreibung eines Analog-Digital-Wandlers für das "Igel"-Projekt mit dem AT89C2051 (Hard- und Software - natürlich in Forth). Das Prinzip, sagt der Autor, wird auch in den Atmel-Unterlagen beschrieben, wird hier aber vereinfacht angewandt. Prinzip: Komparator, Kondensator über Widerstand aufladen, am anderen Eingang Spannung anlegen, in kleinen Schritten aufladen, messen und vergleichen, Aufladezeit bis zur Spannungsdifferenz 0 ist ein Maß für die am anderen Eingang angelegte Spannung. 1 CODE-Definition, 5 Colon-Definitionen. Codelänge 133 Byte. Kann mit Trick um 35 Byte gekürzt werden. Mit Schaltbild: AT89C2051, 7 Widerstände, 6 Leuchtdioden, 1 Kondensator, 1 Potentiometer - das ist alles.

FORTH-code lezen
Albert Nijhof

Für RTYPE , LTYPE und TOKEN? je 3 verschiedene Definitionen und Frage an den Leser, welche die beste und warum.

De FORTH hoek
De Schoolmeester

Das ist die Schwierigkeit des Übersetzers: Wie erkenne ich Ironie als Ironie? Ich müßte im entsprechenden Umfeld leben. Es gibt Namen wie De Waal, De Vries, De Bakker, warum soll es nicht auch einen Autor namens De Schoolmeester geben? Andererseits dreht sich aber der ganze Artikel um die **schulmeisterliche** Beschwerde (eines Herrn Irgendwer an den Redakteur?) über 20000 40000 Max . [ret] 20000 ok . "Das kann doch nicht wahr sein." Kann es doch! In einem 16-Bit-System. 40000 = -25536 bei einem MAX , das mit vorzeichenbehafteten Ganzzahlen von 16 Bit Breite arbeitet.. Der (an sich recht lehrreiche) Artikel geht, wenn man sich die Verkleinerungen des Vijgeblaadjes wieder rückgängig gemacht vorstellt, über etwa 3 Seiten weiter und endet bei der Diskussion eines UMAX. - Da fällt mir ein: Man sehe sich in der neuesten Forthwrite (Nr.102) den Bericht über die momentane comp.lang.forth-Diskussion über "Charles Moores genialen MAX-Vorschlag" an. Da wird man nachdenklich. Da kann doch was nicht stimmen! Da ist doch sicher beim Berichterstatten die Nachrichten-Enthropie gestiegen?

Bijeenkomst

Auf dem diesmaligen Treffen (am 12.6.99 wieder in der Volkssternwarte Utrecht) hat Pieter Surie über CORDIC (Coordinate Rotation Digital Computer) gesprochen (Kreis- und Hyperbel-Funktionen und deren Umkehrungen, trickreich erzeugt).

Dragon Graphics

Forth, OpenGL und 3D-Turtle-Graphics

von
Bernd Paysan
(vorgestellt auf der Jahrestagung '99)

Zusammenfassung:

Eine 3D-Turtle-Graphics auf der Basis von OpenGL wird vorgestellt. Die Turtle bewegt sich im Raum, und zieht eine Spur, die das Skelett der 3D-Objekte definiert. Die Oberfläche wird ausgehend von der Turtle mittles verschiedener Koordinatensysteme (besonders beliebt: Zylinderkoordinaten) gesetzt. Normalenvektor und Texturkoordinaten werden algorithmisch generiert. Anhand zweier Beispiele, eines Baums und des Swap-Drachens, der dieser Technik auch den Spitznamen gibt, wird die Vorgehensweise demonstriert.

Einleitung

Auf der letzten Forth-Tagung *(1998, red)* habe ich eine direkte OpenGL-Anbindung in Forth vorgestellt. OpenGL ist eine 3D-Grafik-Library, die einem viel Arbeit abnimmt. Allerdings ist OpenGL relativ low-level, und bietet ,,nur" Koordinatentransformationen sowie das Zeichnen von Strips, also Aneinanderreihungen von Dreiecken oder Rechtecken an. Zudem benötigt OpenGL Normalenvektoren und Texturkoordinaten, die man aber automatisch berechnen kann.

Mein Vorhaben war daher, OpenGL in eine einfacher zu benutzende Library zu kapseln, eine Art 3D-Turtle-Grafik. Um den Jahreswechsel gab es eine Diskussion in comp.lang.forth über eine solche 3D-Turtle-Grafik. Dave Taliaferro stellte eine in pForth geschriebene 3D-Turtle-Grafik vor. Marcel Hendrix implementierte kurz darauf etwas Vergleichbares in iForth.

Beide Turtles können sich durch den Raum bewegen und hinterlassen dabei eine Spur aus OpenGL-Objekten, etwa Zylindern oder Kugeln. Man kann damit also keine komplexen Körper erzeugen.

Das hier vorgestellte System setzt auf dem Turtle-Prinzip auf, erlaubt es aber, Körper zu beschreiben. Da es diese nicht als Komposition aus starren Einzelteilen aufbaut, ist eine echte Skelettanimation möglich, etwas, was auch bei Hollywood-Tools noch mit viel Aufwand verbunden ist. Nicht zufällig haben die abendfüllenden Streifen Insekten, also Außenskelette, als Darsteller. Animationen mit Innenskeletten beschränken sich auf kurze Sequenzen. 3000 Punkte (der Drache) kann man auch nicht einfach von Hand eingeben.

Das Prinzip

Eine normale 2D-Turtle-Grafik kann vorwärts und rückwärts fahren, sowie sich nach rechts und links drehen. Dabei hinterläßt sie Spuren, also Striche. Das Prinzip läßt sich auch auf Flächen erweitern, indem man die von der Turtle gezeichneten Polygone auffüllt.

Im Raum ist die Turtle in ihrem richtigen Element (unter Wasser). Statt schwerfällig herumzukriechen, kann sie auch nach oben und unten schwimmen, sowie um ihre Achse rollen. Man muß sich nun überlegen, wie die ,,Spur" aussehen soll, und wie man von Strichen auf Flächen und noch wichtiger Körper kommt.

Statt einfach vorgefertigte Objekte fallenzulassen, erlaubt es diese 3D-Turtle-Grafik, Schnitte durch den Körper zu beschreiben. Diese Schnittebenen werden dann miteinander verbunden, um einen Körper zu formen. Um etwa einen Zylinder darzustellen, verbindet man zwei Kreise miteinander. Kreise werden durch Vielecke angenähert.

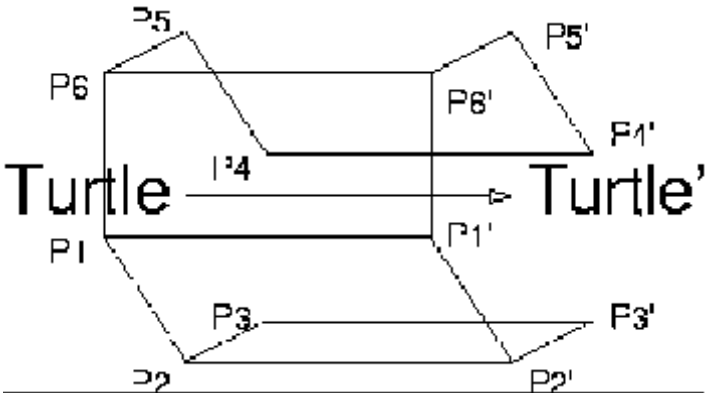


Abb. 1: 3D-Turtle-Prinzip

Die 3D-Turtle-Grafik stellt für diese Schnitte keine 2D-Turtle-Grafik zur Verfügung (obwohl das ja irgendwie naheliegend wäre), sondern verschiedene Koordinatensysteme, etwa Zylinder-Koordinaten. Man kann natürlich auch die 3D-Turtle verwenden, Umrisse abzufahren. Der Ursprung ist durch die Turtle bestimmt, die Ausrichtung des Koordinatensystem entspricht der Blickrichtung der Turtle.

Ein einfaches Beispiel

Als einfaches Beispiel soll uns ein Baum dienen. Ein Baum besteht aus einem Stamm und Zweigen, die wir hier durch mit Sechsecken angenäherte Zylinder darstellen. Als Blatt soll eine einfache Kugel-Näherung dienen.

Unser Baum hat ein paar Parameter: die Verzweigungstiefe, und die Anzahl der Äste. Der oben dargestellte Baum hat auch
noch eine Wahrscheinlichkeit, mit der Äste ausfallen, die wollen wir hier aber nicht implementieren. Fangen wir also mit dem Stumpf an. Zunächst brauchen wir eine untere

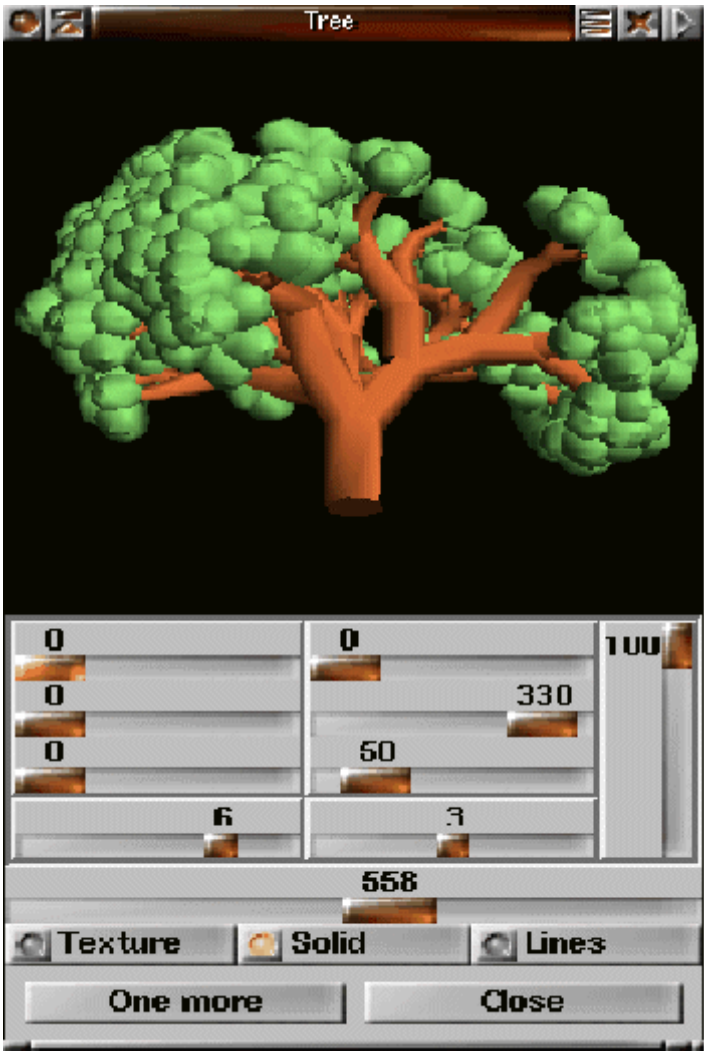


Abb. 2: Der Baum

Begrenzungsfläche, hier erst einmal ein Sechseck. Wir lassen die Turtle, wo sie gerade steht, und öffnen einen Pfad mit sechs Punkten pro Runde.

```
: baum ( m n - )
    .brown .color  6 open-path
```

Die Sechsecke haben einen Winkel von π/3 pro Schritt, den können wir uns schon mal vormerken. Er bestimmt die Schrittweite für die Funktionen, die keinen Winkel als Parameter haben.

```
 pi 3 fm/ set-dphi
```

Nun fangen wir mit sechs Punkten in der Mitte an. Wir müssen zunächst die sechs Punkte hinzufügen (der Pfad ist am Anfang leer), und dann in der nächsten Runde nochmal setzen, um die Normalenvektoren richtig zu setzen (aller Anfang ist schwer -- da die Normalenvektoren sich auf die letzte Runde beziehen, gibt es in der ersten Runde noch gar keine).

```
 6 0 DO  add  LOOP next-round
 6 0 DO  set  LOOP next-round
```

Um sie herum werden in der nächsten Runde die Dreiecke gezeichnet, die das Boden-Sechseck ergeben. Die Größe der Dreiecke ist hier aus der Verzweigungstiefe errechnet, indem die mit 0.03 multipliziert wird. Da OpenGL selbst mit Fließkommazahlen arbeitet, verwendet auch die Turtle-Grafik solche Zahlen.

```
 6 0 DO dup !.03 fm* set-r LOOP next-round
```

Nun kommt ein kleiner Trick, um eine scharfe Kante zu setzen - die 3D-Turtle-Grafik berechnet nämlich die Normalenvektoren an einem Punkt aus der Summe der Kreuzprodukte der Vektoren nach links/hinten und nach rechts/vorne. Ein weiterer Schnitt an derselben Stelle bewirkt, daß nur eine Richtung für den Normalenvektor berücksichtigt wird.

```
 6 0 DO  dup !.03 fm* set-r  LOOP
```

Nun können wir zum eigentlichen rekursiven Teil kommen, den Zweigen:

```
 zweige ;

: zweige ( m n - )  recursive
```

Um eine doppelte Rekursion zu vermeiden, verwende ich eine Schleife für die Endrekursion.

```
 BEGIN  dup  WHILE
```

Auch hier müssen wir erstmal eine neue Runde anfangen. Damit der Baum nicht so plattgepreßt in der Ebene steht, drehen wir ihn pro Verzweigung um 54 Grad.

```
    next-round  pi !.3 f* roll-left
```

Als nächstes müssen wir entsprechend der Verzweigungstiefe vorwärts gehen, und einen neuen Ring zeichnen.

```
    dup !.1 fm* forward
    6 0 DO dup !.03 fm* set-r LOOP
```

Für die weiteren Verzweigungen brauchen wir eine Schleife -- bis auf die letzte Verzweigung, die wird ja von der Endrekursion abgearbeitet.

```
    over 1 ?DO
```

Jeder Ast wird durch Rotieren um die Blickachse gedreht -- I' ist hier das Ende der Schleife. Der Befehl >turtle sichert den aktuellen Status der Turtle auf einem Turtle-Stack, turtle> nimmt ihn wieder herunter. Ich verwende eine lokale Variable, da die Turtle etwas Returnstackplatz braucht, und damit I und I' nicht verfügbar sind. Den Fließkommastack darf man auch nur für Zwischenberechnungen verwenden, da die C-Library von einem leeren Stack ausgeht.

Nach der Drehung müssen wir nach rechts (um 18 Grad hier), und danach die Turtle wieder zurückdrehen - damit die Punkte der jeweiligen Schnitte zusammenpassen. Die geänderte Blickrichtung der Turtle bleibt durch diese Operation erhalten, nur ihre Ausrichtung im Raum wird zurückgesetzt.

```
2pi I I' fm*/ { f: di |
>turtle
   di roll-left pi 5 fm/ right
   di roll-right
   2dup 1- zweige
turtle> }
```

So, nun noch die Schleife fertigmachen

```
LOOP
```

und für die Endrekursion nach rechts kippen (diesmal ist die Drehung 0 Grad).

```
      pi 5 fm/ right
 1- REPEAT
```

Am Schluß noch den Pfad zumachen, und ein Blatt zeichnen.

```
 close-path leaf 2drop ;
```

Das Blatt selbst ist eine einfache angenäherte Kugel:

```
: leaf ( - )
  .green .color
  6 open-path  6 0 DO  add  LOOP
  next-round !.1 forward
     6 0 DO  !.2 set-r  LOOP
  next-round !.2 forward
     6 0 DO  !.2 set-r  LOOP
  next-round !.1 forward
     6 0 DO  !.1 set-r  LOOP
  next-round
     6 0 DO  !0  set-r  LOOP
  close-path .brown .color ;
```

Das sind noch nicht die ganzen Sourcen, wir brauchen noch etwas Overhead, um die Ansicht auf den Baum zu verändern. Die ganzen Sourcen finden sich in der Datei tree.str (3D-Grafik) und tree.m (Benutzeroberfläche).

Ein komplexeres Beispiel: Der Drache

Da der Drache sehr komplex ist, beschreibe ich hier nur die wesentlichen Punkte. In typischer Forth-Tradition wird der Drache vom Schwanz her aufgezäumt.

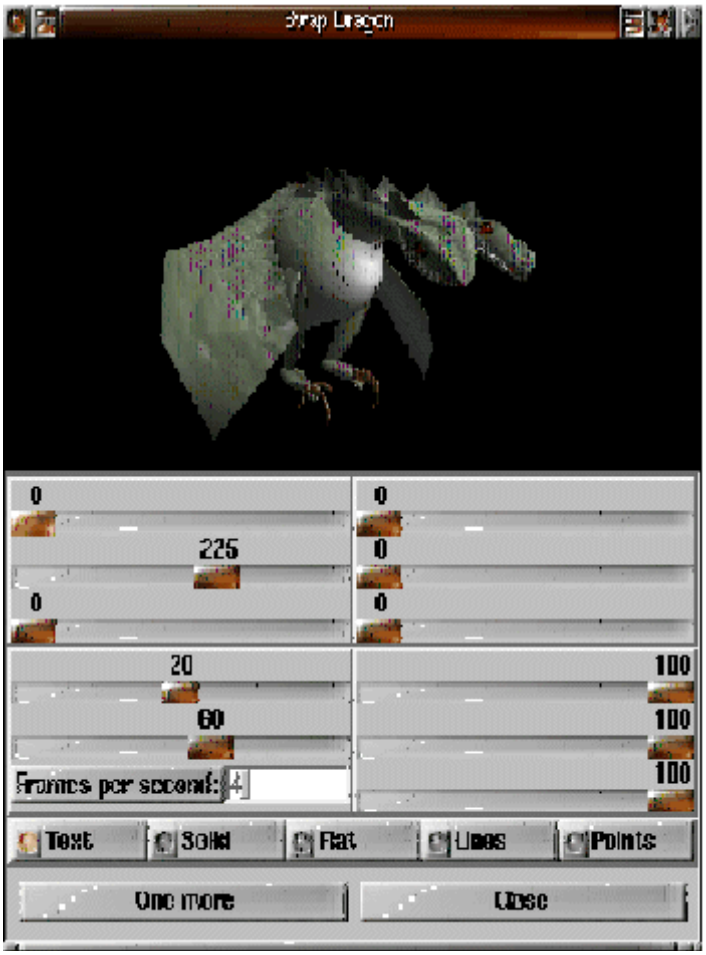


Abb. 3: Der SWAP Drache

Schwanz

Der Drache besteht aus einzelnen Segmenten, die im wesentlichen einen Kreis mit einem Zacken darstellen:

```
: dragon-segment ( ri ro n - )
  { f: ri f: ro | next-round
    ro set-r  1 DO  ri set-r  LOOP
    ro !-0.0001 set-rp !0 phi df! } ;
```

Damit der Schwanz so schön wackelt, und auch um die anderen Bewegungen zu synchronisieren, gibt es einen Timer, der in einen Winkel [0,2π[umgesetzt wird.

Variable tail-time

```
: time' ( - 0..2pi )
  tail-time @ &24 &60 &30 * * um* drop
  0 d>f !$2'-8 pi f* f* ;
```

Das eigentliche Schwanzwackeln wird dann aus der Segmentnummer und der Zeit berechnet -- das Ergebnis ist die Verschiebung nach links bzw. rechts.

```
: tail-wag ( n - f )
  >r pi r@ 1 + fm* !.2 f* time' f+
```

```
  fsin r> 2+ dup * 1+ fm/ !30 f* ;
```

Der Ursprung des Drachens liegt im Bauch, nicht an der Schwanzspitze. Gezeichnet wird der Drache aber von der Schwanzspitze her -- also muß zunächst eine Kompensation berechnet werden, sonst wackelt der Schwanz mit dem Drachen.

```
: tail-compensate ( n - f )  !0
  0 DO  I 2+ tail-wag f+ !1.1 f/  LOOP
  !1.1 !20 f** f* fnegate ;
```

Der eigentliche Schwanz ist damit recht einfach: erstmal zur Schwanzspitze zurück, und einen Punkt als Anfangspolygon setzen. Dann Schritt für Schritt den Schwanz wackeln lassen, ein Stück vorwärts gehen, und ein Drachensegment zeichnen. Jedes zweite Drachensegment hat einen Zacken nach oben, und die Scalierung macht den Schwanz auch immer dicker. Der Radius wird zusätzlich vergrößert. Diese Scalierung muß natürlich zuerst in die andere Richtung vorgenommen werden. Als Texture-Mapping-Funktion wird z,φ verwendet, also Bewegung der Turtle für die eine Texturkoordinate, und der Winkel gegen die Senkrechte für die andere.

```
: dragon-tail ( ri r+ h n - ri h )
  zphi-texture
  { f: ri f: r+ f: h n |
  !1.05 !-20 f**
  !1.1  !-20 f** !1 scale-xyz
  h -&15 fm* &20 tail-compensate
  h -&25 fm* forward-xyz
  n 1+ 0 DO  add  LOOP
  20 0 DO !0  i 2+ tail-wag  h forward-xyz
     pi &90 fm/ up
     ri fdup I 1 and 0= IF  r+ f+ THEN
     n dragon-segment
     !1.05 !1.1 !1 scale-xyz
     !.025 ri f+ to ri
  LOOP  ri r+ h } ;
```

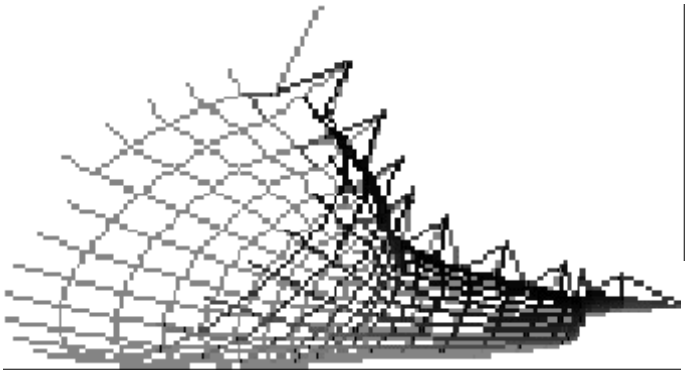


Abb. 4: Schwanz

Körper

Der Körper des Drachens besteht aus genau denselben Segmenten wie der Schwanz, nur statt weiter zu wachsen, muß der Körper sich wieder schließen.

```
: dragon-wamp ( ri r+ h ri+ n - ri' )
  { f: ri f: r+ f: h f: ri+ n |
  8 0 DO  h forward
     ri fdup I 1 and 0= IF  r+ f+ THEN
     n dragon-segment
     ri+ ri f+ to ri !-0.02 ri+ f+ to ri+
  LOOP  ri ri+ !.02 f+ f- } ;
```

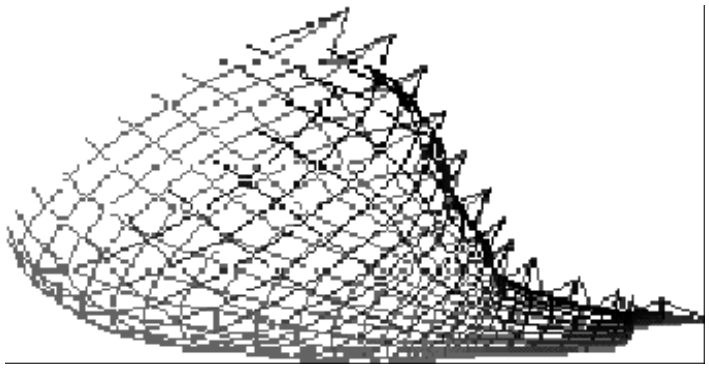


Abb. 5: Körper

Hals

Auch der Hals besteht aus diesen Segmenten; allerdings gibt es hier zwei verschiedene Wachstumsfunktionen, eine für die Schulter (schnelle Größenabnahme), und eine für den eigentlichen Hals (langsame Abnahme). Die Schulter biegt sich nach links, der Hals wieder nach rechts. Entsprechend wird die Funktion dragon-neck-part zweimal aufgerufen.

```
: dragon-neck-part
  ( ri r+ h factor angle n m - ri' )
  swap { f: ri f: r+ f: h f: factor f: angle n |
  0 ?DO  h forward  angle left
     pi &30 fm/
     time' fsin !.01 f* f+ down
     factor ri f* to ri
     ri fdup I 1 and 0= IF  r+ f+ THEN
     n dragon-segment
  LOOP ri } ;
```

```
: dragon-neck ( ri r+ h angle n - )
  { f: r+ f: h f: angle n |
  r+ h !.82 angle
    n 4 dragon-neck-part
  r+ h !.92 angle f2/ fnegate
    n 6 dragon-neck-part
  fdrop close-path } ;
```

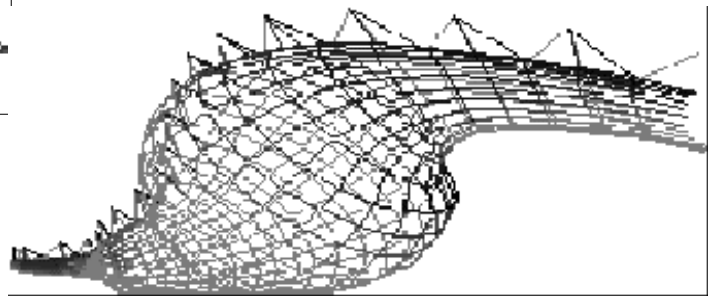


Abb. 6: Hals

Kopf

Der Kopf besteht aus einem abgerundeten Rechteck mit einer Einritzung für die Zähne. Die Funktion ist nicht so einfach zu generieren, deshalb verwende ich ein Array für die Koordinaten, allerdings nur für die linke Hälfte des Kopfs; die rechte wird durch Spiegelung an der Y-Achse gewonnen. Die Größenverhältnisse der Schnitte zueinander entsprechen in etwa dem Bauch. Der Kopf hat eine andere Textur, eine mit Augen, Nasenlöchern und Zähnen.

```
Create head-xy
      !0.28 f>fs , !0.0 f>fs ,
      !0.30 f>fs , !0.5 f>fs ,
      !0.25 f>fs , !0.6 f>fs ,
      !0.05 f>fs , !0.6 f>fs ,
      !0.00 f>fs , !0.5 f>fs ,
      !-.05 f>fs , !0.6 f>fs ,
      !-.10 f>fs , !0.6 f>fs ,
      !-.15 f>fs , !0.5 f>fs ,

: dragon-head ( t1 shade - )  !text
  pi 6 fm/ down  !1.2 !.4 !.4 scale-xyz
  !-.65 forward
  !.5 x-text df!
  16 open-path  16 0 DO  add  LOOP
  6 0 DO
     I 5 = IF    !.25
          ELSE  I 0= IF !0 ELSE !.35 THEN
          THEN forward
     >matrix
     pi !0.1 f* I 2* 5 - fm* fcos
     fdup !.5 f+ !1 scale-xyz
     next-round
     head-xy 16 cells bounds DO
        I sf@ I cell+ sf@ set-xy
        2 cells +LOOP
     head-xy dup 14 cells + DO
        I sf@ I cell+ sf@
        !1'-6 f+ fnegate set-xy
        -2 cells +LOOP
     matrix>
  LOOP
  !1 x-text df!
  close-path ;
```

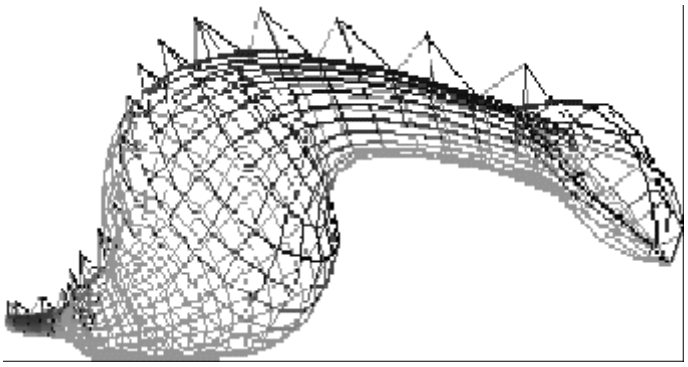


Abb. 7: Kopf

Der zweite Hals und Kopf werden mit entsprechend negierten Winkeln gezeichnet. Ähnlich dem vorherigen Beispiel wird dazu der Status der Turtle gesichert, und vom selben Status erneut ausgegangen.

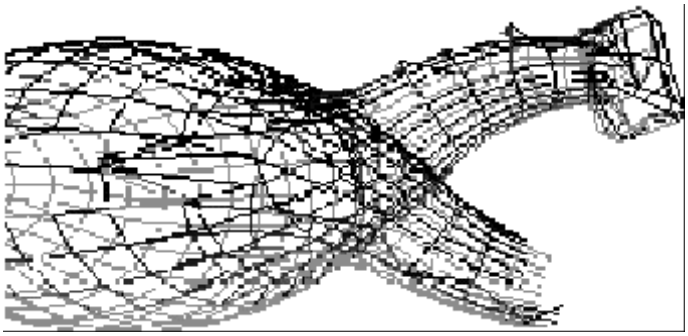


Abb. 8: Zweiter Hals

Flügel

Der Flügel hat ein einfaches, flachgestrecktes Sechseck als Schnitt. Dieses Sechseck sorgt für die Krümmung des Flügels, und wird zur Modellierung der ,,Finger" verwendet.

```
: wing-step { f: f2 f: f3 |
  next-round
  !0 f2 fnegate                 set-xy
  f3 f2/ f2 fnegate              set-xy
  f3 f3 !.125 f*                set-xy
  f3 !.001 f- f3 !.125 f* !.001 f+ set-xy
  f3 f2/ f2                     set-xy
  !0.001 f2 fmin f2               set-xy } ;
```

Die Falte-Funktion des Flügels sorgt für eine Bewegung von Arm/Unterarm und der Finger abhängig von der Zeit für eine Auf/Abwärtsbewegung des Flügels. f2 ist ein additioneller Term zum Cosinus, f1 ein multiplikativer.

```
: wing-fold ( f1 f2 - )
  time pi 5 fm/ f- fcos f+ f* down ;
```

Die Bewegung und der Aufbau des Flügels sind kompliziert; deshalb erkläre ich nicht alle Einzelheiten. Auch hier wird zunächst ein Pfad geöffnet. Danach werden schrittweise Flügelansatz, Ober- und Unterarm, und zuletzt die drei Finger gezeichnet.

```
: wing ( - )
  8 open-path !.9 scale
  6 0 DO  add  LOOP
  !.02 !1.2 wing-step !.3 forwardAnsatz
  pi &10 fm/ down  pi &8 fm/ roll-left
  time' fsin !1.3 f* !.2 f+ right
  !.02 !1 wing-stepOberarm
  pi 5 fm/ up  pi &10 fm/ right  !1 forward
  pi 5 fm/ down  pi &20 fm/ left
  time' fcos !-.25 f* !.5 f- roll-left
  time' fcos pi 6 fm/ f* down
  !.02 !1 wing-stepUnterarm
```

```
time' !1 f- fcos !1 f+ pi 8 fm/ f* right
pi -3 fm/ !-1.0 wing-fold
pi &10 fm/ left !1 forward
pi 4 fm/ !-1.5 wing-fold
!.02 !2 wing-step
2 0 DO  !.025 forward
      pi &12 fm/ !1.2 wing-fold
      pi &10 fm/ right !.05 forward
      !.02 !2 wing-stepFinger
LOOP
!0 !2 wing-stepAbschluß
close-path ;
```

Der eigentliche Flügel wird für rechts und links grundsätzlich gleich gezeichnet. Die Symmetrie wird durch eine Spiegelung an der Y-Achse erreicht. Hier muß noch ein Wort zu OpenGL gesagt werden: Nur die Vorderseiten der Dreiecke werden tatsächlich gezeichnet. Durch so eine Spiegelung werden aber aus allen Vorderseiten „Rückseiten``, weil sich die Umlaufrichtung ändert. Also muß man das OpenGL mitteilen, und das macht flip-clock.

```
: right-wing ( h - )
  pi/4 roll-right pi/2 right
  !2 f*  forward pi !.3 f* roll-left
  zp-texture !.13 y-text df! wing ;

: left-wing ( h - ) !1 !-1 !1 scale-xyz
  flip-clock right-wing flip-clock ;
```

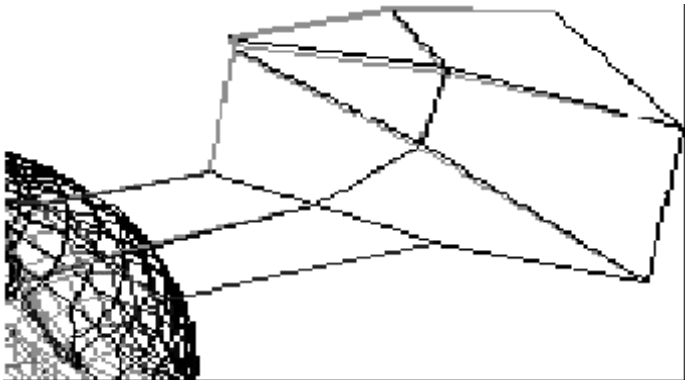


Abb. 9: Flügel

Der ganze Drache

Die Beine lasse ich hier mal weg, sie sind nicht so interessant, da sie aus statischen Teilen bestehen (im Wesentlichen langestreckte Ellipsoide und bogenförmige Klauen). Kommen wir zum Hauptprogramm:

Zunächst wackelt der Drache bei jedem Flügelschlag etwas auf und ab. Dann muß für die Drachensegmente noch der Winkel gesetzt werden.

```
: dragon-body
  ( t0 s t3 s t1 s t3 s t2 s n - ) >r
  time' fsin !.1 f* !0 !0 forward-xyz
  pi f2* r@ fm/ set-dphi
  r@ 1+ open-path
```

Zuerst wird, wie gesagt, der Schwanz gezeichnet.

```
  !.1 !.3 !.2 r@ dragon-tail
```

Die Rückgabeparameter des Schwanz werden auch für den Bauch weiterverwendet.

```
  r> { f: ri f: r+ f: h n |
  ri r+ h !.06 n dragon-wamp fdrop
```

Hals und Kopf werden jeweils für rechts und links aufsetzend von derselben Stelle gezeichnet; jeweils mit negiertem Winkelparameter.

```
  >turtle
    ri r+ h  !10 grad>rad n dragon-neck
    2dup dragon-head  2swap !text
  turtle>  >matrix
    ri r+ h !-10 grad>rad n dragon-neck
    dragon-head  2drop
  matrix>
```

Danach muß die Textur geändert werden, und die beiden Flügel werden gezeichnet.

```
  2dup !text
  h !2 f* forward
  >turtle  h right-wing   turtle>
  >turtle  h left-wing   turtle>
```

Analog werden auch die Füße gezeichnet.

```
  h !-6 f* forward
  >turtle  right-leg   turtle>
  >turtle  left-leg    turtle>
  2drop 2drop } ;
```

Ausblick

Was kann man damit machen, was fehlt noch? Als seriöse Anwendung kann natürlich die Darstellung von dreidimensionalen Daten gelten. „Unseriösere" Anwendungen wären etwa
Computerspiele. Dafür braucht man dann Kollisionserkennung, und wahrscheinlich ein hierarchisches Modell, um Räume und bewegte/bewegbare Objekte einzuordnen. Auch unterschiedliche Levels of Detail abhängig von der Größe des Objekts am Bildschirm müssen jetzt noch mühsam von Hand programmiert werden. Ob es hier für animierte Objekte überhaupt eine andere Möglichkeit gibt, ist mir nicht klar.

Die Behandlung von verschiedenen Texturen ist im Moment noch zu aufwendig; sie müssen auf dem Stack herumgeschleppt werden. Hier muß auch das 3D-Turtle-Objekt selbst noch mehr Tools zur Verfügung stellen.

Und wie immer macht Windows Schwierigkeiten. Obwohl man nicht behaupten kann, daß die MESA-Bibliothek unter Linux fehlerfrei ist, so implementiert sie zumindest alle Features von OpenGL 1.2. Die Windows 95-OpenGL-Library von Mikrosoft läßt Texturen gleich ganz weg, und funktioniert auch sonst recht wenig zuverlässig. Der Drache wird jedenfalls schwarz auf schwarz dargestellt. Da Silicon Graphics ihre GLX-Sourcen freigegeben haben, sind die verbleibenden Linux-Probleme und die fehlende Hardwareunterstützung (nur 3Dfx wird unterstützt) wahrscheinlich in Kürze ausgeräumt.

Das Ganze kann man downloaden unter
http://www.jwdt.com/~paysan/bigforth.html

Anhang: Befehle der 3D-Turtle-Grafik

Navigation

left	(f -) turns the turtle's head left
right	(f -) turns the turtle's head right
up	(f -) turns the turtle's head up
down	(f -) turns the turtle's head down
roll-left	(f -) rolls the turtle's head left
roll-right	(f -) rolls the turtle's head right
x-left	(f -) rotate the turtle left around the x axis
x-right	(f -) rotate the turtle right around the x axis
y-left	(f -) rotate the turtle left around the y axis
y-right	(f -) rotate the turtle right around the y axis
z-left	(f -) rotate the turtle left around the z axis
z-right	(f -) rotate the turtle right around the z axis
forward	(f -) move the turtle in z direction
forward-xyz	(fx fy fz -) move the turtle
degrees	(f -) steps per circle. Common cases: 2π for radians (default), 360 for deg, 64 for asian degrees, or whatever you find suits your application best.
scale	(f -) scales the turtle's step width by the factor f
scale-xyz	(fx fy fz -) scale the turtle's step width in x , y , and z direction
flip-clock	(-) change default coordinate from left hand to right or the other way round. Use that after scale-xyz with an odd number of negative scale factors.

Turtle state

> matrix	(-) push turtle matrix on the matrix stack
matrix >	(-) pop turtle matrix from the matrix stack
matrix@	(-) copy turtle matrix from the stack
1matrix	(-) initialize turtle state with the identity matrix
matrix*	(-) multiply current transformation matrix with the one on the top of the matrix stack (and pop that one)
clone	(- o) create a clone of the turtle
> turtle	(-) clone the turtle and use it as current object
turtle >	(-) destroy current turtle and pop previuos incarnation

Pathes

open-path	(n -) opens a path with n points in the first round
close-path	(-) closes a path and performs the final rendering action
next-round	(-) closes a round and opens the next one
open-round	(n -) opens a round with n points (obsolete)
close-round	(-) closes a round (by copying the first point as last point) and performs the per-round rendering action (obsolete)
finish-round	(-) performs the per-round rendering action without closing the round first (this is for open objects) (obsolete)
add-xyz	(fx fy fz -) adds the point at the x,y,z -coordinates relative to the turtle. x is up from the turtle, y right, z before. The point is connected to the same point of the previous round as the point before.
set-xyz	(fx fy fz -) sets a point with x,y,z - coordinates. The point is connected to the next point of the previous round as the point before.
drop-point	(-) skips one point, set-xyz is equal to add-xyz drop-point
set-rpz	(fr fphi fz -) set with cylinder coordinates
set-xy	(fx fy -) set-xyz with z=0
set-rp	(fr fphi -) set with cylinder coordinates, z=0
set-r	(fr -) set with cylinder coordinates, z=0 , φ=φcur , φ cur=φ cur+Δφ
set	(-) set at current turtle location
add-rpz	(fr fphi fz -) add with cylinder coordinates
add-xy	(fx fy -) add-xyz with z=0
add-rp	(fr fphi -) add with cylinder coordinates, z=0
add-r	(fr -) add with cylinder coordinates, z=0 , φ=φcur , φ cur=φ cur+Δφ
add	(-) add at current turtle location
set-dphi	(fdphi -) sets Δφ

Drawing Modes

points	(-) draw only vertex points
lines	(-) draw a wire frame
triangles	(-) draw solid triangles
textured	(-) draw textured triangles
smooth	(-) variable: set on for smooth normals when rendering textured, set off for

	non-smooth rendering
xy-texture	(-) texture mapping based on x and y coordinates
zphi-texture	(-) texture mapping based on z and φ coordinates
rphi-texture	(-) texture mapping based on r and φ coordinates
zp-texture	(-) texture mapping based on z and the point number coordinates
load-texture	(addr u - t) loads a ppm file with the name addr u and returns the texture index t
set-light	(par1..4 par n -) Set light source n

Bernd Paysan

Gehaltvolles

zusammengestellt und übertragen
von Fred Behringer

FORTHWRITE der FIG UK, Großbritannien

Nr. 102, Juni 1999

1 Editorial

Chris Jakeman begrüßt 4 neue Mitglieder, darunter Heiko Gross "from Germany". Wir dürfen uns dem Gruß anschließen. Haben wir Sie auch in unserer Liste? Bei uns kostet es (für Studenten und Pensionäre) auch nicht mehr. Unsere englischen Forth-Freunde haben, wie Chris verkündet, Mitglieder aus 7 anderen Ländern. Da müssen wir uns aber ranhalten (?) - Die Forthwrite trägt jetzt eine ISSN-Nummer. Damit ist sie in der British Library (und folglich in allen Bibliotheken der Welt (?)) offiziell bekannt.

2 Forth News
Chris Jakeman

Unsere neue Web-Site wird bekanntgegeben und unsere Absicht, alles auch ins Englische zu übertragen. Weitere Stichworte: Schwer zu findende Forth-Bücher; Triangle Forth-Hardware seit 18 Jahren; russische FIG: www.forth.org.ru/ (!!!); Forth IRC chat room; kForth jetzt mit Datenprüfstack; Vorschlag von Philip Preston (FIG UK) zur Compilation aus dem Interpreter-Modus heraus - Gegenargumente von Elizabeth Rather (Unternehmerin, FIG US und ANS-Komitee); Cross-Compiler-Erweiterung von ANS; ShBoom im Internet; Entwicklung einer kleinen Forth-CPU - offen für jeden potentiellen Teilnehmer; TpForth 2.5 ; Forth-Wettbewerb bei Ultra Technology; Bernd Paysan und der geflügelte Drachen in MINOS; 4th; Win32Forth v4.1; pForth für Psion 5; DELTA Forth - freies Forth auf Java; ein paar anerkennende Zeilen über den Drachenpreis bei uns und seinen diesjährigen Träger; Ankündigung, daß Charles Moore von jetzt ab sein bisheriges Schweigen in comp.lang.forth bricht (ist wohl doch nicht jedermanns Sache); Expertenhilfe für LMI-Meacompiler von FIG US gesucht; Forth im Atommeiler auf Siemens AS990; Nachruf auf Robert Reiling, ehemaliger Präsident von FIG US und Hauptorganisator der FORML-Konferenzen.

5 Handling Literals
Jack Brien

Mal ehrlich, haben Sie gewußt, daß die englischen "literals", zumindest in den Druckfahnen, einfach nur Druckfehler sind? Wörter ohne erkennbare Bedeutung, die man nicht einordnen kann. Das ist das sprachliche Umfeld, das uns fehlt, wenn wir Forth-Worte wie LITERAL antreffen. Zudem liefert mir das Wörterbuch auf Anfrage nur "literal error". Wie soll man darauf kommen, daß das den Begriff lediglich näher erläuternde Eigenschaftswort wichtiger ist als das den Begriff eigentlich festlegende Hauptwort? Das ist eben unser Problem. Da nützt die ganze Schulbildung nichts. Das schaffen wir nie! (Noch dazu in einer Umgebung, die zu "mobile phone" Handy sagt und das für Englisch hält.) - Jack Brien greift die genannte Bedeutung von LITERAL auf. Forth hat einen ausgefeilten Literal-Handler (für Worte, die nicht mehr im Forth-Dictionary zu finden sind, aber vielleicht doch noch eine Bedeutung (im Sinne von Forth) haben können (Ganzzahl, einfach genau, doppelt genau, Konstante ...) und dann vom Interpreter entsprechend vrarbeitet werden sollen. Jack macht Vorschläge, wie man den üblichen Literal-Handler noch ausgefeilter gestalten kann (auch noch Sonderdaten und vorgegebene wichtige Strings aussondieren und zur Verarbeitung weiterleiten, wenn sie auftreten ...). Auf der dritten Seite des Artikels ein von Bernd Paysan übernommener Trick mit POSTPONE und LITERAL .

9 Nominations for the FIG UK Awards

Vorgeschlagene Kandidaten: Philip Preston oder Alan Wenham für wesentliche Forth-Errungenschaften, Paul Bennett oder Ray Allwright für wichtige Forthwrite-Beiträge. Jetzt ist die Jury dran. Das nächste Mal dann die gekürten Preisträger. Ich kenne die WebForth-Arbeiten von Philip Preston einigermaßen. Sie sind gut. Aber die Besprechungsleistungen von Alan Wenham sind für uns (in der FG) von größter Wichtigkeit. Ich würde ihm den Preis, für den er vorgeschlagen wurde, wünschen.

10 Forth for Virtual Reality
Joe Anderson

"Was heißt 'virtuelle Realität'?", fragt der Autor (der Rezensent: 'Scheinwelt' heißt es bestimmt nicht) und gibt sich gleich die Antwort: "Dinge, die bis in Kleinigkeiten hinein realistisch anmuten und in Echtzeit ablaufen." "Was brauchen wir dazu?" "Eine Sprache, die ..." usw. ... mit einem Wort, "Forth." Joe zeigt, wie es gemacht werden kann. - Aber mehr noch. Er weist begeistert auf eine Webseite von Marc de Groot hin: www. immersive.com. Marc verwendet Meme (Multi-platform Extensible Messaging Environment). Joe meint, wahrscheinlich in Anlehung an die von Richard Dawkins gebrauchte Bezeichnung 'Meme' für eine Speichereinheit in dessen Gen-Modell. Marcs Meme baut auf das in C geschriebene Forth C-Forth83 von Mitch Bradley auf. Ähnlich wie Java hat Meme aus Portabilitätsgründen eine virtuelle Maschine zur Grundlage. - "Virtuelle Welten nur für Peng-Peng-Spiele? Vielleicht fallen einem Atommeiler ein... Man denke aber ruhig einmal auch an die Chirurgie. Kleine Roboter in die Venen. Auf der Suche nach dem Kankheitsherd...." , soweit der Autor des Artikels. Interessant. Ich werde mir das von ihm gepriesene Demo von Marc de Groot holen.

15 Web Forth Project
Chris Jakeman

Bericht über den bisherigen Hergang, den augenblicklichen Stand und die geplante Zukunft unseres internationalen (4 Länder, 3 Konti-

nente) Vorhabens: Forth mit Schwerpunkt "interaktive Einführung" auf Java im Internet. Mehr wird wieder in comp.lang.forth zu lesen sein, wo auch um Kommentare und Verbesserungsvorschläge gebeten werden wird. Unsere deutschen Netsurfer können selbstverständlich alle Englisch. Viele halten es daher auch nicht für nötig, sich mal den deutschen Teil anzusehen. Aber gerade hier wären mir (dem Rezensenten) Kommentare höchst willkommen. Man möchte nicht gern ins Blaue hinein arbeiten. Schließlich ist ja eine deutsche Fassung, wenn sie nicht gänzlich vermurkst wird, für den Anfänger doch schneller zu überblicken als eine englische. Wenn das nicht so wäre, bräuchte sich Thomas Beierlein auch nicht immer wieder so viel Mühe mit der Übersetzung von "Henry's Reports" zu machen.

17 Dutch Forth Users Group

Wieder eine Mitgliederwerbeanzeige der holländischen Forth-Freunde. Das nächste Mal sind wir dann wieder dran. So war es jedenfalls ausgemacht.

18 1xForth
Charles Moore

Der Titel nimmt darauf Bezug, daß Charles Moore der Meinung ist, die meisten Forth-Programmierer schreiben 10x mehr Code als nötig wäre. "Schreiben Sie 1xForth, nicht 10xForth" (Einfachforth, nicht Zehnfachforth). Dieses Interview mit Charles (Chuck) Moore wurde von Jeff Fox aufgenommen. Jeff hat die letzten 8 Jahre mit Chuck zusammengearbeitet. 7 Seiten. Originalton Chuck: "Mit Forth zu arbeiten, macht Riesenspaß, bei der Hardware ist der Spaß nicht ganz so groß." ... "Ich bin überzeugt davon, daß es da einen optimalen Grad an Komplexität gibt, den das Gehirn gerade noch verarbeiten kann. Macht man es zu einfach, langweilen sich die Leute, macht man es zu kompliziert, kommen sie nicht mehr mit." ... "Für den i21 verwende ich Color Forth." ... "Ein Forth-Wort sollte mit einem oder zwei Argumenten auskommen. Die Stacktiefe sollte nie mehr als drei oder vier betragen." Und so geht es dann weiter. Im wesentlichen wird Color Forth für den i21 beschrieben. ... "Statt eines Colons (:) werden die Namen von Colon-Dfinitionen rot eingefärbt, die aufzurufenden Worte in der Mitte der Definition grün usw." ... "ELSE habe ich aus IF ELSE THEN ganz verbannt. Ich verwende stattdessen IF - - - ; THEN - - -" ... "Festplatten braucht man überhaupt nicht mehr. Mit Megabytes an Speichern kann man es sich leisten, sämtliche Daten ins RAM zu laden und von dort aus abzuarbeiten." ... "Ich bin über die Statistiken nicht genau informiert, aber ich möchte mal annehmen, daß die meisten Computer überhaupt keine Berechnungen anstellen. Sie schieben Bytes hin und her." ... "Und ich lasse mich nicht davon abbringen: Lokale Variablen sind nicht nur unnütz, sie sind schädlich." Der Rezensent: Eine wahre Freude, diese Interviews mit Charles Moore. Aus denen lerne ich mehr als aus noch so vielen Artikeln.

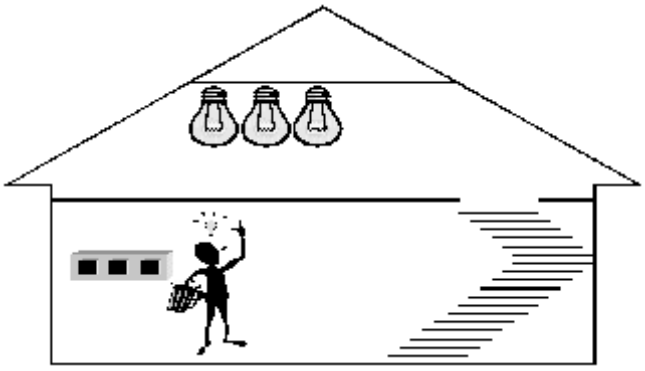
26 Skeletons - Designing a Recursive Application
Graham Telfer

"In Prolog kennt man die Methode 'Skeleton and Enhancement'." Ich übersetze das nicht, da ich weder von Prolog noch von der genannten Methode genügend Ahnung habe. Es geht um Rekursion. Und es sind immer wieder dieselben Leute, die in den Forth-Gruppen schreibenderweise auftreten. Der Autor, Graham Telfer, arbeitet im WebForth-Projekt mit und ist mir von da her wohlbekannt. Er ist, soweit ich mich erinnere, Englischlehrer irgendwo in Japan. Die Welt ist klein geworden. Und so schnell überbrückbar. Graham greift auf Ideen von David (Dave) Pochin aus der Forthwrite 100 zurück. Auch ein Mitglied des WebForth-Teams. Um was geht es im vorliegenden Artikel? Ein "Warenkorb" (ein Begriff aus der Wirtschaftswissenschaft). An der Kasse soll ganz schnell der Summenbetrag bestimmt werden. Graham bemüht sich, das Wesen der Rekursion (überhaupt und am Beispiel) zu erklären. Er gibt als Quelle seiner Überlegungen eine amerikanische Fachzeitschrift aus der Mathematik an. Erstaunlich, was man als Forth-Enthusiast für Energie aufbringen kann. Meine Hochachtung! Das Prinzip: "Egal, ob man weiß oder nicht weiß, auf welchem Weg man die ersten n Stufen des Prozesses bewältigt soll, der Übergang von n zu n+1 ist genau geregelt." Der Rezensent: An die Dynamische Optimierung von Richard Bellman und deren Optimalitätsprinzip wird dabei nicht gedacht? An das Beweisprinzip der Vollständigen Induktion auch nicht? ... Der Artikel regt mich zum Nachdenken an. Drum ist er gut.

33 From the 'Net - Various
Chris Jakeman

8 Seiten trickreiche Programmvorschläge aus dem Internet. Einiges recht interessant - anderes: naja, da kann man sich so seine Gedanken machen. Eine "geniale" Definition von Charles Moore für MAX und MIN. Die geht doch gar nicht! Das weiß man. Weder für vorzeichenbehaftete noch für vorzeichenlose Ganzzahlen! Da stimmt doch was nicht. Ohne Zusatzerklärung ist das einfach falsch.

Schlampiger Elektriker



Schlampige Elektriker soll es hin und wieder geben - hier jedenfalls hat dieser Handwerker ‘vergessen’, die drei Lichtschalter im Erdgeschoß des Hauses zu kennzeichnen. Der Hausherr weiß nicht, welcher Schalter zu welcher Glühlampe im Obergeschoß gehört.

Als ausgemachter Faulpelz (a*uch nicht besser als sein Elektriker* !) will er höchstens einmal in das Obergeschoß steigen, um festzustellen, welche Lampe zu welchem Schalter gehört. Der große Krug Bier in seiner rechten Hand, wird ihm bei der Lösung dieser Aufgabe behilflich sein. Dabei kommt dem Bier keine andere Aufgabe zu, als diejenige, die sich der Braumeister gedacht hat - es wird einfach nur getrunken. Es ist nicht einmal notwendig zur Lösung, aber durchaus hilfreich ! Und selbstverständlich läßt sich der Krug nicht als Spiegel mißbrauchen !

Nur einmal nachsehen dürfen - und trotzdem ‘durchblicken’ ?

Wie soll das gehen ? Wenn Sie die Lösung haben, schicken Sie diese doch bitte an die Redaktion !!!

Forth-Gruppen regional

Moers
Friederich Prinz
Tel.: 02841-58398 (p) (Q)
(Bitte den Anrufbeantworter nutzen !)
(Besucher: Bitte anmelden !)
Treffen: (fast) jeden Samstag,
14:00 Uhr, **MALZ, Donaustraße 1**
47443 Moers

Mannheim
Thomas Prinz
Tel.: 06271-2830 (p)
Ewald Rieger
Tel.: 06239-8632 (p)
Treffen: jeden 1. Mittwoch im Monat
Vereinslokal Segelverein Mannheim e.V.
Flugplatz Mannheim-Neuostheim

München
Jens Wilke
Tel.: 089-89 76 890
Treffen: jeden 4. Mittwoch im
Monat, **China Restaurant XIANG**
Morungerstraße 8
München-Parsing

µP-Controller Verleih

Thomas Prinz
Tel.: 06271-2830 (p)
micro@forth-ev.de

Gruppengründungen, Kontakte

Fachbezogen 8051 ... (Forth statt Basic, e-Forth)
Thomas Prinz
Tel.: 06271-2830 (p)

Forth-Hilfe für Ratsuchende

Forth allgemein
Jörg Plewe
Tel.: 0208-49 70 68 (p)

Jörg Staben
Tel.: 02103-24 06 09 (p)

Karl Schroer
Tel.: 02845-2 89 51 (p)

Spezielle Fachgebiete

Arbeitsgruppe MARC4	Rafael Deliano Tel./Fax: 089-841 83 17 (p)
FORTHchips (FRP 1600, RTX, Novix)	Klaus Schleisiek-Kern Tel.: 040-375 008 03 (g)
F-PC & TCOM, Asyst, (Meßtechnik), embedded Controller, (H8/5xx// TDS2020, TDS8092) Fuzzy	Arndt Klingelnberg, Consultants akg@forth-ev.de Tel.: 02404-6 16 48 (p) (g) (Q)
KI, Object Oriented Forth, Sicherheitskritische Systeme	Ulrich Hoffmann Tel.: 04351 -712 217 (p) Fax: -712 216
Forth-Vertrieb	volksFORTH / ultraFORTH RTX / FG / Super8 / KK-FORTH Ingenieurbüro Klaus Kohl Tel.: 08233-3 05 24 (p) Fax : 08233-99 71 mailorder@forth-ev.de
Forth-Mailbox (KBBS)	0431-533 98 98 (8 N 1) Sysop Holger Petersen hp@kbbs.org Tel.: 0431-533 98 96 (p) bis 22:00 Fax : 0431-533 98 97 Helsinkistraße 52 24109 Kiel

Möchten Sie gerne in Ihrer Umgebung eine lokale Forthgruppe gründen, oder einfach nur regelmäßige Treffen initiieren ? Oder können Sie sich vorstellen, ratsuchenden Forthern zu Forth (oder anderen Themen) Hilfestellung zu leisten ? Möchten Sie gerne Kontakte knüpfen, die über die VD und das jährliche Mitgliedertreffen hinausgehen ?

Schreiben Sie einfach der VD - oder rufen Sie an - oder schikken Sie uns eine E-Mail !

Hinweise zu den Angaben nach den Telefonnummern:

Q = Anrufbeantworter
p = privat, außerhalb typischer Arbeistzeiten
g = geschäftlich

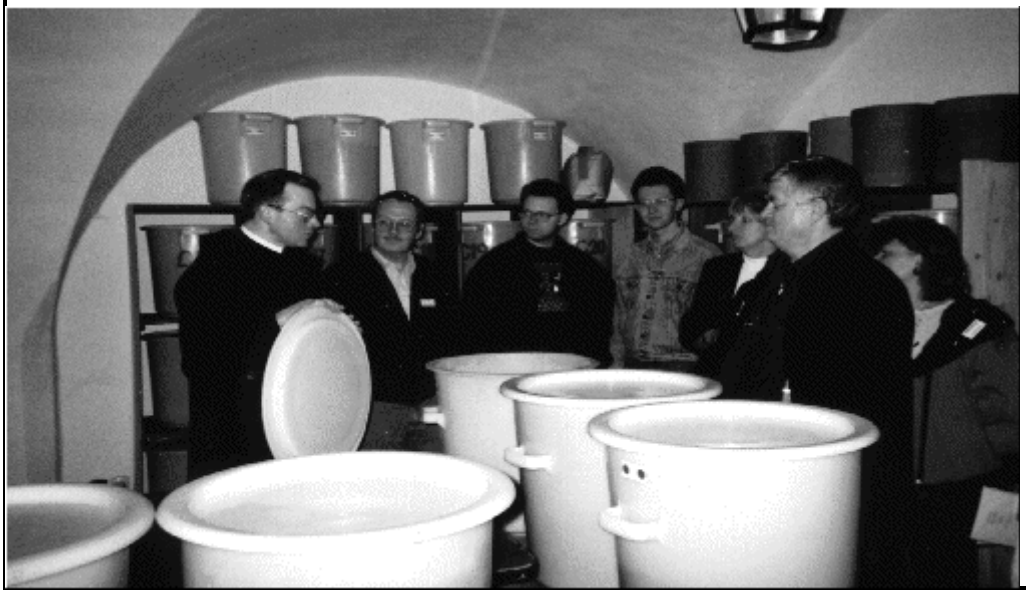
Die Adressen des Büros der Forthgesellschaft und der VD finden Sie im Impressum des Heftes.

Jahrestagung 1999
Impressionen aus Oberammergau

Das obligatorsiche Abschlußfoto

J. Wilke; E. & U. Woitzel; F.Prinz auf dem Gipfel des Kofels

Wolfgang Allinger probiert die Drogen des Klosters



Im Drogenraum des Klosters Ettal stehen nicht nur Zutaten zur Herstellung von Likören bereit.

Trotzdem war das Interesse groß.