Zeitschrift: Visionen : Magazin des Vereins der Informatik Studierenden an der ETH Zürich

Herausgeber: Verein der Informatik Studierenden an der ETH Zürich

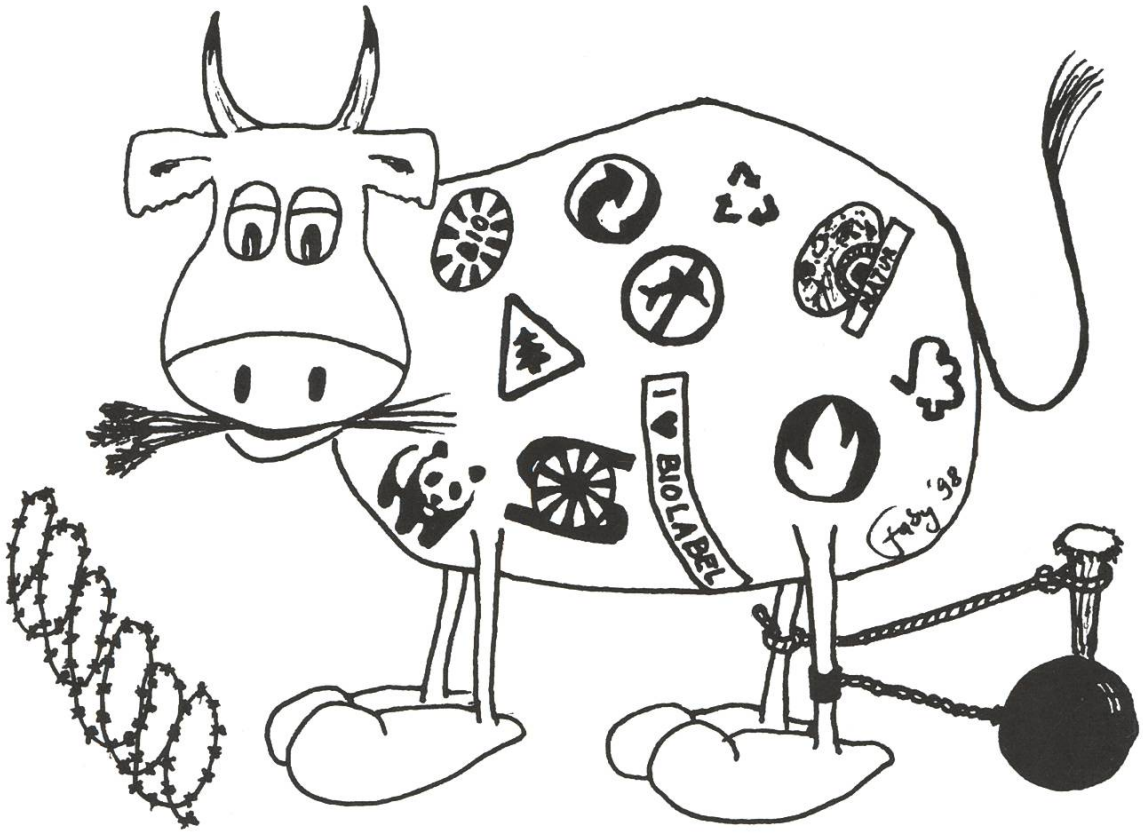
Band: - (1998)

Heft: 5

Heft

Visionen

5.98
Juni



Labelinth in den Läden

DirectX-, Direct3D Programmierung
Pretty Good Privacy
Python - die High-Level-Skriptsprache
Be OS

ZIP Handling

Visionen Herausgegeben vom Verein der Informatikstudierenden der ETH Zürich

Impressum

«Visionen»

Magazin des Vereins der Informatikstudierenden an der ETH Zürich (VIS)

Erscheingsweise: 9x jährlich

Auflage: 1250

Titelbild: Faby Honegger

Jahresabonnement: CHF 25.–

Mitarbeiter an dieser Ausgabe:

Michael Baumer, Roland Brand, Beat Christen, Christian Fritz, Kai Jauslin, Nicky Kern, Adam Moravanszky, Michael Psarros, Pascal Wittmann

Konzept & Realisation:

Stephan Würmlin

Anschrift Verlag & Redaktion:

Verein der Informatikstudierenden (VIS)
ETH Zentrum, IFW B29
8092 Zürich

Tel.: 01/632 72 12 (zu Präsenzzeiten)
Fax: 01/632 11 72

Präsenzzeiten: Mo–Fr, 12.15–13.00
e-mail: vis@iiic.ethz.ch
http://www.vis.inf.ethz.ch/Visionen/

Postkonto: 80-32779-3

Inserate:

1/1 Seite, schwarz/weiss	CHF 500.–
1/1 Seite, s/w + 1 Farbe	CHF 750.–
1/2 Seite, schwarz/weiss	CHF 250.–

Andere Formate auf Anfrage.

Druck:

Kaspar Schnelldruck AG
Birkenweg 2, 8304 Wallisellen

44 Seiten Visionen

Das ist Rekord.
Das letzte Mal hatten wir das irgendwann im 92 oder 93...

Dabei hatte alles so schlecht angefangen. Durch das Organisieren der Kinonacht mit Nicky kamen die Visionen über lange Zeit zu kurz. Das führte schon fast dazu, dass ich einen ausserordentliche Doppelnummer machen wollte.
Doch dann: Auf einmal erhielt ich Berichte en masse, und jetzt (Dienstag 26.5.) kurz vor der Cinenight hatte ich sogar noch Zeit die Visionen zu layouten.
Ich glaube, diese Visionen sind etwas besonderes. Warum ?
Ihr könnt den Bericht eines Agronomiestudenten lesen, der euch etwas über **gesunde Ernährung** erzählt (ab Seite 5).
Die Betriebssystemserie wurde unerwartet doch noch fortgesetzt, dieses Mal mit einem Bericht über **Be OS**... (ab Seite 13, ich warte immer noch auf den Bericht über *OBERON*...).
In dieser Ausgabe könnt ihr auch etwas lernen:
Ihr lest etwas über die **DirectX SDK** und wie man die programmiert (ab Seite 32) , dazu bekommt ihr die Vorzüge der Scriptsprache **Python** aufgezeigt (ab Seite 17).
Ausserdem erfahrt ihr noch etwas über **PGP** und wann die nächste **Signing Session** stattfindet (ab Seite 21)

Viel Vergnügen

Stephan Würmlin

Vorwärts ins Mittelalter

Zur Genschutz-Initiative, über welche dieser Tage abgestimmt wird, ist schon vieles gesagt und vor allem geschrieben worden. „Über den bevorzugten Ausgang der Abstimmung kann man geteilter Meinung sein.

Es ist jedoch offensichtlich, welcher Meinung die ETH und ihre Forscher sind. Die Demo in der Zürcher Innenstadt war wohl das erste mal seit Jahrzehnten, dass sich Akademiker wieder um richtige Politik kümmerten. Auf der anderen Seite steht „das Volk", bei welchem Umfragen zufolge die Initiative eine hohe Akzeptanz hat. Eine Spaltung zwischen Wissenschaft und Normalbürgern ist klar erkennbar.

Wie konnte es soweit kommen? Darauf gibt es viele Antworten. Vier davon sind wohl am Wichtigsten.

Erstens wollen einzelne Initianten der „bösen Industrie" an den Karren fahren. Dies ist teilweise verständlich, denn anonyme Grosskonzerne nehmen tatsächlich selten Rücksicht auf andere als ihre eigenen Wünsche. Andererseits ist die Zeit, als Geldverdienen an sich eine Bösartigkeit sondergleichen war, eigentlich schon ein Weilchen vorbei. Schade ist nur, dass es z.B. einer sehr grossen Chemiefirma in Basel ziemlich egal sein dürfte, wie am 7. Juni abgestimmt wird, schliesslich liegt Frankreich nahe.

Zweitens ist vielen Konsumenten die rigorose Einführung von „Gentech-Food" sauer aufgestossen. Schliesslich nützt die schönste Deklarationsplicht nichts, wenn nur noch genmanipulierte Nahrung verkauft wird. Leider klammert die Initiative diesem Bereich völlig aus.

Drittens hat eine nicht zu unterschätzende Anzahl von BefürwortenAngst vor einem „Dr. Frankenstein", welcher lizensierbare Kinder kreiert. Davon abgesehen, dass solche Leute einmal die Bundesverfassung lesen sollten, welche solches verbietet, handelt es sich hier um eine gefährliche Argumentation. Die Genforschung wird als geradzu „blasphemisch" angesehen. Wer Gene manipuliere, behaupte damit „Gott" habe schlechte Arbeit geleistet. Dazu kann ich nur sagen: Der Eintritt ins Mittelalter ist geschafft. Es hat doch nichts mit Atheismus zu tun, wenn man versucht Erkentnis „über natürliche Vorgänge zu erlangen. Auch eine Genmanipulation, welche z.B. eine Krankheit heilt ist wohl kaum eine Gotteslästerung. Ansonsten könnte man auch darüber diskutieren, ob die Entfernung des Blinddarms zulässig ist (Gott wollte offenbar, dass er da ist). Ich glaube es wird klar, was ich von solcher Gesinnung halte, dennoch: Ich habe auch Verständnis für sie. Wenn die Menschen nicht einmal Ansatzweise begreifen, wie Technik und Forschung funktioniert, ist es klar, dass auf Schlagworte zurückgegriffen wird. Der Mensch hat normalerweise Angst vor Veränderungen. Wenn ihm dann noch nicht mal klar ist, was sich überhaupt „ändert, muss sich niemand wundern, wenn er panisch reagiert. Diese fehlende Information leitet direkt zum vierten Punkt über:

Auch die ETH ist an der Genschutzinitiative nicht ganz unschuldig. Im interantionel Umfeld sieht sie sich gerne als Eliteschule. Es werden „globale" Strategien entworfen, es werden „Benchmarks" durchgeführt, Allianzen werden geschmiedet und so weiter und so fort. Kurzum es wird „globalisiert". Die ETH ist schon bald wie eine „richtige" Firma, womit der letzte Schatten auf dem Selbstbewusstsein schwindet. Man mag davon halten, was man will, es bleibt nur ein kleines Problem: In ihrer Selbstbeweihräucherung hat die ETH übersehen, dass sie nicht im luftleeren Raum steht. Der unwissende „Mann auf der Strasse" mag belächelt werden, als eigenössische Schule ist die ETH aber trotz Globalisierung national gebunden und hier, nicht in den USA, Rechenschaft schuldig. Der „Konzern" ETH muss seine Ansichten seinen Besitzern gegenüber glaubhaft vertreten. Mit Aufrufen und Drohungen (Stichwort: Arbeitsplatzabbau) ist es nicht getan. Die ETH kassiert hier als erste die Rechnung dafür, dass in heutigen Strategieberichten der Mensch nur noch als kostenverursachendes „Übel, welches sowieso keine Ahnung hat, angesehen wird.

Die ETH wäre gut beraten, diese Haltung zu überdenken, für den 7. Juni freilich ist es schon lang zu spät.

Michael Baumer,
Präsident VIS

Ressortverteilung SS 98

Präsident:
Michael Baumer mgb
e-mail: baumi@vis.inf.ethz.ch

Quästor &Exkursionen:
Kai Jauslin kj
e-mail: kai@vis.inf.ethz.ch

Aktuar & Vizepräsident:
Nicky Kern nk
e-mail: nicky@vis.inf.ethz.ch

Redaktion:
Stephan Würmlin sw
e-mail: stephi@vis.inf.ethz.ch

Feste & WWW:
Martin Näf mn
e-mail: martin@vis.inf.ethz.ch

VD / SD & Infrastruktur:
Eric Dondelinger ed
e-mail: aim@vis.inf.ethz.ch

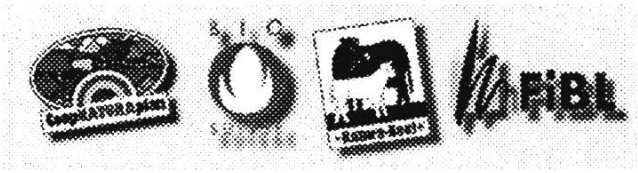
Verlag & Feste:
Pascal Kurtansky pk
e-mail: pascal@vis.inf.ethz.ch

Rechneradministration:
Thomas Andres ta
e-mail: tandres@vis.inf.ethz.ch

Unterricht & Information:
Michael Psarros mp
e-mail: mpsarros@vis.inf.ethz.ch

WWW:
Roland Brand rb
e-mail: roland@vis.inf.ethz.ch

Labelinth in den Läden



Da auch Informatiker zwischen Sourcecodes und Debugging essen müssen, habe ich mich mal aufgemacht, nachzusehen, was es ausser Bier und Chips sonst noch Nahrhaftes gibt. Und siehe da, ausser den erwarteten Einheitsbüchsen und Familien-Sparpackungen sprangen mir viele kleine bunte Bildchen von Kühen, Bauernhöfen mit Regenbögen und Knospen ins Auge (aua). Da es sich dabei nicht um Briefmarken handeln konnte, und kaum ein bekanntes Motiv (z. B. Intel™ inside) dabei war, mussten halt Informationen darüber her.

Eingeführt wurden diese Bildchen (Labels) weil für viele Konsumenten heute wichtig ist, dass sie über die Herkunft der Nahrungsmittel informiert werden. Über 90 % der Kunden betrachten die Labels als Orientierungshilfe beim Einkaufen, die ihnen eine spezielle Qualität oder eine spezielle Leistung (z. B. ökologische Produktion) garantiert, welche andere Produkte nicht vorweisen können. Aber viele Leute geben auch an durch die Vielfalt der Labels verwirrt zu sein, so wie auch viele Begriffe aus dem Informatikbereich für die meisten Leute Chinesisch sind.

Gründe wie lange Transportwege, undurchsichtige Produktionsweisen, Hormonskandale, BSE und andere sind für viele Konsumenten ausschlaggebend, schweizerische Bioprodukte zu kaufen und nach Qualitätsgarantien zu fragen. So sind die meisten einkaufenden Leute in der Lage, die wichtigsten Label zu erkennen, wissen aber oft nicht genau, was diese eigentlich garantieren. Beim Signet Coop-Naturaplan beispielsweise weiss eine große Mehrheit, dass damit Bio-Produkte ausgezeichnet werden, ist aber zum Grossteil gleichzeitig nicht im Bild, dass es sich dabei nicht unbedingt um Schweizer Produkte handeln muss. Beim Migros-Sano Label ist der Irrtum meist noch gravierender, viele Konsumenten sind der Ansicht, dass es sich dabei um Bio-Produkte handelt, obwohl damit lediglich IP-Standard garantiert wird. Die wichtigsten dieser Label will ich deshalb hier vorstellen, in der Hoffnung, dass Ihr alle auch das kaufen könnt, was Ihr wollt:

Die BioSuisse-Knospe ist das eigentliche Biolabel der Schweiz. Alle Produkte, welche dieses Label tragen, stehen unter der Kontrolle der VSBLO (Vereinigung der schweizerischen Bio-Landbau Organisationen). Trägt diese Knospe den Zusatz "SUISSE", so handelt es sich um Produkte schweizerischer Herkunft. Fehlt die Bezeichnung "Suisse", so kann es sich auch um ausländische Waren handeln, die jedoch dem schweizer Biostandard entsprechen müssen.

Stellt ein Landwirtschaftlicher Betrieb auf biologische Produktion um, so erhält er frühestens nach 2 Jahren, nachdem sein Betrieb voll und ganz biologisch produziert das Knospenlabel. Während den ersten zwei Jahren ist er aber berechtigt, die Umstellungsknospe zur Warenkennzeichnung zu gebrauchen.

Die Bezeichnung +Natura-Beef+ ist für eine besondere Art von tiergerechter Rindermast reserviert. Dabei wächst das Kalb zusammen mit seiner Mutter oder einer Amme auf. Daher spricht man auch von Mutter- oder Ammenkuhhaltung. Solange das Gras wächst, sind die Kühe und ihre Kälber täglich mehrere Stunden auf der Weide, oft rund um die Uhr. Im Stall dürfen die Kälber nicht angebunden werden und sind in Gruppen zu halten. Die Nahrung der Kälber ist weitgehend naturbelassen, also Milch und Grünfutter. Wachstumsfördernde Zusatzstoffe (z.B. Hormonkuren) sind verboten. Die Betriebe unterstehen der Kontrolle durch die SVAMH (Schw. Vereinigung der Ammen- und Mutterkuhhalter).

Beratung und Forschung des Biolandbaus geschieht zu einem grossen Teil durch das FiBL (Forschungsinstitut für biologischen Landbau) in Frick. Das FiBL führt im Auftrag der VSBLO auch die Kontrollen der Biobetriebe durch. Dieses Logo findet man allerdings kaum je auf Nahrungsmitteln.

Die Grossverteiler haben eigene Label, Migros zum Beispiel ist mit ihrem Migros-Sano Label recht bekannt geworden. Dieses Label erstreckt sich über die ganze Palette der Nahrungsmittel. Es begrenzt den Einsatz von Spritzmitteln und Dünger auf ein vertretbares Minimum, regelt die Tierhaltung und Bodenbewirtschaftung. Migros bietet den Migros-Sano-Produzenten auch Beratung an.

Da immer mehr Konsumenten (darunter auch Informatikstudenten ...) vollbiologische Produkte mit vollständigem Verzicht auf chemische Produkte wünschen, bietet Migros solche Nahrungsmittel unter dem MigrosBio-Label an. Diese Waren entsprechen zwar den Anforderungen des Knospenlabels, sind aber nicht garantiert aus einheimischer Produktion. Dieses Label wird wie die Knospe durch das FiBL kontrolliert.

Etwas später als Migros, dafür mit grösserem ökonomischen Erfolg führte Coop sein Label Coop

SPRACH DESIGN

SPRACHDESIGN

SPRACHDESIGNSPRACHDESIGN

SPRACH DESIGN

Wir suchen eine(n) herausragende(n) Informatiker(in) für Design und Prototyping anwendungsspezifischer Sprachen. Detaillierte Information zum Projekt unter www.tik.ee.ethz.ch/~montages

KONTAKTADRESSE: PROF. DR. THIELE, INSTITUT TIK, ETH ZÜRICH

NATURAplan ein. Dieses Label beinhaltet Knospenprodukte und +Naturabeef+ sowie Schweine aus kontrollierter Freilandhaltung. (z.B. Porco Fidelio oder Natura-Porc um noch ein paar andere Label zu nennen).

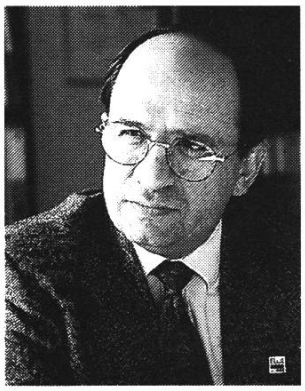
Ein eher exotisches Label ist das Demeter, dass für biologisch-dynamische Produkte steht, welche nach den Weisungen Rudolf Steiners angebaut werden. Solcherart erzeugte Nahrungsmittel sind jedoch recht teuer, und eher in Reformhäusern und auf dem Markt als in normalen Läden zu finden.

Neben den Biolabeln gibt es noch die IP-SUISSE Label, ein Mittelding zwischen Bio und konventioneller Produktion. In der Schweiz sind die meisten landwirtschaftlichen Betriebe IP-Betriebe. Wie der Name IP (integrierte Produktion) schon sagt, wird hierbei grosser Wert auf ökologische Kreisläufe gelegt, und nur die Nährstoffe, welche den Betrieb in Form von Milch, Fleisch oder pflanzlichen Produkten verlassen, dürfen in Form von Mineraldünger wieder zugekauft werden. Damit soll ein Gleichgewicht geschaffen werden, das ein stabiles Ökosystem ermöglicht. Spritzmittel werden so wenig wie möglich eingesetzt. Ansonsten ist IP der Bioproduktion sehr ähnlich.

In IP-Suisse sind auch, wie in NATURAplan (allerdings ist IP nicht von einem Grossverteiler), mehrere Label zusammengefasst, z. B. sind IP-Obst, IP-Kartoffeln, IP-Gemüse, Agrinatura (Schweinefleisch), VINATURA (Wein), oder eben Migros-Sano IP-Label.
Was genau BIO eigentlich ausmacht, machen die Grundideen des Biolandbaus ersichtlich:

+ schonungsvoller Umgang mit Natur und Umwelt
+ richtet sich nach den natürlichen Kreisläufen
+ selbstregulierende Ökosysteme fördern (Artenvielfalt)
+ natürliche Bodenfruchtbarkeit durch geeignete Kulturmassnahmen steigern
+ keine chemisch-synthetischen Dünger einsetzen
+ vorbeugender Pflanzenschutz durch Wahl klimatisch geeigneter, widerstandsfähiger Sorten und Arten
+ keine chemisch-synthetischen Pestizide
+ artgerechte Tierhaltung und -fütterung
+ Quantität darf nicht auf Kosten der inneren Qualität erzielt werden
+ vollwertige Lebensmittel für eine gesunde Ernährung erzeugen und diese erkennbar machen (Label)
+ Bei der Herstellung von Bio-Produkten wird nicht nur auf den Einsatz gentechnisch veränderter Organismen (GVO), sondern auch deren Folgeprodukte (v. a. Enzyme) verzichtet.
(NB: gentechnisch verändert ist, wessen Erbsubstanz verändert wurde, also z. B. Antimatschtomaten, bei denen ein Gen stillgelegt wurde, welches für die Ethylenproduktion zuständig ist, also für die Reife. Klone hingegen sind nicht

Daniel Gorostidi
Generaldirektor
ELCA Informatik AG



Um uns immer wieder neu für die Betreuung anspruchsvollster Informatik-Projekte zu qualifizieren, brauchen wir die besten Spezialisten*. Gute Beziehungen zu ETH, welche kompetente Absolventinnen und Absolventen hervorbringen, gehören zu unserer Erfolgsstrategie.

* unter anderem Informatiker, Mathematiker, Elektroingenieure und Physiker

Die ELCA Informatik AG ist einer der bedeutendsten Anbieter von Informatik Dienstleistungen in der Schweiz. Dank den Leistungen seiner 135 Ingenieure konnte das Unternehmen bisher einen grossen nationalen und internationalen Kundenkreis aufbauen und erfolgreich betreuen. Um ihre zukünftige Entwicklung zu sichern und neue Herausforderungen von seiten der Auftraggeber annehmen zu können, sucht die ELCA junge Menschen, die sich für faszinierende Aufgaben engagieren wollen.

ELCA
INFORMATIK

zertifiziertes Qualitätssystem
SQS
ISO 9001/ EN 29001
Reg.No 10730

ELCA Informatik AG ▪ Hofwiesenstrasse 26 ▪ CH-8057 Zürich
Tel. 01/ 363 46 00 ▪ Fax 01/ 363 49 46
e-mail : info@elca.ch ▪ www.elca.ch

Lausanne ▪ Genf ▪ Bern

gentechnisch verändert, weil kein Eingriff in die Erbsubstanz vorgenommen wurde. Ebenso ist künstliche Besamung keine Gentechnik. Beim Klonen und der künstlichen Besamung handelt es sich um Biotechnologien.)

Biologisch produziert ist gut....

Wieso wird eigentlich noch etwas anderes als Bio angebaut, wenn doch alles so gut tönt?
Die Tiere werden artgerecht gehalten, keine Chemie....

.....aber nicht alles was glänzt ist Gold

Auch Biobetriebe haben Probleme mit Krankheiten von Tieren und Pflanzen. So werden Pilzkrankheiten von Pflanzen häufig mit Kupferpräparaten behandelt, welche giftig sind und im Boden nicht abgebaut werden können. In Weinbergen hat man mancherorts schon eine Kupferkonzentration im Boden, welche höher ist, als die in den Kupferminen.
Auch kann man mit den Hofdüngern (Gülle und Mist), welche ja auch im Biolandbau anfallen, eine Menge falsch machen. Falsch ausgebrachte Gülle kann schnell den Nitratgehalt des Grundwassers in die Höhe schnellen lassen, allerdings unabhängig vom Label...

Falls Ihr das Gefühl habt, jetzt den Durchblick zu haben, muss ich Euch darauf hinweisen, dass Ihr aufpassen müsst, dass Ihr nicht irrtümlicherweise das Logo von EUROFARM, vom SGU/USL (Schw. Gemüseunion) vom SKK/CSP (Schw. Kartoffelkommission) vom SOV (Schw. Obstverband) oder von sonst irgendwem für ein Label haltet.
Ausserdem wird es demnächst ein neues Label geben, das SwissQuality-Label.
Dieses Label garantiert dann u. a., dass höchstens 15 % ausländische Rohstoffe verwendet wurden, und gewisse Qualitätsansprüche bezüglich Aussehen, Inhaltsstoffen, Hygiene und Produktionsweise befriedigt werden.

Pascal „Moses“ Wittmann
pwittman@agrl.ethz.ch

Weiterführende Informationen über Bio, IP und Labels:

http://www.fibl.ch

http://www.ipgesellschaft.ch

http://www.coop.ch

http://www.migros.ch

ausserdem:
Landwirtschaftliches Handbuch 1997 oder 1998, Verlag Wirz, Basel

aber das interessiert ja eh niemanden...

ZIP Drives im Rif/Raf-Raum

Dass die ZIP-Drives (ZD) nun verfügbar sind, dürfte sich bei Euch und allen RifRaf-Usern inzwischen herumgesprochen haben. Doch wie werden sie bedient?

Erste Hilfe leistet der Zettel, beim Eingang zum Raf-Raum: „zipmnt -h" eintippen und vielleicht schon das erste Aha-Erlebnis feiern. Doch vielleicht auch nicht. Der zipmnt-Befehl funktioniert nur auf Rechnern mit eingebautem ZD richtig. Natürlich ist aber auch per remote login (den man heutzutage besser mit ssh [secure shell] durchführt) der Zugriff möglich.

Anders als bei den Floppies wird eine Zipdisk NICHT in ein Unix-Verzeichnis gemountet, wo über Standardbefehle auf die Daten zugegriffen werden kann.
Immer ist der kleine Umweg über zipmnt noetig. Zipmnt unterstuetzt viele der sogenannten mtools, also den MS-DOS-Tools, die schon für Floppies verwendet werden konnten. Beispielsweise zeigt „zipmnt mdir z:" den Inhalt des aktuellen Verzeichnisses an. Leider geht mcd nicht, d.h. der Verzeichniswechsel ist nicht möglich. Demnach kann man Dateien in Unterverzeichnissen nur mit dem vollen Pfadnamen ansprechen oder per Schalter /s, was bei einigen m-Befehlen bedeutet „beziehe Unterverz. mit ein".
Wer die Zips einfach für den Datentransfer Home - ETH - Home braucht, den stört dieser Umstand gar nicht. Er/sie packt mit alle Dateien und Unterverzeichnisse per „tar" und „gzip" in ein grosses File und speichert es im Wurzelverzeichnis der Zipdisk. Zuhause wird mit Winzip (auf Mac wohl StuffIt) alles wieder entpackt. Auch für den Retourweg gibt es kleine Helfer.

Ein weiterer Unterschied zu den Floppies ist die Ownership einer eingelegten Disk: der erstmalige zipmnt-Aufruf nach Einlegen der Zipdisk dauert ca. 10s und setzt einen Lock auf die Disk. Wenn die Benutzerin das ZD nun ständig braucht (d.h. maximale Zeitspanne, wo nichts läuft, höchstens 300s) und sich nicht von der Maschine ausloggt, kann niemand anders als sie mit ihren Daten arbeiten. Bei max. 100 MByte persönlichen Daten ein wichtiger Umstand, da sonst, sei es versehentlich oder absichtlich, fremde Daten gelöscht werden könnten. Achtung: nach 300s Idle-Time erlischt dieser Lock. Und falls kurz nach dem Einschieben der Zipdisk jemand dem rechtmässigen Besitzer zuvor kommt mit zipmnt, hat die falsche Person alle Rechte. Hier kann einzig an die Vernunft der Informatikstudierenden appelliert werden, solche Spässe zu unterlassen.

Was noch kommen wird, allerdings ohne Angabe eines Termins, ist einerseits ein grafisches Tool (xzip) zur Verwaltung des ZD und andererseits JAZ-Drives. Hardwareanschaffungen sind eine Frage des Budgets - ZIP und JAZ waren 1998 nicht darin vorgesehen. Wir können uns glücklich schätzen, dass es trotzdem zu vier ZD gereicht hat. Danke an dieser Stelle dem Departementsvorsteher Prof. Gander für sein rasches und unbürokratisches Handeln.

cf

Testat- und Zulassungskontrolle

zu den Prüfungen im Herbst 1998

Montag 22. Juni 1998
bis und mit
Freitag, 26. Juni 1998

jeweils ganztags

Wichtig:

auch Kandidaten des Fachstudiums müssen sich zur Zulassung melden, obschon dort keine Testate mehr erforderlich sind.
Das gleiche gilt für Repetenten jeder Prüfungsstufe.

Das Studiensekretariat

Präsenzen im SS 98

VIS Büro
IFW B29
Tel. 01 632 72 12
Fax: 01 632 11 72

Präsenzzeiten:
Montag bis Freitag, 12.15-13.00

Mo: **Kai Jauslin**

Di: **Martin Näf**

Mi: **Thomas Andres**

Do: **Eric Dondelinger**

Fr: **Pascal Kurtansky**

FrameMaker im Rif/Raf - Raum

Neu gibt es auch den FrameMaker 5.5 von Adobe zu benutzen im RifRaf.

Interessieren dürften sich vor allem Diplomanden oder andere Studierende, die auch ohne TeX-Kenntnisse eine saubere Arbeit abliefern möchten. Vorgehen zur Benutzung: „module add framemaker" eingeben.
(Übrigens zeigt „module avail" alle verfügbaren Modules an!)
Mit „maker" das Programm starten. Nach einer Weile kommt das GUI (graphical user interface) und ihr fühlt Euch wie zu Hause...

Christian Fritz

Be OS Release 3

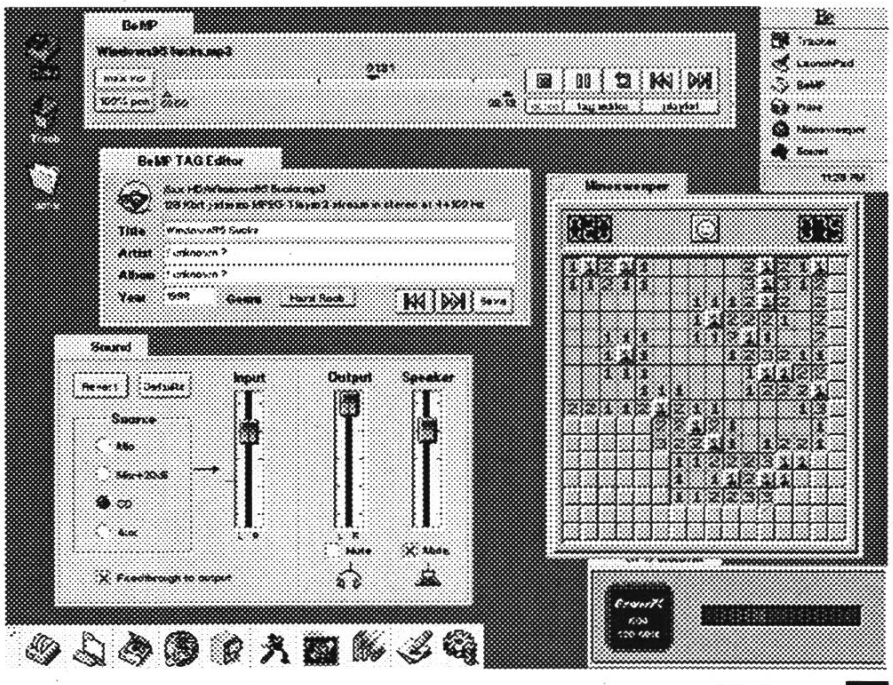
Einen Bericht über Oberon System 3 kann ich Euch diese Ausgabe nicht bringen, dafür hab ich mir kurz mal was über das neueste Release des BeOS aus den Fingern gesogen. Mittlerweile liegt dieses Betriebssystem in der Version 3 vor, erstmals auch für Intel-Maschinen. Ich selbst teste die PPC-Version - mangels PC. (Nicht, dass ich einen wollte.) Mal schauen, was sich getan hat.

Die Installation

Genug gelabert. Als die CD mit der neuesten Version endlich in meinem Briefkasten lag, wurde sofort eine Partition auf der Festplatte geräumt (Dann werden jetzt halt mal keine CDs mehr gebrannt!) und mit der Installation begonnen. Und da traf ich auch schon auf den ersten Bug. Den gab's schon in der Release 2. (Eigentlich schon seit der Developer Release 8, meinen ersten Kontakt mit der damals neben MkLinux der einzigen Alternative zum MacOS auf dem Mac, nur unterstützte DR8 eh nur 640*480, so fiel das gar nicht auf.) Wenn man beim Start des Installers die Bildschirm-Auflösung nicht auf 640*480 eingestellt ist, bleibt dieser schwarz. So ist es natürlich nur unwesentlich schwierig, die richtige Platte und die richtigen Buttons zu erwischen. Aber wenn man dies mal korrigiert hat, gestaltet sich der Installationsprozess denkbar einfach: Man gibt an, wohin die Daten geschmissen werden sollen und ob man die optionalen Items mitinstallieren möchte. Dann noch ein Klick auf den Start-Button (Nein, nicht DEN!) und los geht's.

Das System

Mit dem BeOS steht ein weiteres „modernes" Betriebssystem zur Verfügung. „Modern" heisst hier konkret Multitasking (Mehrere Prozesse gleichzeitig), Multithreading (Mehrere Teilprozesse innerhalb einer Applikation) und Speicherschutz (ein abstürzendes Programm reisst nicht noch andere mit ins Datennirvana. Tausend Zeilen C++ sind Strafe genug!). Ausserdem ist es komplett Objekt-orientiert, was man besonders deutlich sieht, wenn man sich die Mühe macht, durch die APIs (Application Programming Interface) zu stöbern. Dass man Druckertreiber und ähnliches zur



Laufzeit installieren und ändern kann, ist ein weiterer Vorteil.

Das BeOS schimpft sich „MediaOS". Damit wollen die Entwickler von Be ausdrücken, an welche Zielgruppe sie sich in erster Linie richtet. Tatsächlich jongliert das BeOS mit Video- und Audiodaten so leicht wie andere mit Text. Im wesentlichen ist auch dieses System ein Unix-Derivat wie beispielsweise Linux, AmigaOS oder Rhapsody. Die Optik ist zwar anders als gewohnt, ausser dem Gag, den Windowtitel nicht über die gesamte Breite des Fenster zu ziehen, sondern als Tab obenlinks. √Ñhnlich wie bei Win95 oder NT gibt es auch hier eine Art Taskbar, den „Deskbar", der die laufenden Prozesse auflistet. Das Filehandling ist ähnlich wie beim Mac. Allgemein sieht man überall, dass das MacOS für das GUI Pate stand. Allerdings hat Be einige Kleinigkeiten verbessert und macht einen soliden Eindruck.
Womit ich auch schon bei der Stabilität wäre. Abstürze, die das System mit in den (virtuellen) Tod reissen, sind mir noch nicht untergekommen. Die eine oder andere Applikation beendet sich manchmal mit sehr informativen Fehlermeldungen (MacOS lässt grüssen!), doch sonst muss ich sagen, dass seit der Entwicklerversion 8 (Vorher kannte ich es noch nicht.) das System unglaublich stabil ist.
Was mir als Mac-Veteran zunächst zuwider war, will ich heute nicht mehr missen: die Shell. (Hätte nie gedacht, dass ich so was mal sagen würde.) Be hat dem Anwender eine hübsche Bash spendiert, und wem der graphische Overkill zu viel ist, kann auch (fast) nur Kommandozeilen-orientiert arbeiten.

Software

Dem Betriebssystem liegt schon eine beachtliche Menge an Software bei; von einfachen Demos bis zu ausgewachsenen Applikation ist alles dabei. Für Entwickler besonders interessant ist die IDE (Integrated Development Environment) von Metrowerks. Obwohl der CodeWarrior in dieser Lite-Version nur Projekte bis 64 KB erlaubt, lässt sich somit ein guter Einblick in die Programmierung des BeOS gewinnen. Die APIs sind gut gegliedert und mindestens genauso gut dokumentiert, aber dazu später.
Es existiert sogar schon ein MP3-Player, der das Abspielen von Files, die man noch aus dem Netz runterlädt, zulässt. (Seit ich das Teil gefunden habe, plärrt mir immer die Verarschung der Win95-Werbung um die Ohren ;-)
Auch der Spass soll nicht zu kurz kommen und deshalp sind auf der CD ein paar Games mit drauf. Dabei reicht die Palette vom allgegenwärtigen Minesweeper über Asteroids bis zu Jump'n'Runs. Auch Doom ist schon für BeOS portiert worden.
Auch die Netzwerk-Tools können sich sehen lassen. Ein Mini-Webserver wird beispielsweise gleich mitgeliefert. Der Be-eigene Browser versteht zwar nur reines HTML (also kein Javascript oder Java und schon gar kein (Radio-)Active-X), ist dafür aber relativ schnell und übersichtlich zu bedienen. Auch der mitgelieferte eMail-client reicht für den einfache Zwecke. Und wem der nicht reicht, kann sich einen komplexeren aus dem Internet runterladen.
Damit bin ich beim Thema Softwarebeschaffung. Allgemein findet man alle verfügbare Software für BeOS auf der Webpage von Be ausfindig machen.

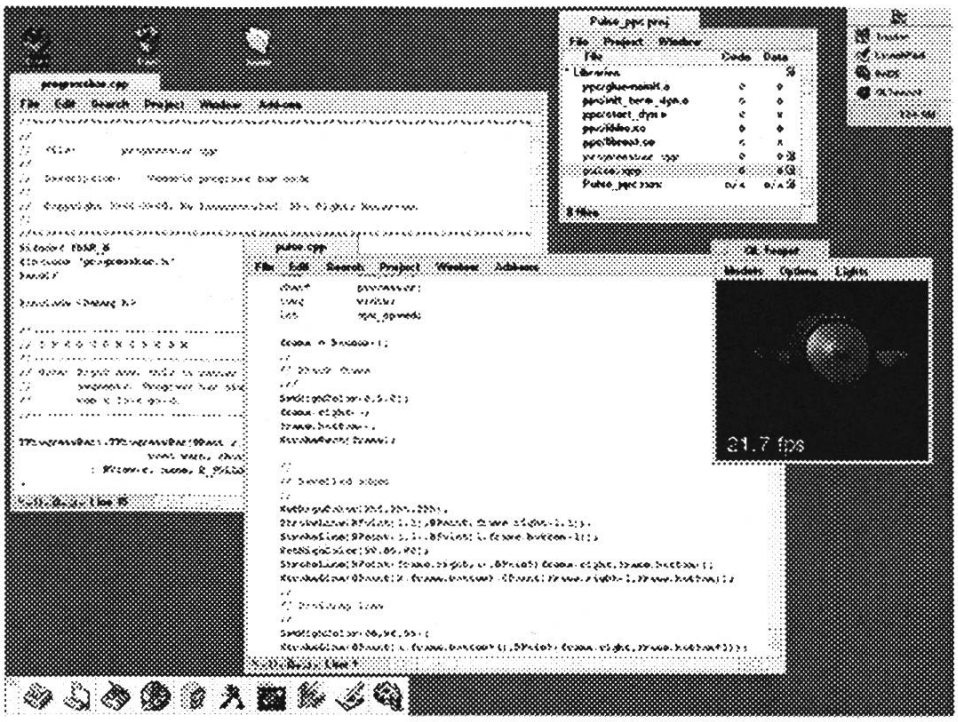
(http://www.be.com/beware) Das ist auch gleich die beste Adresse, wenn man irgendwas wissen muss. Die Mitarbeiter von Be sind sehr hilfsbereit und versuchen alles, um Schwierigkeiten aus dem Weg zu räumen. Schon jetzt stehen sehr viele Programme zum Download zur Verfügung. (Und über allen Downloads wacht die Quota. Amen!) Noch viele mehr sind in Entwicklung. Man kann also gespannt sein.

Im Westen was neues

Der „Tracker" - so heisst der Finder (für Mac-User), resp. Datei-Explorer (für diejenigen, die sich von MS knechten lassen) - lässt sich nun scripten. (Wenigstens ein Stück weit.) Ausserdem der Support für die „Add-Ons" vereinfacht worden. Ein „Add-On" ist im Prinzip nichts anderes als eine Erweiterung des Systems. (Aber das legt der Name ja schon nahe.) Ich hab bis jetzt noch kein Betriebssystem gesehen, dass es so einfach ermöglicht, einem Menübefehl einen Keyboard-Shortcut zuzuweisen. Das Ergänzen des AddOn-Names mit einem Strich (-) gefolgt vom Shortcut ergibt den gewünschten Effekt.

Die 3D-Engine scheint auch neu geschrieben worden.(Das hätten sie besser bleiben lassen.) Jedenfalls ist die neue Engine ziemlich langsam. (Wenn ich langsam sage, meine ich auch laangsaaam!) So konnte ich in der Release 2 locker einen Würfel mit vier 10 MB Animationen

ruckelfrei in Echtzeit drehen, während er jetzt schon ohne Animation nicht mehr so flüssig dreht. Naja, wenn das der einzige Drawback bleibt, den ich entdecke, bin ich doch noch ganz zufrieden. Immerhin kann ich eine 10 MB Animation in Echtzeit beinahe ruckelfrei skalieren. Das muss ich mit dem MacOS und QuickTime 2.5 auf dem gleichen Mac nicht probieren.



Dokumentation

Sehr schnell entdeckt man auch tausende HTML-Files. So viele, dass man meinen koennte, HTML sei ein Ersatz für die auf anderen Systemen einer Firma, die nicht weiter genannt zu werden braucht, allseits beliebten DLLs. Doch weit gefehlt. Be gibt sich alle nur denkbare Mühe, das System zu dokumentieren. Andere Firmen könnten sich von dieser Dokumentationswut eine Scheibe abschneiden. Vor allem als Programmierer profitiert man davon, dass die APIs so ausführlich erklärt werden.

Fazit

Ich glaube nicht, dass Be mit ihrem System auf dem Weltmarkt Fuss fassen können, aber für Programmierer und andere Freaks stellt es auf jeden Fall eine interessante und erfrischende Alternative zu dem Einerlei, der einem sonst so geboten wird, dar. Es sind einige Konzepte sehr konsequent durchgesetzt worden, die viel Raum zum experimentieren lassen. So sind die Mechnanismen, wie zwei Applikationen miteinander kommunizieren können, einfach und genial, ebenso wie die Replikator-Technologie, die dem leider eingestellten OpenDoc von Apple sehr ähnlich sieht. Damit ist es prinzipiell möglich, Threads einer andern Applikation in der eigenen ablaufen zu lassen. Das BeOS wird nie mein Primärsystem ersetzen, aber trotzdem werde ich es behalten, nur schon um eben diese Dinge noch zu erforschen. Vielleicht wird ja mal was grösseres draus...

Kai Jauslin

Python - die High-Level-Skriptsprache

Als Ingenieur kann es mal vorkommen, dass man von seinem Arbeitgeber Nägel bekommt und sie mit Motorsägen einschlagen muss. Jeder war mal in der Situation, dass er/sie einen billigen Hack abliefern musste, weil keine anderen Tools vorhanden waren, oder noch schlimmer: weil sie vom Management aus irgendwelchen Gründen verboten wurden (‚Dafür finden wir nie einen neuen Entwickler', ‚Produkt M ist ein strategisches Produkt, und wird deshalb verwendet'). Die ‚lustigste' Idee die ich mal persönlich vorgeschlagen bekam, war folgende: Ein Relais (eine Art Bit für Starkstrom) musste telefonisch ein-/ausgeschaltet werden können. Als Programm sollte Access Basic verwendet werden, weil das die einzige Sprache war, welche diese ‚Informatiker' kannten. Ein Overkill, der a) das Problem nicht gut löst und b) eher belustigt, als schockiert. Leider passiert es immer wieder, dass man nicht das beste Werkzeug kennt. Ich zeige Euch darum hier eine Programmiersprache, welche folgende Eigenschaften besitzt:

- *einfach zu lernen.*
- *extrem mächtig.*
- *existiert auf allen bekannten Plattformen: windows, unices, mac, vms, amiga, atari und noch ein paar mehr.*
- *Wie bei Java gilt in Python: ‚Compile once, use everywhere'.*

So verteilt z.B. unser WG-Linux-server via NFS ein Skript zum Starten der Internetverbindung für alle Dosen und Linuxen im Netz, welche vom OS-spezifischen Interpreter ausgeführt werden. Der Client ist mit Benutzeranmeldung, Netz ein-/ausschalten, Server anhalten und einem GUI gerade mal 55 Zeilen lang, der Server (welcher die Anfragen auf einem Socket entgegen nimmt), hat 155 Zeilen...

Was ist nun dieses Python?

Python ist eine Skriptsprache, die sehr viele tägliche (einfache) Probleme lösen kann, aber auch bei schwierigen nicht gleich ins Gras beisst. Man programmiert mit den gleichen Compound Statements wie in Systemsprachen (for, while, if etc.), jedoch sind sie viel mächtiger: for kann zum Beispiel auf Listen operieren, und auf allen Schlüsseln oder Werten einer gehashten Liste. Diese beiden neuen Datentypen machen wohl schon einen grossen Teil der neuen Bequemlichkeit aus. Intern werden alle Objekte, die nicht einem Basistyp entsprechen (int, float, string) in einer gehashten Liste gespeichert: z.B. Module und Klasseninstanzen. Dies funktioniert, weil man nur mit Referenzen arbeitet, nie mit dem Objekt selber.
Python ist aber auch eine vollständige OO-Sprache: Mehrfachvererbung, Polymorphie, Operator-Overloading sind dabei.
Für die meisten Aufgabengebiete gibt es Module in der Library, so dass man kaum das Rad neu erfinden muss. Die Effizienz von Python kommt daher, dass die wichtigen Library-Module in C geschrieben sind. Teile, welche der Benutzer effizient haben möchte, würde er somit ebenfalls in C programmieren. Dadurch ist ein neues Paradigma der

Programmierung geschaffen worden: ‚Event-driven C-Programming'. Wer die letzten 20 Jahre nicht in einer Höhle verbracht hat, wird wohl auch gleich verstehen um was es geht: Wieso soll man ein effizientes C-Programm, mit doppelt so vielen Zeilen Code um ein grafisches Interface erweitern, damit es nachher nur auf einer Plattform funktioniert?
‚Event-driven C-Programming' bietet Callback-Funktionen in einer Library an, welche von einer Hochsprache benutzt werden. Dies funktioniert auch für Libraries, von denen man nur die Headerfiles hat. Mit dem ‚Simple Wrapper and Interface Generator' (SWIG) kann man mit einem Headerfile als Interfacebeschreibung für alle momentan wichtigen Skriptsprachen Interface-Module generieren. Die Autoren von SWIG hatten 20 Minuten, bis sie OpenGL als Modul in der Skriptsprache integriert hatten. Nebenbei dokumentiert es noch schnell alle Funktionen, wenn man sie richtig kommentiert hat.
Ich habe SWIG kürzlich getestet, und die 20 Minuten sind wirklich kein Witz: Das Handbuch ist vom Feinsten und erklärt für alle Kombinationen von Skript- und Systemsprachen die Bedienung, abgerundet von einem Referenzmanual. Ich kenne kein kommerzielles Produkt, dass da rankommt...
Wer nicht glaubt, dass Freeware brauchbar sein kann, der soll sich mal dieses Paket anschauen!!!

History

Python wurde von Guido van Rossum iniziert, einem holländischen Informatiker. Das ist wichtig, weil ja heute schon Linguisten Programmiersprachen schreiben dürfen (Larry Wall mit Perl).
Als Vorfahren von Python sind sicher die Shelldialekte (csh, ksh, bash etc.) zu nennen, aber auch Perl und Tcl in ihren Anfangsstadien haben die Sprache inspiriert! Der Name hat übrigens nichts mit den gleichnamigen Schlangen zu tun, sondern ist als Ehrung der englischen Comedytruppe ‚Monty Python' gedacht.
Python tauchte 1991 das erste Mal in der einschlägigen Szene auf, und seit 1994 gibt es auch eine Newsgroup (comp.lang.python) dafür.
Die Benutzerzahlen sind seither exponentiell am steigen. Die bekanntesten Anwendungen von Python sind wohl die Systemadministrationstools von Red Hat Linux. Verbreitung findet es vor allem in der Industrie und an Universitaeten.

Sprachvergleich

Die Auswahl an High-Level-Sprachen ist sehr gross: wieso sollte man nun ausgerechnet Python nehmen?
Wie Tcl, eignet sich Python gut für das Einbetten in Systemprogrammiersprachen. Jedoch verfügt Tcl nur über String Datentypen: Zahlen werden in Strings umgewandelt und umgekehrt. Python hingegen ist eine vollwertige, abgeschlossene Programmiersprache, welche man in kurzer Zeit erlernen kann: Aus einigen Vorlesungen wissen wir ja, dass eine Sprache auf einem A4 Blatt in EBNF Platz haben muss. Mit momentan knapp 30 reservierten Wörtern ist sie auch weit enfernt von Larry Wall's Perl, welches einen ganzen Namespace reserviert, damit auch ja nie die reservierten Wörter ausgehen. Variablen muessen in Perl also

mit einem Sonderzeichen beginnen, was die Sprache schon mal visuell hässlich macht.

Perl hat seine coolen Features, ist jedoch eine Write-Only-Sprache: Ist der Code einmal geschrieben, so ist er nicht mehr lesbar und 2 Monate später, wenn man ihn mühevoll wieder verstehen will, merkt man, dass es viel einfachere Wege gäbe, um dieses Problem zu lösen. Leider wiederholt sich dieser Zyklus ein ganzes Programmiererleben lang, und Mitarbeiter, welche noch nicht den selben Perl-Wizardry-Level erklommen haben sind sowieso chancenlos im Erkennen der Funktion von fremdem Perl-Code. Als Beweis hier ein Codefragment aus einem Perlbuch. Setzt jetzt bitte die offizielle Perl-Zauberer-Antigravitations-Kappe auf oder haltet Euch irgendwo fest.
Was macht dieser Einzeiler?

```
@t = map { chomp; [ split] ; } sort <>;
```

Wer es nicht rafft, hier der Code in Python:

```
import string
 #enthält string routinen
import sys
 # enthält systemspezifisches
t = []
 # wir instanzieren eine liste
for i in sys.stdin.readlines():
 # loop über alle zeilen von stdin
t.append(string.split(i))
 # zeilen nach spaces splitten und
 # in unsere liste einfügen.
t.sort()
 # die liste sortieren
```

Falls ihr den Perl-Code nicht verstanden habt, dann seid ihr nicht allein:
Ich habe mal einen gestandenen Perl-Guru damit konfrontiert, und er benötigte auch eine Minute bis er es mit einem ‚Cool!' klassifizieren konnte.
Mit der in letzter Zeit viel gehypten Sprache Java hat es folgende gemeinsame Eigenschaften:

- Unterstützung von OOP
- Kompilation nach Byte-Code
- Portabilität
- Modularisierung
- Applet Programmierung möglich in Netscape (plugin), IE (ActiveX) und Grail (Python Webbrowser).
- Garbage Collection

Vorteile gegenüber Java sind:

- Higher Level: Das ‚dynamic typing', eine orthogonale Sprache (mit möglichst wenig Keywords moeglichst viel erreichen), das First-Class-Objekt Modell (man arbeitet vor allem mit Klassen und Datenobjekten) und die eingebauten Datentypen machen Python zu einer einfacheren Sprache.
- Absolut gratis: Im Gegensatz zu Java probiert bei Python niemand sich zu profilieren oder bald Geld abzuschöpfen (Sun, Microsoft et al). Die Lizenz erlaubt jegliche kommerzielle Benutzung: Man dürfte sogar die Sourcen des Compilers nehmen und ihn verkaufen, wenn man denn einen Käufer fände!
- Generalität: Nichts an Python ist Web-spezifisch. Es eignet sich auch für das Einbetten und Erweitern von Systemsprachen (auch Java), generelle GUI Programmierung, Prototyping (‚Ach der Client/Server Prototyp soll bis heute abend fertig sein? Kein Problem!'),
- Reflection: Python Objekte können sich selbst beschreiben, weil die Sprache erst zur Laufzeit entscheidet, ob eine Methode/ein Attribut definiert ist. Bei

Java muessen sogenannte ‚Factories' Objekte erzeugen. Unter Python speichert man alle Daten (Klassen, Daten, Module) in einen Hash oder einer Liste und übergibt sie ‚pickle' (engl. einmachen), welches sie auf Disk oder in einer Datenbank abspeichert (Oracle, Informix, Sybase, m(y)SQL). Dabei ist es egal welche Datenbank man hat, da das Datenbank-Objekt gegen Python dasselbe Interface verwendet.

- Schutz vor Reengineering von Byte-code: Python Code lässt sich zu einem einzigen ausführbaren Programm hinunter'squeezen', welches man auch verschlüsseln kann. Im Gegensatz zu Perl wird dabei nicht ein Coredump des laufenden Interpreters gemacht (echt!), um ihn später an der gleichen Stelle fortzusetzen, sondern der Python-Interpreter wird mit dem Byte-Code des Skripts zu einem einzigen File compiliert (C-Code). Der Geschwindigkeitsgewinn ist minim (ca. 1-3%), jedoch ist man nicht mehr abhängig von Änderungen in den Modulen oder deren Platzierung im Filesystem. Der Stabilität einer Installation tut es sicher gut!

Bei Vergleichen von Java und Python ist es wie bei denen von Äpfeln und Birnen:
Es macht keinen Sinn (manche sagen sogar, es handle sich bei den beiden Sprachen um verschiedene Nahrungsmittelgruppen).

Python hat sicher auch Nachteile, welche ich hier nicht verschweigen will. Als interpretierte Sprache ist sie natürlich nicht so schnell, was jedoch bei den meisten Anwendungen heute keine Rolle mehr spielt. Rechenintensive Teile schreibt man sowieso am besten in C, welches mit SWIG einfach in Python integriert wird.
Momentan fehlt noch ein eigenes, plattformübergreifendes GUI Toolkit:
Das von Tcl ‚geklaute' Tk ist über einen Tcl-Interpreter in Python integriert. Jedoch gibt es viele OS-spezifische Toolkits, welche bei einem genau spezifizierten Plattformen-Zoo (z.B. nur Unix und Windows) auch Verwendung finden koennen: MFC, Java AWT, Tk, Qt, gtk, V und viele mehr.

Damit dieser Artikel nicht explodiert, stehen alle Programmbeispiele auf dem VIS-Server
http://www.vis.inf.ethz.ch/python/
zum Herunterladen bereit. Es hat dort auch Links zu den wichtigsten Sites und den coolsten aktuellen Projekten.

Ach ja, lasst Euch von der ‚visuellen' Programmierung (Zeilen mit der gleichen Einrückung gehören zum gleichen Statement Block) nicht abschrecken; sie hat auch positive Aspekte.

Beat Christen
IIIC / 8

Pretty Good Privacy

nk. *Anlässlich der nächsten VIS PGP Key Signing Party gibt es wieder einmal einen Artikel über das Thema Kryptographie im Allgemeinen und PGP im Speziellen.*

Der Artikel ist in mehrere Teile aufgeteilt, die separat gelesen werden können. Ich habe mich bemüht, sie so zusammenzusetzen, dass der Artikel als Ganzes noch lesbar bleibt. Auf diese Art, kann man sich aber die interessanten Teile aus dem Artikel herauspicken (z.B. den Teil über PGP 5 oder über die nächste Signing-Session), oder, falls man noch nichts oder nur wenig mit PGP zu tun hatte, den Artikel als Ganzes lesen.

Zum warmwerden gibt es im Teil „Was ist PGP und warum soll ich es benutzen?" eine allgemeine Einführung zum Thema Public-Key-Kryptographie und Internet-Sicherheit.

Für das genaue Verständnis der Funktionsweise von PGP ist das „Web of Trust" unerlässlich. Es ist so wichtig, dass ihm ein eigener Teil gewidmet wird.

Letztes Jahr kam die „neue" Version von PGP heraus. Sie verwendet andere Schlüssel, einen anderen Verschlüsselungsalgorithmus, und überhaupt ist alles neu (und mehr oder weniger inkompatibel zu dem, was es bisher gab). Diesem Thema widmet sich der Teil „Versionenärger".

Zum Abschluss kommt die genaue Beschreibung der nächsten Signing-Session. (Hoffentlich) alle interessanten Fragen werden kurz angesprochen: wer darf teilnehmen, wann/wo findet sie statt, wie kann/muss ich mich anmelden, ...

Ich habe mir Mühe gegeben, PGP 5.x und 2.6.x überall gleichermassen zu beschreiben.

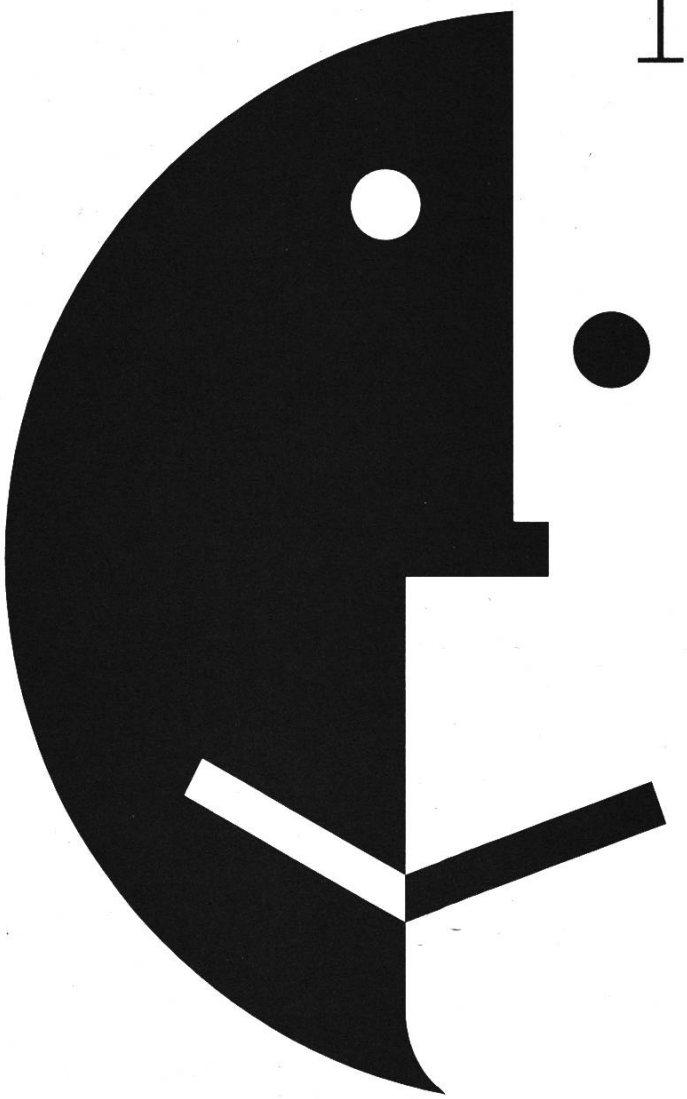
Ich möchte an dieser Stelle Patrick Feisthammel herzlich danken. Ich konnte seinen letzten Artikel über PGP als Basis für diesen verwenden (der Teil über das Web of Trust ist annährend wörtlich von ihm übernommen). Ausserdem hat er die (ausgezeichnete) PGP-Seite des VIS erstellt.

Was ist PGP und warum soll ich es benutzen?

PGP ist ein Programm zum Verschlüsseln von E-Mails. Es wurde von Phil Zimmermann geschrieben, um Verschlüsselungssoftware vielen Menschen verfügbar zu machen (das Motto ist: „PGP - Cryptographie for the masses").
PGP bietet im wesentlichen zwei Operationen an: das Verschlüsseln und das Signieren von Botschaften.

PGP basiert auf einem (mittlerweile sogar zwei) Public-Key-Verfahren. Bei diesem Verfahren werden zwei Schlüssel verwendet: einer ist geheim und einer öffentlich. Der öffentliche Schlüssel sollte so weit wie möglich verbreitet werden, während der geheim möglichst geheim bleiben soll (d.h. es darf *niemand* Zugang

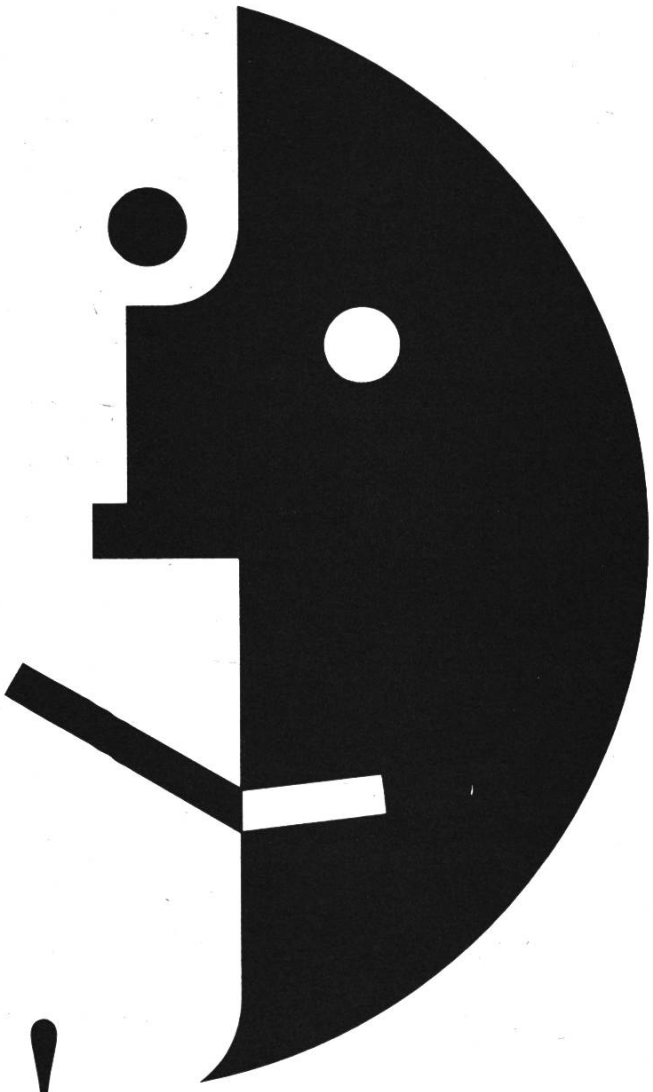
Nach o



Nac

Wir suchen die besten Köpfe. Also: Lust, mit hochkarätigen Leuten zusammenzuarbeiten? Und Unternehmen zu beraten.
kommen? Schreiben, faxen oder telefonieren Sie uns doch. Price Waterhouse Management Consultants, Stampfenbach

ben?



h oben!

vie sie noch besser werden können? Und dabei selber voranzu-
trasse 109, 8035 Zürich, Telefon 01 365 69 45, Fax 01 365 62 40.

Price Waterhouse
Management Consultants

zu diesem Schlüssel erhalten).

Für eine Signatur wird nun die Botschaft mit dem geheimen Schlüssel verschlüsselt (um ganz genau zu sein, wird nur eine Art Prüfsumme der Botschaft verwendet). Diese verschlüsselte Botschaft kann nun mit dem öffentlichen Schlüssel von jedem gelesen werden. Da nur der „Besitzer" eines Schlüssels Zugang zu dessen geheimen Schlüssel hat, ist nun sichergestellt, dass die Botschaft wirklich von ihm stammt, da niemand anders die Signatur hätte erzeugen können.

Zum Verschlüsseln wird die Botschaft mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Da nur dieser den Zugang zum zugehörigen geheimen Schlüssel hat, kann nur dieser die Botschaft lesen.

Nun stellt sich die Frage: warum soll ich meine E-Mail verschlüsseln? Ich habe doch nichts zu verbergen?

Ich spare mir hier die Antwort und überlasse Phil Zimmerman das Wort (aus der Dokumentation von PGP 2.6.x): [Diesen Teil kann man vielleicht in den Visionen irgendwie kursiv setzen, oder sonstwie klarmachen, dass er ein Zitat ist.....?]
It's personal. It's private. And it's no one's business but yours. You may be planning a political campaign, discussing your taxes, or having an illicit affair. Or you may be doing something that you feel shouldn't be illegal, but is. Whatever it is, you don't want your private electronic mail (E-mail) or confidential documents read by anyone else. There's nothing wrong with asserting your privacy. Privacy is as apple-pie as the Constitution.

Perhaps you think your E-mail is legitimate enough that encryption is unwarranted. If you really are a law-abiding citizen with nothing to hide, then why don't you always send your paper mail on postcards? Why not submit to drug testing on demand? Why require a warrant for police searches of your house? Are you trying to hide something? You must be a subversive or a drug dealer if you hide your mail inside envelopes. Or maybe a paranoid nut. Do law-abiding citizens have any need to encrypt their E-mail?

What if everyone believed that law-abiding citizens should use postcards for their mail? If some brave soul tried to assert his privacy by using an envelope for his mail, it would draw suspicion. Perhaps the authorities would open his mail to see what he's hiding. Fortunately, we don't live in that kind of world, because everyone protects most of their mail with envelopes. So no one draws suspicion by asserting their privacy with an envelope. There's safety in numbers. Analogously, it would be nice if everyone routinely used encryption for all their E-mail, innocent or not, so that no one drew suspicion by asserting their E-mail privacy with encryption. Think of it as a form of solidarity.

Today, if the Government wants to violate the privacy of ordinary citizens, it has to expend a certain amount of expense and labor to intercept and steam open and read paper mail, and listen to and possibly transcribe spoken telephone conversation. This kind of labor-intensive monitoring is not practical on a large scale. This is only

done in important cases when it seems worthwhile.

More and more of our private communications are being routed through electronic channels. Electronic mail is gradually replacing conventional paper mail. E-mail messages are just too easy to intercept and scan for interesting keywords. This can be done easily, routinely, automatically, and undetectably on a grand scale. International cablegrams are already scanned this way on a large scale by the NSA.

Versionenärger

Im letzten Jahr wurde die Version 5.0 als Nachfolgeversion von PGP 2.6.x vorgestellt. Aus verschiedenen Gründen wurde ein neuer Verschlüsselungsalgorithmus eingeführt (Diffie-Hellmann statt RSA). Das führt dazu, das neue Schlüssel nicht mehr von alten PGP Versionen gelesen werden können. Da die (bisherigen) RSA Schlüssel sehr weit verbreitet sind, ist das ein grosses Problem.
Die (alten) RSA-Schlüssel bieten den Vorteil, dass sie mittlerweile auf fast jeder Plattform verfügbar sind, und zudem der Source-Code sehr gut bekannt und auf Fehler (d.h. Sicherheitslücken) untersucht ist. Ausserdem sind die verwendeten Algorithmen sehr gut untersucht (und für sicher befunden worden).
Die neuen Schlüssel sind nur für Benutzer von PGP 5.x verwendbar, und ausserdem noch nicht auf allen Plattformen verfügbar (wobei für Wintel und Mac mittlerweile sogar ein GUI existiert, was der Verbreitung von PGP 5.x nicht schaden sollte...). Der Nachteil ist, dass weder der Source-Code noch die verwendete Variante des Verschlüsselungsalgorithmus bisher gut geprüft wurden.

Es bleibt somit jedem selber überlassen, welchen Schlüssel er (oder sie) verwenden möchte. Sollte man schon einen RSA-Schlüssel besitzen, so ist es vielleicht langsam an der Zeit, sich einen neuen zu generieren, ihn herumzureichen und unterschreiben zu lassen, damit man einen brauchbaren Schlüssel hat, wenn man auf PGP 5.x umsteigen wollen sollte.
Hat man noch gar keinen Schlüssel, so kann man eigentlich nur empfehlen, mit einem RSA-Schlüssel zu beginnen, und evtl. auch gleich noch einen (neuen) DH-Schlüssel zu generien (aus den oben erwähnten Gründen).

Was ist das Web of Trust ?

Bei jedem Public Key Verfahren ist die Authentizität ("Echtheit") der Public Keys ein Problem. Ein kleines Beispiel: Ich erhalte von Germano Caronni eine elektronisch unterschriebene Mail. Um die **Unterschrift zu prüfen**, brauche ich seinen *public key*. Eine Mail an *pgp-public-keys@keys.ch.pgp.net* mit dem Subject *GET caronni@tik* liefert mir seinen *public key*. Aber ist das wirklich **sein** public key? Jemand (ein Betrüger) könnte einen Key mit dem Namen Germano Caronni erzeugt und auf dem Keyserver deponiert haben.
Ich muss nun die **Echtheit dieses *public keys* überprüfen**. Dazu habe ich verschiedene Möglichkeiten, z.B:

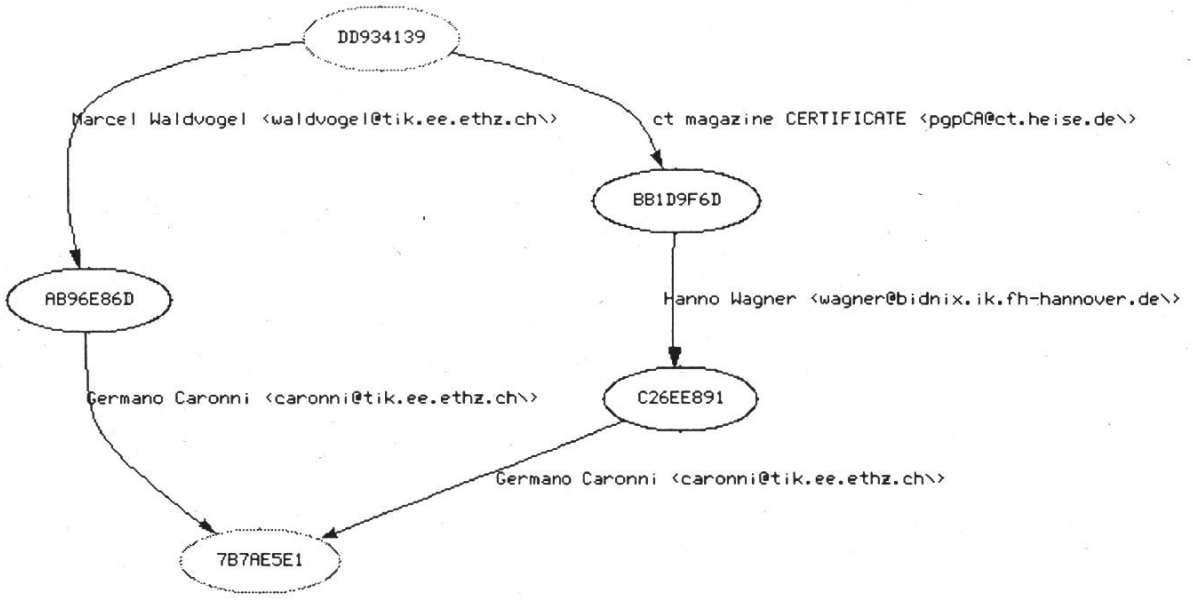
1. Ich suche im offiziellen Telefonbuch nach seiner Telefonnummer, rufe ihn an

und lasse mir von ihm seine **Schlüsseldaten** (siehe Glossar) geben. Dazu muss ich ihn aber kennen (Stimmenidentifikation), denn es könnte sich ja eine andere Person als Germano Caronni ausgeben.

2. Ich gehe bei ihm vorbei, lasse mir seinen Ausweis zeigen und die Schlüsseldaten geben.

3. Ich schaue, ob jemand den ich kenne, seine Identität elektronisch bestätigt.

Die ersten beiden Möglichkeiten sind aufwendig und zum Teil undurchführbar (wenn der Key z.B. einer Person weit weg gehört). Die Dritte ist dagegen relativ einfach. Dazu gibt es sogar eine Hilfe, den **AT&T PathServer**, erreichbar unter http://www.research.att.com/~reiter/PathServer/. Dort kann ich die Schlüssel-ID meines *public keys* (DD934139) eingeben und die Schlüssel-ID des *public keys* von Germano (7B7AE5E1). Als Resultat erhalte ich dieses Bild:



Paths from Key 0xDD934139 to Key 0x7B7AE5E1

Daraus ersehe ich: Marcel Waldvogel bestätigt, dass der Schlüssel 7B7AE5E1 Germano Caronni gehört (Marcel hat den Schlüssel von Germano unterschrieben). Ich wiederum habe den Schlüssel von Marcel Waldvogel unterschrieben und bestätige, dass der Schlüssel AB96E86D ihm gehört. Daneben gibt es noch einen weiteren Pfad über das ct'magazin und Hanno Wagner.

Damit solche Pfade existieren können (und möglichst kurz und disjunkt sind), müssen möglichst viele Schlüsselinhaber die Schlüssel anderer Schlüsselinhaber unterschreiben und diese Unterschriften auf den Keyservern (siehe Glossar) veröffentlichen, damit Dritte diese Bestätigungen auch nutzen können. Das Ganze wird dann **web of trust** genannt.

Warum müssen die Pfade **kurz und disjunkt** sein? Die Pfade sind Ketten von Bestätigungen. Je länger der Pfad zwischen meinem Schlüssel und dem Schlüssel vom Germano ist, desto weniger sicher kann ich mir über die Echtheit des Schlüssel von

Germano sein. Pfade, die keinen gemeinsamen Schlüssel zwischen dem Start- und Endschlüssel haben, sind **disjunkt**. Je mehr disjunkte Pfade existieren, desto geringer ist die Wahrscheinlichkeit, dass ein ‚schwarzes Schaf' durch eine falsch ausgestellte Unterschriften eine Bestätigungskette vortäuscht, die so gar nicht existiert. (Durch mehrere Schlüssel, die von derselben Person stammen, ist das dennoch möglich).

Die PGP-Signing-Party

Was ist die PGP-Signing-Party?
An der Signing-Party kommen (möglichst viele) Besitzer von PGP-Schlüsseln zusammen, die sich gegenseitig ihre Identität beweisen, und sich im Anschluss daran ihre Schlüssel gegenseitig unterschreiben. Dies ist ein ausgezeichneter Mutliplikator für das Web of Trust.

Wie wird die Party ablaufen?
An der Party wird eine Liste mit den Schlüsseln aller anwesenden Personen ausgeteilt. Dann muss jeder die Daten auf der Liste mit den wirklichen Schlüsseldaten und den Identitäten der Besitzer abgleichen. Das genaue Vorgehen wird auf dem Web und http://www.vis.inf.ethz.ch/pgp/kspa/199806 erklärt.

Wer darf teilnehmen?
Jeder.

Wann und wo?
Da an der letzten Party zu viele da waren, gibt es diesmal drei Termine: von Montag, den 22.6.98 bis einschliesslich Mittwoch, den 24.6.98 wird an jedem Tag um 16:00 Uhr im IFW A36 eine Party stattfinden. Es wird fünf Leute geben, die auf alle diese Partys gehen und sie auf diese Weise zusammenhängen. Zusätzlich dazu werden maximal 25 andere Leute kommen (dürfen).

Wie muss ich mich anmelden?
Schicke Deine(n) PGP-Schlüssel an mich (nicky@vis.inf.ethz.ch) und sage mir, an welchem Tag Du gern kommen möchtest. Ich werde Dir dann eine Bestätigung des Termins zuschicken. Damit bist Du angemeldet.

Was muss ich mitbringen? Deinen PGP-Fingerprint und einen amtlichen Ausweis (die *Legi ist nicht ausreichend!).*

Wo gibts mehr Informationen?
Genauere Informationen zu PGP und zur Key-Signing-Party finden sich auf dem Web unter
http://www.vis.inf.ethz.ch/pgp
(allgemeine PGP - Informationen) und **http://www.vis.inf.ethz.ch/pgp/kspa/199806**
(Informationen zur Key-Signing-Party).

Auf diesen Seiten gibt es auch eine Einführung in die Benutzung von PGP mit einer detailierten Anleitung zum (womöglich ersten) Erstellen eines eigenen Schlüssels - ab sofort auch für PGP 5.x.

Hier gibt es auch Anleitungen zum Einbinden von PGP in verschiedene Mailtools.

Für nähere Informationen kann ich nur die weiterführenden Links auf dieser Seite empfehlen.

SAP COLLEGE

NACHDIPLOMSTUDIUM 1998

2. STUDIENBEGINN: MONTAG, 10. AUGUST

Mit SAP R/3 in eine faszinierende Zukunft.

Holen Sie sich im SAPCollege, was der dynamische Markt fordert. Dass weltweit immer mehr Unternehmen ihre Position mit Standard-Software von SAP stärken, ergibt Berufschancen, an die Sie vielleicht bis jetzt noch nicht gedacht haben: Lernen Sie in 80 Tagen, mit den führenden Software-Werkzeugen von SAP umzugehen, und starten Sie Ihre Laufbahn mit Vorsprung.

Ich interessiere mich für das SAPCollege:

❒ Bitte senden Sie mir Unterlagen.

❒ Bitte rufen Sie mich an.

Name _____

Vorname _____

Adresse _____

Ort _____

Telefon _____

Administration und Anmeldung

SAP (Schweiz) AG
SAPCollege
Leugenestrasse 6
2504 Biel
Telefon 032 344 73 79
Telefax 032 344 71 32

E-Mail: training.switzerland@sap-ag.de
http://www.sap.com/swiss

BUSINESS SOFTWARE FOR YOU

SAP®

Vorlesungsumfrage WS 1997/98

mp/rb. *Wie sich einige vielleicht noch erinnern mögen, führten wir am Ende des letzten Semesters eine Vorlesungsumfrage durch. Wegen technischen Problemen und Zeitmangel (auch wir hatten Prüfungen) hat sich die Auswertung leider etwas verzögert. Mit über 750 Abgaben konnten wir einen neuen Rekord verbuchen. Das sind ca. 100 Abgaben mehr als bei der letzten Durchführung im WS96/97. Diese Steigerung haben wir hauptsächlich den Leuten im Grundstudium zu verdanken, aus dem Fachstudium gab es leider nur wenige Bewertungen.*

Als kleinen Anreiz zur Teilnahme haben wir dieses Mal einige Preise für die Teilnehmer ausgesetzt. Die Gewinner sind:

1. Preis (50.- Büchergutschein)
Werner Mueller
2. Preis (30.- Büchergutschein)
Peter Matter
3. Preis (50MB Quota oder ein VIS T-Shirt)
Christian Rupp
4. Preis (VIS T-Shirt oder 40MB Quota)
Nikolaos Kaintantzis
5. Preis (VIS T-Shirt oder 40MB Quota)
Patrick Bichler
6. Preis (VIS T-Shirt oder 40MB Quota)
Samuel Isenegger
7. Preis (VIS T-Shirt oder 40MB Quota)
Christian Schaer
8. Preis (VIS T-Shirt oder 40MB Quota)
Andreas Stauder
9. Preis (30MB Quota oder VD-Bündel)
Stefan Walthert
10. Preis (20MB Quota oder VD-Bündel)
Stefan Hilzinger

Herzliche Gratulation. Die Preise können während den üblichen Präsenzzeiten im VIS-Büro abgeholt werden.

Die Vorlesungen wurden im allgemeinen positiv bewertet. Begleitende Web-Seiten scheinen ebenfalls sehr beliebt zu sein. Wo es sie gab, wurden sie durchwegs als hilfreich kommentiert, bei den andern wurde deren Erstellung angeregt.
Die Assistenten wurden, anders als die Professoren, sehr unterschiedlich beurteilt. Es gab alles, von hochnäsigen Assistierenden bis zu Musterassistenten. Einige Dozierende sollten vielleicht ihre Helfer etwas besser aussuchen.

Nun aber zu den einzelnen Vorlesungen:

Grundstudium:
Informatik I (91 Abgaben)
Von den meisten wurde diese Vorlesung als die beste des ersten Semester bezeichnet. Prof. Widmayer macht eine interessante, gut verständliche Vorlesung, die den Lernzielen entspricht und hilft zu Verstehen. Die Tafeldarstellung dagegen wird als chaotisch bezeichnet (mühsam daraus Notizen zu machen) und ein ausführlicheres Skript wäre wünschenswert.

Logik (73 Abgaben)
Logik war auch eine interessante Vorlesung. Einige wünschten sich aber die Benutzung des Mikrofons. Die Übungsserien gehörten zu den interessantesten.

Algebra I (57 Abgaben)
Prof. Nipp's Vorlesung wurde mehrheitlich als langweilig eingestuft. Folien wurden zu schnell aneinandergelegt und es wurden mehrheitlich Beweise vorgestellt als Beispiele.

Analysis I (35 Abgaben)
Die Vorlesungen beider Professoren waren interessant und hatten eine gute Tafeldarstellung. Die Übungen waren auch verständlich. Die Vorlesungsunterlagen waren leider wenig und schlecht verständlich, die WWW Seite aber, und die dort verfügbaren Musterlösungen, wurden als vorbildlich bezeichnet.

Physik I (44 Abgaben)
Physik war wieder mal für die meisten mühsam und setzte zu hohe Kenntnisse voraus. Die Unterlagen waren zuwenig ausführlich und schlecht verständlich. Die Übungen brauchten einen grossen Aufwand und waren schwierig, halfen aber, die Vorlesung zu verstehen. Der Sinn von soviel Physik bei Informatikern wurde des öftern in Frage gestellt. Mit dem Verlegen auf das erste Jahr wurden leider noch nicht die Mathematische Voraussetzungen neuevaluiert. Eine bessere Koordination mit den Analysis Professoren ist dringend erwünscht.

Informatik III (70 Abgaben)
Wie Informatik I war auch die Vorlesung von Prof. Norrie ziemlich interessant, relevant und verständlich. Obwohl auf Englisch gehalten, hatten die Studierende keine Mühe mit der Sprache.

Systemprogrammierung (88 Abgaben)
Prof. Gross war bei seinen Folien eher unübersichtlich, speziell wenn mehrere Folien aufeinandergelegt wurden. Das Tempo war schnell und damit auch anspruchsvoll. Prof. Stricker war dagegen angenehmer. Die Vorlesung war für die meisten Studierenden interessant.

Information und Kommunikation (61 Abgaben)
Die Vorlesung wurde als interessant, klar gegliedert und verständlich eingestuft. Der Zeitdruck war aber in der Vorlesung klar zu spüren. Es wäre vielleicht besser, das ganze wieder in zwei Vorlesungen zu spalten.

Elektrotechnik (53 Abgaben)
Prof. Vahldieck setzte eher zu hohe Voraussetzungen und war schlecht verständlich. Bei Prof. Weiler war das besser. Der rote Faden war in der Vorlesung ersichtlich, die Übungen waren aber zu schwierig.

Numerisches und symbolisches Rechnen (46 Abgaben)
Die Vorlesung wird als schwierig und mit hohen Voraussetzungen bezeichnet. Prof. Gruntz macht selbst einen kompetenten Eindruck, das Problem ist der äusserst trockene Stoff. Kritisiert wurde auch der grosse Einsatz von Maple, was die Übungen Prüfungsirrelevant und schwierig macht.

Kernfächer:
System-Software (27 Abgaben)
Im grossen und ganzen eine interessante Vorlesung mit einem ersichtlichen Konzept. Es wurde aber nicht immer genügend auf die Details eingegangen. Unterlagen waren wie immer nicht existent. Die Übungen waren aufwendig und

manchmal nicht gut mit der Vorlesung koordiniert.

Informationssysteme (17 Abgaben)
Prof. Schek war verstdndlich und die Vorlesung gut gegliedert. Es gab zu Beginn einige Schwierigkeiten, da die Vorlesung zum ersten mal gegeben wurde. Die häufigen Fehler waren ziemlich mühsam, und die Übungen
verlangten grossen Aufwand.

Wissenschaftliches Rechnen III (19 Abgaben)
Eine gute Vorlesung, obwohl manchmal der rote Faden nicht ersichtlich war. Ein Skript wäre noch wünschenswert.

Vertiefungen:
Informationssicherheit und Kryptographie (15 Abgaben)
Die Vorlesung von Prof. Maurer wurde als ziemlich interessant bewertet. Der rote Faden war ersichtlich und das gute Skript rundet das ganze ab. Bei den Übungen waren die Meinungen geteilt: einige wünschten wieder Musterlösungen und andere nicht.

Interprozess-Kommunikation in UNIX (6 Abgaben)
Von den meisten als interessant und mit einen klar ersichtlichen roten Faden bewertet. Der Dozent sei kompetent und mit Praxisbezug, die Assistenten engagiert. Aber der Stoff sei auch zuviel.

Ergänzungen:
Informatik Projektentwicklung (9 Abgaben)
Die Vorlesung wurde als interessant und mit klarem Konzept eingestuft. Prof. Zehnder wirkte kompetent und die praxisnahen Beispiele sind lehrreich.

Anwendungen:
Bei „Technologietransfer / -strategien" waren es insgesamt zwei Informatiker, bei „Projektführung und -abwicklung in der Praxis" gerade mal vier. Der einzige, der beide Vorlesungen besuchte, war ich selbst (mp). Ich finde es wirklich schade, dass dieser Teil so wenig besucht wird und die „Anwendungsentwicklung" dieses Semester gestrichen wurde. Prof. Zehnder wurde gelobt für seine praxisnähe und die interessanten Beispiele. Dr. Weydert's Projektführung war aber eine Fortsetzung, die mindestens so interessant war. Es wurden mehrmals Referenten eingeladen, die über ihre Erfahrungen sprachen. Eine ganze Doppelstunde wurde sogar dem Konfliktmanagement gewidmet, was normalerweise nicht so gern angesprochen wird, aber nichtsdestotrotz einen wichtigen Punkt für den Erfolg oder Misserfolg eines Projektes darstellt.
Technologietransfer / -strategien war nicht so informatiknah (über 50 Studierende, meistens IIIE), deswegen gab es auch einen sehr guten Ueberblick ueber die Technologie (schliesslich ist Informatik selbst auch nichts weiteres als eine Technologie).

Ab nächstem Jahr ist aber soweit, dass mindestens eine Anwendung obligatorisch ist. Die Einführung ist leider auch nicht ganz unproblematisch bringt sie doch einige Studienpläne ins Wanken. Bei den neuen Studierenden im Fachstudium wird es aber sicher hilfreich sein.

The DirectX SDK

If you play computer games under Windows, you have probably not gotten around installing DirectX on your computer. DirectX is a technology by Microsoft which was introduced as an add-on to the Windows operating system to facilitate the development of multimedia applications. (And to get game programmers to stop developing for MS-DOS.) Currently, DirectX 5 is available for both Windows 95 and NT, running on 0x86 processors. DirectX 6 is now in beta, and will be available soon.

Overview

DirectX is based on Microsoft's Component Object Model (COM). COM is a system for creating and managing objects, their instances, and their various interfaces at operating system level. For an exhaustive explanation, see the book **Inside COM** by Microsoft Press. You do not have to understand COM to be able to use DirectX, but it helps to know a little about it to understand why you have to do certain weird things.

DirectX contains a set of APIs which help with different aspects of an application:

- DirectDraw's primary purpose is to permit direct frame-buffer access under Windows, which has not been possible under the GDI. While DirectDraw provides much important functionality beyond this, it is important to realize that it is not a graphics library. For example, it has no routines for drawing lines. This is a general concept of DirectX as a whole: It contains important core functionality which a developer cannot create using existing methods, but little beyond that. DirectDraw transparently takes advantage of acceleration by 2D display hardware. Most features which are not supported by the installed display adapter are emulated by DirectDraw in software (HEL). This is generally true for all DirectX APIs, and is another reason to use them, even for tasks which the programmer could theoretically implement more efficiently than it is done in the HEL.
- DirectSound was originally conceived to provide low latency sound for action games, but in DirectX 5 it has been expanded to provide support for 3D sound hardware.
- DirectInput's original purpose was to circumvent the slow and tedious Windows WM_ mouse and keyboard messages when writing apps that need fast input response. DirectX 5 provides support for all types of input devices, and includes force feedback support.
- DirectPlay provides programs with a means to communicate by sending packets of data between computers. It includes drivers for various types of networks, as well as serial and modem connections. The advantage of using DirectPlay, instead of, say, Winsock to write an app's networking code, is that it will support all of the communication media above through a single piece of code, instead of only providing TCP and UDP support. Moreover, DirectPlay makes it easy to create and organize

multiplayer games over the Internet through the ability to easily set up a lobby server where players can meet, chat and see the games in progress.

- Direct3D is perhaps the most exciting and fastest growing part of DirectX. Its main purpose is to provide a standardized interface to the large selection of 3D hardware which has recently become available. In that sense it is similar to Silicon Graphics' OpenGL. While there are proprietary libraries put out by the hardware manufacturers themselves, for example 3dfx I.'s Glide SDK, they only work with the cards of their respective companies. Some manufacturers, like nVidia, have decided not to develop a proprietary SDK, and their hardware is only accessible by Direct3D and OpenGL through the appropriate drivers. Direct3D, like DirectDraw, emulates all core functionality in software, making it theoretically possible to run Direct3D applications without a 3d card.
- There are other, more recent additions to DirectX, for example DirectMusic and DirectShow, as well as trivial components like DirectSetup, which I will not discuss now.

As one can see, DirectX is very much computer game oriented, though much of its functionality can be successfully utilized for scientific projects more becoming of a serious institution like the ETH, where of course nobody has time to write silly games.

Direct3D programming

Since Direct3D and the hardware it supports are such a hot pieces of technology, let us take a look at how one goes about writing a 3d accelerated application. The first decision one has to make is whether one wants to work in Retained (RM) or Immediate (IM) mode.
Retained mode programming can be best described as using trueSpace for Oberon. One creates, configures and positions lights, cameras, and 3d mesh hierarchies by making individual function calls from the program. One applies textures to materials, and materials to meshes; one sets up projections and shadows. While RM provides a certain degree of low level methods, for example the accessing of individual polygons, all of its high level abstractions take considerable processing power. If one wants unlimited access to the 3d data set, as well as uncompromising speed, IM is the only way to go. IM works at a very low level. The primary way of using IM is through its DrawPrimitive methods. This set of functions provide a way to send a stream of polygons (primitives) to the 3d renderer. One changes textures, and sets material properties in general through renderstate changes, which occur between calls to the DrawPrimitive methods. All high level operations, for example organizing meshes into hierarchies, are left to the programmer in IM. Lighting under IM can either be done by the programmer or left to Direct3d.

Since IM programming is some orders of magnitude more difficult than RM, we will examine RM programming below. To be able to program using DirectX in general, one needs to obtain the DirectX SDK. You can either download it from MS' site (caution: its BIG!) or sign up to be a beta tester and have the CD sent to you for free (see address below, beware: you will need to sign an NDA first.). The really easy way is to email me and I will lend you my spare CD. While DirectX supposedly works with all major win32 development environments, it is best enjoyed with MS Visual C 5, for obvious reasons. Moreover, since COM is object oriented, I suggest using C++, which makes the syntax a bit simpler, not to mention the other, non DirectX related advantages.

I am going to assume that you know some basic Windows programming, a definite prerequisite to learning DirectX, and that you can write a Windows program that does little besides the following 3 things: Call the function Startup() just before it goes into idle mode after initializing itself, call Tick() at regular intervals, and call Shutdown() before the program terminates. You will of course also want to watch for some keystroke to decide when its time to quit. Once this is done, we will define the above 3 functions. The program will display an animated 3d model, and rotate it on the screen. Simple enough. We will first need an animated 3d model. You can create one in your favorite modeller (I use 3d Studio), export it as a .3ds file, and export its texture (we will use only one texture to keep things simple) as a .BMP. You need to convert the 3ds file to RM's native .X file format using conv3ds, a utility that comes with the SDK. If you don't feel like modeling, you can download the sources to this example from my site, which includes an .X file and its texture.

I am putting all of the DirectX related code into a file called DxStuff.cpp. Below is the top part of the file, which includes the global data definitions. You will have to link your program with ddraw.lib, dxguid.lib and d3drm.lib.

```
#include <ddraw.h>
#include <d3drmwin.h>

#define TRY(a) if (FAILED(a)) return
true;

  //Windows stuff:
HWND        hWindow;
HINSTANCE   hProg;

  //DirectDraw stuff:
LPDIRECTDRAW fpDD=NULL;
LPDIRECTDRAWSURFACE
ddsFrameBuffer=NULL,ddsBackBuffer=NULL;
DDSURFACEDESC ddsd;

  //D3D stuff:
LPDIRECT3D lpD3D=NULL;
LPDIRECT3DRM lpD3DRM=NULL;
LPDIRECT3DRMFRAME FScene=NULL,
  // Master frame in which others are
placed
FMain,FCamera;
LPDIRECT3DRMDEVICE
lpD3DRMDevice=NULL;
   //this represents the buffer we
render to
LPDIRECT3DRMVIEWPORT ViewPort;
LPDIRECT3DRMANIMATIONSET
AnimSet=NULL;
  //this is the loaded animated actor
```

Lets define the startup function first. Your program needs to call this function at startup with the program instance which is an argument of WinMain, as well as a pointer to your message processing callback function. The application should abort (call ShutDown() first!) if this function returns true, which it does through the TRY() macro defined above.

```
bool StartUp(HINSTANCE hProga,
WNDPROC WndProc)
{
```

First, we need a full screen window. We will not support windowed mode for the sake of brevity, plus cards like the 3dfx can only run in full screen mode.

```
WNDCLASS    Wcl;
hProg=hProga;
Wcl.hInstance=hProg;
Wcl.lpszClassName=ClassName;
Wcl.lpfnWndProc = WndProc;
Wcl.style=0;

Wcl.hIcon=LoadIcon(NULL,IDI_APPLICATION);
Wcl.hCursor =
LoadCursor(NULL,IDC_ARROW);
Wcl.lpszMenuName=NULL;
Wcl.cbClsExtra = 0;
Wcl.cbWndExtra = 0;
Wcl.hbrBackground=(HBRUSH)
GetStockObject(NULL_BRUSH);
if (!RegisterClass(&Wcl))
return true;
hWindow=
CreateWindowEx(0,ClassName,"DxExample",
WS_VISIBLE  | WS_POPUP,

0,0,GetSystemMetrics(SM_CXSCREEN),
GetSystemMetrics(SM_CYSCREEN),
NULL,NULL,hProg,NULL);
if (hWindow==NULL)
return true;
UpdateWindow(hWindow);
```

Next, we need to start DirectDraw, which is used by Direct3D. The Choose2DDriver function returns a pointer to a GUID which represents the display device which we will use. Writing a good Choose2DDriver function (either making a decision based on device information, or listing the available drivers in a dialog for the user to choose from) is a bit involved, so I will skip that

here. You should replace this with NULL (default choice) until you are ready to write such a function.

```
DDSCAPS             ddscaps;
TRY(DirectDrawCreate(Choose2DDriver(),
&fpDD, NULL_))
```

Now we have to tell DirectDraw how we intend to use it. We say that the program has exclusive access to the frame-buffer, and that we are of course full screen.

```
TRY(fpDD-
>SetCooperativeLevel(hWindow,
DDSCL_EXCLUSIVE |DSCL_FULLSCREEN
|DDSCL_ALLOWREBOOT))
```

We then set the display mode to 640x480, 16 bits, a setting which all cards support.

```
TRY(fpDD-
>SetDisplayMode(640,480,16))
```

Next, we create an abstraction for the frame-buffer. This is basically the part of video memory which you see on your screen. With the same call, we also request a back buffer, which is where we will render our scene to. DirectDraw refers to these bitmaps as 'surfaces'. You need to be careful about something here: Most 3d hardware can only render to video memory, so you need to create the back buffer appropriately. However, if you only have 1meg of video ram, you will not be able to do double buffering. I can't afford to go into a lot of details about this, so I will just say that when you go from 3d emulation to hardware, you will need to move many things (textures, zbuffers) from system to video memory, so watch out.

```
ddsd.dwSize = sizeof(ddsd);
ddsd.dwFlags= DDSD_CAPS |
DDSD_BACKBUFFERCOUNT;
ddsd.ddsCaps.dwCaps=
DDSCAPS_PRIMARYSURFACE |

DDSCAPS_FLIP |   DDSCAPS_COMPLEX |

DDSCAPS_3DDEVICE;
  // Create a D3D compatible surface
ddsd.dwBackBufferCount  = 1;

TRY(fpDD->CreateSurface      (&ddsd,
&ddsFrameBuffer, NULL))
  //get a pointer to the back buffer
ddscaps.dwCaps = DDSCAPS_BACKBUFFER;
TRY(ddsFrameBuffer-
>GetAttachedSurface (&ddscaps,
&ddsBackBuffer))
```

Now we start up Direct3D Retained Mode. Under COM, you need to ask DirectDraw for a special interface to Direct3D, which is what the QueryInterface() call does. Then we need to create a bunch of frames. Frames in D3D are basically coordinate systems which can be linked into hierarchies. A frame may contain visuals such as meshes, as well as lights. They may even act as a camera. Frames also store general data about the scene, like the sort order as well as the z buffer option I set below. Child frames automatically inherit these settings. Below, I create a top level frame, as well as a two light frames and a camera.

```
LPDIRECT3DRMFRAME
FLightAmbient=NULL,FLightDirect=NULL;
  //light frames
LPDIRECT3DRMLIGHT LAmbient,LDirect;

TRY(fpDD-
>QueryInterface(IID_IDirect3D,
(void**) &lpD3D))

TRY(Direct3DRMCreate(&lpD3DRM))

TRY(lpD3DRM->CreateFrame(NULL,
```

```
&FScene))
  //create scene object
TRY(FScene-
>SetSortMode(D3DRMSORT_FRONTTOBACK))
  //for speed w. zbuffer
TRY(FScene-
>SetZbufferMode(D3DRMZBUFFER_ENABLE))

TRY(lpD3DRM->CreateFrame(FScene,
 &FLightAmbient))
TRY(lpD3DRM-
>CreateLightRGB(D3DRMLIGHT_AMBIENT,
0.2f,0.2f,0.2f, &LAmbient))
TRY(FLightAmbient-
>AddLight(LAmbient))
LAmbient->Release();
FLightAmbient->Release();

lpD3DRM->CreateFrame(FScene,
 &FLightDirect);
FLightDirect-
>AddRotation(D3DRMCOMBINE_AFTER,
0.0f,1.0f,0.0f,-0.7f);
  //horizontal
FLightDirect-
>AddRotation(D3DRMCOMBINE_AFTER,
0.0f,0.0f,1.0f,0.6f);
  //vertical

TRY(lpD3DRM-
>CreateLightRGB(D3DRMLIGHT_DIRECTIONAL,
0.3f,0.3f,0.3f, &LDirect))
FLightDirect->AddLight(LDirect);
LDirect->Release();
FLightDirect->Release();

TRY(lpD3DRM->CreateFrame(FScene,
 &FCamera))
TRY(FCamera->SetPosition(FScene,
 0.0f, 0.0f, -9.0f))
TRY(lpD3DRM-
>CreateFrame(FScene,&FMain))
TRY(FMain->SetPosition(FScene,0.0f,-
6.0f,0.0f))
```

Below, we create a 'device', which is basically an abstraction of the rendering surface we use as a target. The first argument is again a GUID identifying your 3D device driver. NULL is default once again, and once again, you should make a more intelligent choice later as with the 2D device. Next, we set a couple of options like the render quality.

```
TRY(lpD3DRM-
>CreateDeviceFromSurface(NULL,fpDD,
ddsBackBuffer,&lpD3DRMDevice))
TRY(lpD3DRMDevice-
>SetBufferCount(2))
  //we use double buffering
  // Render using gouraud shading
TRY(lpD3DRMDevice-
>SetQuality(D3DRMRENDER_GOURAUD))
TRY(lpD3DRMDevice->SetDither(FALSE))
TRY(lpD3DRMDevice-
>SetTextureQuality(D3DRMTEXTURE_NEAREST))
TRY(lpD3DRMDevice->SetShades(16))
  // Set the number of shades for
  lighting
TRY(FScene-
>SetSceneBackgroundRGB(0.1f,0.5f,0.3f))
```

Now we create a viewport, which is the area of the back buffer we will render to. You could make this the whole back-buffer if you wanted to. Note that we specify the FCamera frame as the point of view to render from.

```
TRY(lpD3DRM-
>CreateViewport(lpD3DRMDevice,FCamera,
20,20,400,400,&ViewPort));
ViewPort-
>SetProjection(D3DRMPROJECT_PERSPECTIVE);
```

Here is an example of how much work we save under RM. We just create an animation object, and load the appropriate file into it, which is actually a collection of frames and meshes, along with keyframe data. All this will be allocated as a subtree under the Fmain frame we created. Finally, we ask the camera frame to 'look at' our animation hierarchy.

```
TRY(lpD3DRM-
```

```
>CreateAnimationSet(&AnimSet))
TRY(AnimSet->Load("marine.x",  NULL,
D3DRMLOAD_FROMFILE,LoadTexturesCallBack,
NULL,FMain))
  //load the animset from file
TRY(FCamera-
>LookAt(FMain,FScene,D3DRMCONSTRAIN_Z))
return 0;
}
```

OK, the other two functions are a lot simpler; we look at Tick() next. Delta is the time elapsed since we were called the last time, in miliseconds. Hint: use timeGetTime(). This function simply advances the time of the animation (which will automatically loop when it reaches the end), and rotates the frame it is in, causing the whole thing to turn. The Flip() function displays the results on the frame-buffer.

```
void Tick(DWORD Delta)
{
static float CurrentTime=0.0f;
float fDelta=(float)Delta;
CurrentTime+=fDelta/64;
  //# controls speed of playback
AnimSet->SetTime(CurrentTime);
FMain-
>AddRotation(D3DRMCOMBINE_AFTER,
0.0f,1.0f,0.0f,fDelta* -0.001f);
ViewPort->Clear();
ViewPort->Render(FMain);
ddsFrameBuffer->Flip (ddsBackBuffer,
DDFLIP_WAIT);
}
```

The ShutDown function is straightforward: It releases all the resources we used.

```
void ShutDown(void)
{
if (AnimSet)       AnimSet-
>Release();
if (ViewPort)      ViewPort-
>Release();
if (FScene)        FScene->Release();
if (lpD3DRMDevice) lpD3DRMDevice-
>Release();
if (lpD3DRM)       lpD3DRM-
>Release();
if (lpD3D)         lpD3D->Release();

if (ddsFrameBuffer) ddsFrameBuffer-
>Release();
if (fpDD)          { fpDD-
>RestoreDisplayMode();  fpDD-
>Release (); }

UnregisterClass(ClassName, hProg);
}
```

This program will work even without a 3d card. In fact, it will probably not take advantage of it even if you have one, since we used the default setting for all of the drivers. To fix this, you will need to learn about Device enumeration. Your primary source for further information should be the SDK's documentation. Good luck!

Adam Moravanszky
admiral@swix.ch

This example program is available at:
http://www.swix.net/clan/admiral/

References

Inside COM by Microsoft Press
Microsoft's DirectX site:
http://www.microsoft.com/DirectX/default.asp

DirectX beta program; send email to:
directx@microsoft.com

Wir sind als Entwicklungsfirma im Bereich Echtzeitsysteme, digitale Signalverarbeitung und Multimedia tätig.
Wir beschäftigen rund 20 Ingenieure, die im Kundenauftrag Machbarkeitsstudien, Systemarchitekturstudien und Entwicklungen durchführen.
Als Betriebssysteme verwenden wir UNIX, Linux, Windows NT oder schreiben unser Betriebssystem selber.
Wir programmieren in C, C++, Fortran, Assembler oder schreiben automatische Codegeneratoren.
Wir setzen PC's und Workstations ein und bauen geeignete Interfaces oder entwickeln das Computersystem selber.

Im Rahmen unserer Projekte suchen wir laufend

Praktikantinnen und Praktikanten

Interessiert?
Rufen Sie uns an und kommen Sie vorbei.

Supercomputing Systems AG
Technoparkstr. 1
8005 Zürich
Tel: 01 445 16 00
Fax: 01 445 16 10
Internet: **http//www.scs.ch**

Kontaktperson:
Dr. Markus Schenkel, schenkel@scs.ch

(i) Programming is like sex: One mistake and you have to support for a lifetime.

(ii) Auf de CeBIT wird Microsoft Network vorgeführt. Das Standpersonal preist unter dem Schlagwort „Informationen at your Fingertips" den Zugang zum Wissen der Welt. Ein Besucher setzt sich an den PC und tippt ein: „Wo ist mein Vater?"
Antwort: „Er wohnt in München und arbeitet bei Siemens." Der Besucher schüttelt den Kopf: „So ein Blödsinn. Mein Vater ist seit 10 Jahren tot!" Antwort aus dem Datennetz: „Tot ist der Mann Ihrer Mutter. Ihr Vater wohnt in München und arbeitet bei Siemens."

(iii) Wie kannst Du eine Blondine in den Wahnsinn treiben?
Gib ihr einen Sack M&M's, und sag ihr, sie soll sie alphabetisch ordnen

(iv) Was ist das: ein Ferrari mit zwei Blondinen drinn?
ein DumDum-Geschoss.

(v) Weckt die Mutter ihren Sohn: ‚Steh auf, mein Junge, Du musst zur Schule gehen!"
Muss ich wirklich, Mutti?"
Aber ja, Du weisst doch, alle Dozenten müssen zur Schule!'

(vi) The Top Ten Ways Microsoft Would Change the Auto Business

10. People would get excited about the features of the latest Microsoft cars, forgetting that these same features had been available from other car makers for years.

9. Every time the lines of the road were repainted, you would have to buy a new car.

8. Occasionally, your car would die for no apparent reason and you would have to restart it. Strangely, you would just accept this as normal.

7. You could have only one person in the car at a time, unless you bought a Car 95 or Car NT—though you would have to buy ten more seats and a new engine.

6. Consumer would be under constant pressure to upgrade. Modestly priced upgrade kits will be available that could be installed by either the dealer or the user; support for upgrade self-installation would cost extra, based on the number of calls.

5. Sun Motorsystems would make a car that was solar-powered, twice as reliable, and five times as fast, but it would only run on 5 percent of all roads.

4. The oil, alternator, gas, and engine warning lights would be replaced by a single ‚General Car Fault' warning light.

3. The U.S. government would be forced to rebuild all of the roads for Microsoft cars (they will run on old roads, but very slowly).

2. We would all have to switch to Microsoft gas.

1. New seats would require everyone to have the same size rear end.

(vii) If Restaurants Functioned Like Microsoft:

Patron: Waiter!

Waiter: Hi, my name is Bill, and I'll be your Support Waiter. What seems to be the problem?
Patron: There's a fly in my soup!
Waiter: Try again, maybe the fly won't be there this time.
Patron: No, it's still there.
Waiter: Maybe it's the way you're using the soup; try eating it with a fork instead.
Patron: Even when I use the fork, the fly is still there.
Waiter: Maybe the soup is incompatible with the bowl; what kind of bowl are you using?
Patron: A SOUP bowl!
Waiter: Hmmm, that should work. Maybe it's a configuration problem; how was the bowl set up?
Patron: You brought it to me on a saucer; what has that to do with the fly in my soup?!
Waiter: Can you remember everything you did before you noticed the fly in your soup?
Patron: I sat down and ordered the Soup of the Day.
Waiter: Have you considered upgrading to the latest Soup of the Day?
Patron: You have more than one Soup of the Day each day??
Waiter: Yes, the Soup of the Day is changed every hour.
Patron: Well, what is the Soup of the Day now?
Waiter: The current Soup of the Day is tomato.
Patron: Fine. Bring me the tomato soup, and the check. I'm running late now.
Waiter leaves and returns with another bowl of soup and the check
Waiter: Here you are, sir. The soup and your check.
Patron: This is potato soup.
Waiter: Yes, the tomato soup wasn't ready yet.
Patron: Well, I'm so hungry now, I'll eat anything.
Waiter leaves.
Patron: Waiter! There's a gnat in my soup!

The check:
Soup of the Day—$5
Upgrade to new Soup of the Day—$2.50
Access to support—$10

(viii) How the World Would Be Different ...

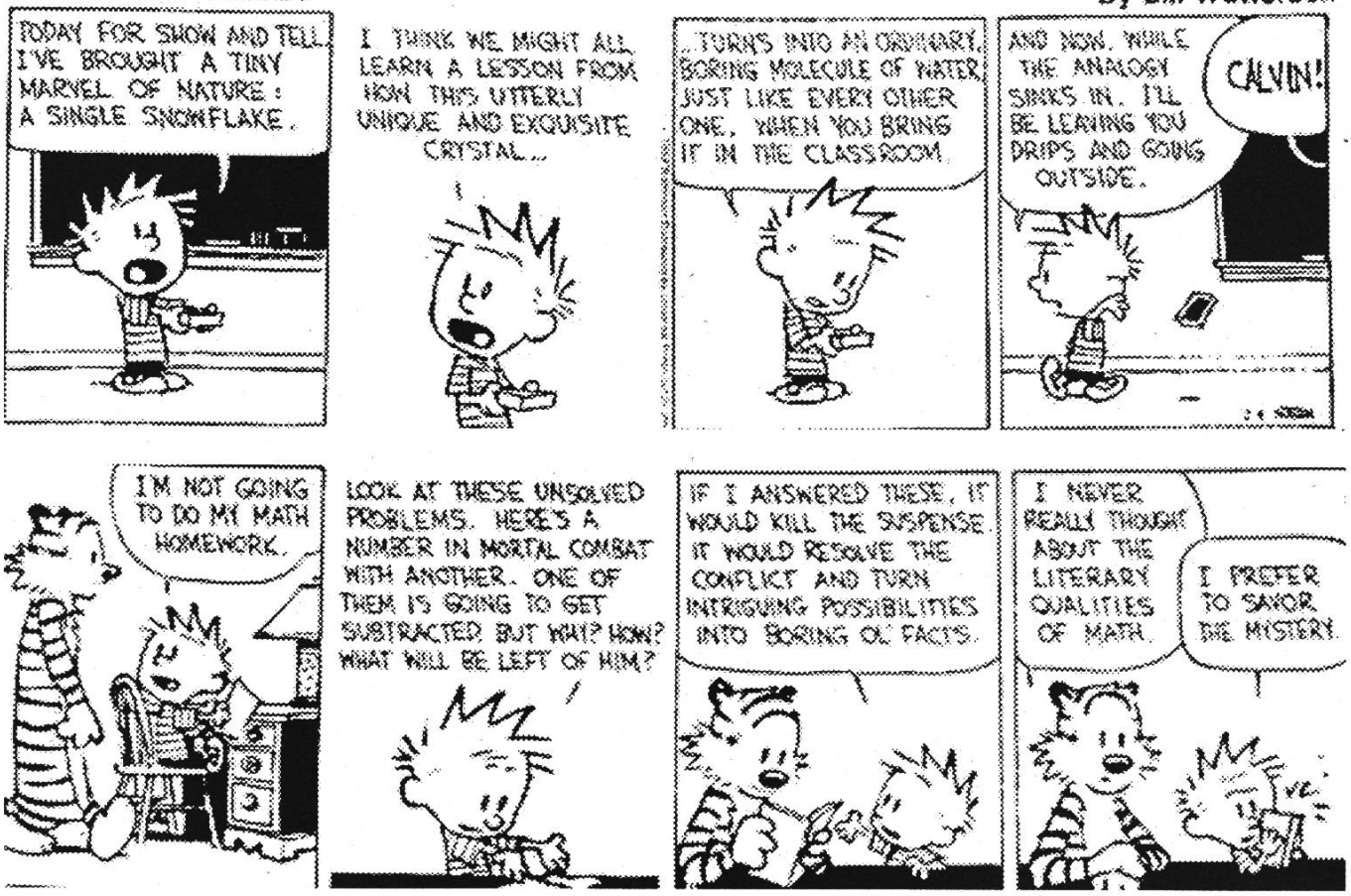
If IBM Made Toasters:
They would have one big toaster where people bring bread to be submitted for overnight toasting. IBM would claim a worldwide market for five, maybe six toasters.

If Apple Made Toasters:
It would do everything the Microsoft toaster does but would have been released five years earlier.

Does Digital Still Make Toasters?
They made good toasters in the seventies, didn't they?

< to be continued ... in visions 6.98 >

CALVIN AND HOBBES **By Bill Watterson**



Termine

Hier noch die wichtigsten Termine aus dem Sommersemester 1998

10. 6. 98
13h15 - 15h00; ML H44; Orientierungsveranstaltung der Abteilung IIIE, Abteilung für Betriebs- und Produktionswissenschaften. Für Studierende des 4. Semesters der Abteilung IIIA, IIIB, IIIC und IIID.

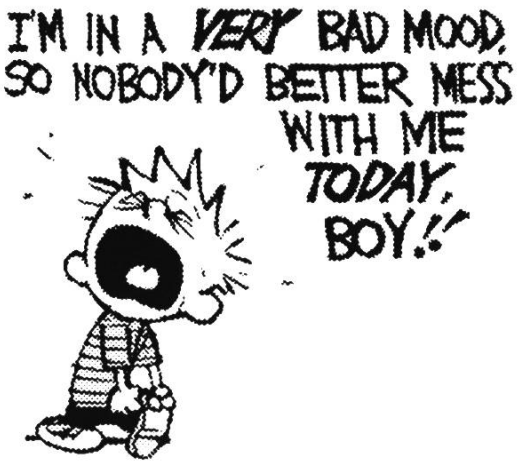
11. 6. 98
VISKAS - das traditionelle Fest des VIS am Katzensee

3. 7. 98
Ende SS 1998

Zukunftsvisionen

Redaktionsschluss der Juli Ausgabe:
16. Juni 1998

Erscheinungsdatum Juli Ausgabe:
1. Juli 1998



P. P. 8304 Wallisellen

Falls unzustellbar bitte zurück an:

Verein der Informatikstudierenden
IFW B29
ETH Zentrum

CH-8092 Zürich

Inhalt

Impressum / Editorial **2**
Vorwärts ins Mittelalter **3**
Labelinth in den Läden **5**
ZIP Drives im Rif/Raf **11**
Departements- / VIS Vorstandsseite **12**
Be OS **13**
Python - die High-Level-Scriptsprache **17**
Pretty Good Privacy **21**
Vorlesungsumfrage WS 1997/98 **29**
The DirectX SDK **32**
Humor **40**
Termine / Zukunftsvisionen **43**