Old circles
New algorithms

Go-faster sprites

Enhancing Windows 95

Writing help with Delphi & VB

Animating the Web

The impact gap

07

9 770268 687008

It never works
as well without
the middle.

C & C++ FOR WINDOWS

Comms
Async Pro for Win 2.1	£135
COMM-DRV/LIB 16.0	£105
Fax C++ DSK for Win16/95	£770
Greenleaf CommLib 5.2	£235

Compression
Crusher! Win DLL w/Source	£235
Greenleaf ArchiveLib 2.0	£210
PKWare Data Comp Lib for Win	£175
TCOMP/Multi-Platform 2.12	£105

Database
CodeBase 6.0	£270
CXBase Pro	£500
DBTools.h++ for ODBC	£1250
DISAM96 for DOS/Win	£715
Greenleaf Database Library 4.0	£180
List & Labels for Win (Pro)	£410
Objective Grid	£299
POET Personal SDK 3.0	£169
ProtoGen+ Client/Server NT	£1575
Raima DBM Engine+EADS 4.0	£740
Velocis + EADS (Offer)	£250
Visual SQL Intro	£725

Graphics - Charting
Charting Tools for Win 2.0	£180
Essential Chart for Win	£320
GraphiC/Win 7.0	£360
Graphics Server 4.0	£245
Real-Time Graphics Tools	£360

Graphics - Image Files
AccuSoft Image Lib/Win 5.0	£610
Ad Oculos (Image Analysis) 2.0	£325
Image SDK Plus for NT	£505
ImageKnife Pro 2.0	£280
ImageMan DLL 32 & 16-bit 5.0	£620
LEADTOOLS DLL Winit32 Pro	£895

Graphics & GUI
3d Graphics Tools 5 (32-bit C)	£230
ProtoGen+ NT 5.0	£445
WinGKS	£575
WinMaker Pro 6.0	£725
zApp	£575
Zinc Engine & Win16/32 Key	£634

Maths & Stats
IMSL C Numerical Libraries	£495
IMSL Math Module for C++	£495
Math.h++ 5.1.3	£495
Money.h++	£1075

Sundry Components
C++ Booch Components	£415
HeapAgent	£420
SEC++	£375
TG-CAD Prof 6.0	£770
Tools.h++ 7.0	£340
WinWidgets++	£240

Tools
CC-RIDER for Win16	£250
KPWin++	£600
Newi Solo Intro	£295
SOMobjects Dev Toolkit	£200
Visual Parse++	£289

JAVA

Java now has enough tools to have its own section! Serious developers should check out the RAD tools such as Café & JFactory. The official books should be published soon.

Borland C++ 5.0	£247
Borland C++ 5.0 Dev Suite	£340
Symantec Café Intro	£119
JFactory (Win32) Intro	£425

C & C++ FOR DOS

Comms
Essential Comm 5.0	£265
MagnaComm/DOS	£215
SilverComm "C" Asynch 4.02	£195

Database
c-tree Plus 6.5A	£565
SoftFocus Btree/ISAM	£75

Graphics & GUI
3D-Ware Prof	£99
Fastgraph 4.0 (Ted Gruber)	£195
GX Graphics 3.0	£155
MetaWINDOW-DOS 5.0	£215
Zinc Engine & DOS Key 4.1	£634

Maths & Scientific
C/Math Toolchest & Grafix	£45
Huge Virtual Array & NAT 3.0	£215
Science, Eng & Graphics Tools	£115

Screen
C/Windows Toolchest	£45
Greenleaf Datawindows 3.0	£225
TCXL UI for DOS/Win 6.2	£135

General & Systems Libraries
GX Sounds	£165
MTASK	£215
TTSR Ram Res Dev Sys 2.03	£100

Tools
C-DOC Pro 6.0	£240
C-Vision for C/C++ 4.0	£145
CodeCheck (Professional)	£475
PC-Lint for C/C++ 7.0	£135

VISUAL BASIC FOR DOS
MS Visual Basic for DOS Std	£100
MS Visual Basic for DOS Prof	£240
db/Lib Prof 3.0	£195
Graphics Workshop	£105
PDQComm 2.62	£65
ProBas 7.1	£210
QuickPak Pro 4.19	£145

VISUAL BASIC 3 ADD-ONS

Comms - Async
Fax Plus for Win	£175
FaxMan SDK	£390

Comms - Network
Distinct TCP/IP Visual Internet	£265
dsSocket 1.25 Intro	£65

Database
ADE/VBX	£350
Smithware VBX for Btrieve	£180
VB/ISAM MU for Win 16-bit	£145

Graphics - Charting
Chart FX 3.0 (16-bit only)	£210
Charting Tools for Win - VB	£180
Real-Time Graphics Tools - VB	£300
VBGraphix	£270

Graphics - Image Files
Image SDK Plus/VBX 2.0	£250
ImageKnife/VBX Std 2.0	£200
ImageMan/VBX 5.0	£230

Multi-Function
Borland Visual Solutions Pack	£59
Designer Widgets 2.0	£99
Muscle (Win)	£125
VBlite 1.0	£130
VBTools 4.0	£115
Visual Developer's Suite (16 bit)	£216
WinWidgets/VBX	£160

Sundry Components
CADControl	£365
d-Barcode VBX/DLL	£94
VB/Magic Controls	£120
Visual CAD Dev Kit	£520

Sundry Controls
3D Graphics Tools 4.0	£130
EDI-VBX 1.0	£705
Gantt/VBX	£195
MediaKnife/VBX	£290
VBX Artist	£240
Visual Instrument Panel Cntrls	£150
VideoSoft VSView/VBX	£105

Text Editor Controls
TX Text-Control Standard	£205

Tools
JET Inspector 2.0	£475
SpyWorks-VB 2.1	£100
TMS Tools 1.1	£99
VBAssist 3.5	£140
VB/DLL 2.05	£165
VERSIONS/VB 1.1	£135

DELPHI
Delphi Desktop 2	£245
Delphi Developer 2	£390
Delphi Client/Server Suite 2	£1260
ABC for Delphi 1.0b	£99
Apiary Dev Suite for NetWare	£195
Asynch Pro for Delphi	£120
Borland RAD Pack for Delphi	£125
Charting Tools for Win - Delphi	£180
Code.Print Pro for Delphi	£85
Component Create	£155
Conversion Assistant Database	£98
Eschalon Power Controls	£98
Helping Hand for Delphi	£150
HyperTERP/Delphi	£120
ImageLib Pro 16-bit	£109
InfoPower	£179
InnoView MultiLanguage	£125
KingCalendar Pro	£65
List & Labels for Delphi 3.5	£295
Mobius Draw Kit	£99
Orpheus	£135
Pumpkin Project Organiser	£15
Real-Time Graphics Tools	£360
Transform: Component Expert	£125
VisualPROS 1.1	£119
WinG Sprite Kit	£99

OBERON & MODULA-2

Oberon Direct-to-COM is the easiest way to write fast, robust fully compiled 32-bit OLE/COM objects

Oberon Direct-to-COM	£1700
Oberon/F	£315
Oberon/F Dev Bundle	£569
Ofront (Oberon-to-C Translator)	£715
Oberon/F Arbitrary Precision Lib	£55
Oberon/F Source Code Analyser	£55
Oberon/F Statistical Profiler	£165
Logitech Modula-2 Dev Sys 4	£560
Stony Brook Prof Modula-2	£300

VISUAL BASIC 4
Visual Basic Enterprise 4.0	£755
Visual Basic Professional 4.0	£365
Visual Basic Standard 4.0	£78
3d Graphics Tools 5 (32bit VB4)	£149
Apiary Dev Suite for NetWare	£195
AutoCoder	£55
BetterState Pro w/VB CodeGen	£249
ButtonMaker	£75
ClassAction (VB4)	£110
ClassAssist (inc Oblets)	£175
CodeBank (VB3/4)	£88
Code Complete	£175
Designer Widgets 2.0	£99
EnQuiry 2.0 (VB4)	£199
ERwin/Desktop for VB4	£399
GeeWiz 2.0	£75
Helping Hand for VB	£150
Into Code (VB4)	£55
List & Labels for VB3/4 3.5	£295
SplashWizard	£75
Spyworks - Prof 4.0 (inc Sub)	£225
Tear Off Menus for VB4	£115
VB AppFramework	£159
VB Assist 4	£130
VB Compress Pro 4.0	£98
VB Language Manager Pro 3.0	£132
Vision StoryBoard 4.0 (VB4)	£175
Visual Bridge (Access → VB4)	£89
Visual Expert Developer	£250
XREF 2.0	£95

SoapFlakes

Paternity question

You probably think Ada Lovelace was the first programmer. Everybody does.

It is one of those little facts taught in every class called *Introduction to computing*. The Countess of Lovelace published notes about Charles Babbage's locomotive-sized Difference Engine (which many call the first computer), thereby inventing our fine art – or so the story goes. The tale has nearly elevated Lovelace, daughter of Lord Byron, to the level of programmers' patron saint.

I propose a new 'father of programming'. His name is familiar, but you have probably never heard of him. Like Lovelace, he didn't have a computer to program. The idea of physical computing probably never occurred to this man, but without his little programs all the hardware in the universe couldn't add two and two.

In the late eighth and early ninth centuries, when Charlemagne ruled in the West and the Muslim dynasty of the Abbasid Caliphs held dominion over Arabia, a man named Mukhammad ibn Musa al'Khorezmi reigned over mathematics and the science of the stars. He and many other scientists of the day were working under the patronage of the Abbasid Caliphs who had established an Arabic Academy of Science.

If Spain's claim over the new world proved to be a good return on investment for Isabella's sponsorship of Columbus, Abd Allah al'Mamun's grants to al'Khorezmi were a windfall. The scientist was the first to define algorithms, use zero ('a little circle') to hold place values, and was the chief architect of the spread of Indian numbers, the 'new' system of counting which evolved into the Arabic numeral system we use today. Meanwhile he invented algebra and analytical geometry.

If it comes up in your next game of Superquiz, you are likely to be the only player who knows that the word Algorithm comes from his name – al'Khorezmi. *Algorism*, the obscure word denoting the Arabic decimal system of numbers, sounds even closer to its eponym.

The first programs

Al'Khorezmi's work comes down to us in the form of several translations. The book which made him famous (at least in his own day) is a tidy little volume called *The Tables*. It includes a chronology with calendar algorithms and the first known tables of the sine and cotangent functions. The calendar calculation, which converts Roman dates to Arab and back, is almost identical to the coded algorithms used for the same purpose by today's computers. Al'Khorezmi created his sine table without the aid of negative values, decimal fractions, or Indian numbers.

This astounding man did not go without a modern counting system for long. His next work, *Arithmetic*, brought him fame in mediaeval courts of science and anchored his name in the word algorithm. In *Arithmetic* Al'Khorezmi introduced Indian numbers, which he had discovered on his travels, and the algorithms for addition, subtraction, multiplication, division, doubling, halving, and square root calculation. With the Caliph's support this system became the Arabic numbering system of choice, and the rest is history.

Al'Khorezmi went on to define new ways of using the numbers he had introduced. His book *Algebra* exists in a first edition, circa 820 AD, and opens with a systemisation of quadratic equations. It includes the infamous primary school equation $x^2 + 10x = 39$ as one of six examples of cases of equation to find for any single missing variable (and since negative values did not exist for Al'Khorezmi the six forms did indeed suffice). The next chapter, *Business Calculation*, aligned the former with real world problems, and again takes the modern reader back to primary school: 'Ten *caffices* cost six drachmas. How much can you buy with four drachmas?' What earthly use would we have for Microsoft Excel if we did not have a way of calculating how much money to charge people?
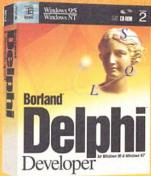
Yet despite the many fundamental principles invented, discovered, or introduced by Al'Khorezmi, the true father of programming has been overlooked and underreported. Some people have chosen to discount all his work because Al'Khorezmi didn't believe negative values existed, and therefore his quadratic equation system was seriously flawed. That logic is akin to insisting Lovelace wasn't a programmer because she didn't know C++.

Perhaps more serious is the lack of academic credit granted to Al'Khorezmi. He was renowned in scientific circles as late as the Renaissance, but he was forgotten in the 17th century, when the 'z' sound in algorithm was replaced with 'th' (to match the Greek). I learned about the man and his relationship to modern mathematics through the work of Dr Heinz Zemanek, an IBM Fellow from Vienna who spent many years researching Al'Khorezmi and in 1979 wrote a paper about the man. Alas Zemanek's research, published in *The Springer Lecture Notes*, Volume 122, 1981, is no longer in print. The copy I have was retrieved from a mass of similar documents marked for disposal by a French university's computer department.

I have only faint hopes that Al'Khorezmi, a remote and ancient Arab, will ever supplant Lovelace, English Society Lady, as the original programmer. However, when you're tipping back the champagne in celebration of Lovelace's 180th birthday this year, remember a slice of cake for Al'Khorezmi.

Adrian Leonard

Mayhem!

From screenless VCR design to arcane printer control panels, we learn what pushes **Jules May's** buttons.

He was astounded. 'What do you mean, you've never heard of it? How can you possibly have survived this long without having heard of the impact gap?'

Whatever it is, it has the kind of name that isn't easily forgotten. It conveys both power and weakness at the same time. On one level it doesn't matter what it means. The name itself is enough to persuade one to spend money. But apparently it does mean something. It represents the difference between what people do with technology and what the technology is capable of. My friend continued his tirade.

'Take video recorders. You can't program your video, can you?'

'Well, actually...'

'Nobody can program their video. Most people are able to put a tape in and press play. Power users can activate the record button as well. But only children can *program* their videos. All that power, all that functionality, all that expensive machinery, and you're using only about 10%.

'All technology is like that. When somebody buys a machine he learns the things that are immediately useful, and the rest goes to waste. That's the impact gap.'

He said all technology is like that, but I'm not convinced. Most people know how to use most of the controls on their hi-fi. Most people know how to use most of the controls on their cars. It's when you get to machines like videos and washing machines that the impact gap becomes visible.

I discovered that this problem is restricted to machines with buttons. Hi-fis and cars have controls which either latch or are continuously variable. Buttons are special, because unlike an accelerator or a tone control, a single button does many different jobs.

I looked at my video again. Even if I am unable to program it, and use it solely for playing tapes, am I truly using only 10% of its functionality? If so, which bits of the machine am I not using? What could I throw away profitably? In fact, none of it. Whether I play or record tapes, or timeshift broadcast programmes, I'm using all the motors, all the amplifiers, all the encoders and decoders. I'm still using the whole machine, even though I'm not using all the functionality.

Consider my PC. It is capable of word processing, spread sheeting, project management, chip design, managing racing cars, and a huge number of other things. I don't do all those things on it: I'm only using it for word processing right now. Does that mean I'm using only a small part of the machine? Of course not!

When an engineer designs a video recorder he's interested in the motors, and the transport, and the amplifiers. The precise manner in which programs are created and entered is relatively uninteresting to him, so those functions are crammed into a corner of a ROM somewhere just because there's a bit of space for them. The functions themselves are virtually cost free. Programmed audio recording from the radio is a function just as useful as programmed video recording, but very few radio-cassette machines have such a function because there's no obvious corner to put it. A whole new subsystem would be needed to provide such a facility, and that would increase the price of the machine.

That's where the impact gap comes from. What my friend calls the impact gap other people describe as the menace of creeping featurism. It may be possible to stuff a device full of features and functions, and many of those functions may be genuinely useful. However, when both buttons and displays cost extra money, designers use as few of them as possible, so the functions become difficult to access. That's why the impact gap is rarely seen in application programs. An extra button or a more graphical display on a computer screen costs nothing, so all the functions can be separated and made clear.

What occurs to me is this: if it's expensive to put buttons into videos and microwave ovens and washing machines, and if they don't work very well when they get there, why put them there at all? Why

not provide a separate network interface onto each of these machines and control them from a PC? If you want to play a video, put the video in the slot and it will start playing. If you want to record a programme, tell the PC to start the recorder at the right time and on the right channel. Since it is possible to provide programme information to the PC, why not just tell the PC which programme you want to record, and let the PC figure out the rest?

Of course not everybody has a PC and a network in their house, though that is changing. But a video recorder can be absolutely certain that it has access to a screen, and a great deal of information can be placed on it, just as on a PC screen. Few video recorders take advantage of the screen, so the machines remain opaque to most users. But what about the times when you know a PC exists? When I look around at the machinery on my desk, every piece is covered with buttons. Consider the printer. It suffers from exactly the same control problems as my video – not enough buttons, not enough displays, and too many functions. A few days ago, my cat sat on the buttons, and quite by accident, he discovered a genuinely useful function which I had no idea existed. Whenever other people use my monitor they always mess with the settings. Wouldn't it be nice if I could control those settings from my login profile, so my monitor always returned to how it should be when I return to the keyboard?

Right here, on my desk, I find the impact gap alive and yawning. I'm supposed to know about machinery, so if I'm suffering from it, non-technical people must be suffering far more. It occurs to me that the people who designed my printer and my monitor also suffer from it, because it didn't occur to them to put their functions on my computer screen, which

they know exists.

Creeping featurism is not a bad thing, but it does contribute to the impact gap. It is buttons that make both features and primary functions difficult to reach. They do more to open up the impact gap in the minds of their luckless users than featurism itself. If we had more switches and dials, even if they were constrained to a PC screen, we'd be more able to use the machinery we've got. ∎

Jules has got sick of waiting for BT to put voice menus onto its redirection, call waiting, and other trick facilities (who ever thought *33#73# meant something?), so he's working on a PC-based program to control his telephone using his modem. If he doesn't break the telephone in the process, you can call him on 01707 662698, and if he doesn't break his modem, you can email him as jules@cix.compulink.co.uk.

the impact gap

Say goodbye to EXplodE, and hello to EXE OnLine

*E*XplodE, the World Wide Web service launched by *EXE* last year, will never see its second birthday. This month a new service called *EXE OnLine* appeared in its place. The new site features a radically different interface, significantly more integration with the magazine, and substantially revised content. Additions include a news service with timely updates, a Usenet-style discussion forum, the most extensive set of developer-related links to be found on the Net, and – of course – revamped design. *EXE OnLine* (or *EOL*, as *EXE* wits have already begun calling it) preserves the best features of *EXplodE*, including the unique Software Training Guide, an overflowing code bin, and an appointments page just for developers.

Like many new sites on the Web, *EXE OnLine* insists on registration before new visitors can reach the majority of the new services. Registration is free and open to anyone willing to provide a valid email address. This helps us to find out which pages you like.

The response from visitors who've been to the site during its schizophrenic period of transition – half *EXE OnLine*, half *EXplodE* – has been encouraging, and your input on the new *EXE Online* is welcome. Tell us what you want to see, and we will do our best to provide it.

▶ *Point your browser at http://www.exe.co.uk.*

JavaOne: now it's JavaOS

The official unveiling of JavaOS was among many announcements at the recent JavaOne conference. The ultra-compact operating system is written 'almost entirely' in Java and will run on devices ranging from set-top boxes to network computers, PDAs, and smart phones. Unlike conventional operating systems JavaOS doesn't provide much in the way of services. Only functionality essential to run the Java Virtual Machine has been incorporated. Therefore it is possible to implement an entire JavaOS network computer system, including space to run applications such as the HotJava browser, in 3 MB of ROM and only 3 MB of RAM. Smaller embedded devices would require only 512 KB ROM and 256 KB RAM.

Companies including Borland, Symantec, and Metrowerks have pledged support for JavaOS. Those three plus SunSoft intend to enable their existing Java development environments to build for it. As a result tools that were once considered PC development packages could become major players in the highly-competitive embedded systems development market.

At the recent STEP conference in the UK SunSoft representatives were keen to point out that the scaleable nature of Java, and by inheritance JavaOS, means developers need only incorporate the functionality they require. Because Java software should run equally well across all platforms, the phonebook software which lives in your Java telephone could be co-opted to run on your Java PC, putting an end to the continual need for conversion filters and integration tools. The Sun philosophy is 'Java everywhere'. No one was talking availability or price at JavaOne, but expect it to be relatively soon and relatively cheap to license.

▶ *SunSoft can be reached on 01494 472900*

SCO cracks Unix DCOM with Server OLE

Unix servers will soon be able to provide Remote Automation services to Windows clients, thanks to Server OLE technology from SCO. Server OLE, which is fully compliant with the Remote OLE standard, will allow standard Windows applications to interface with applications and services running on Unix machines – provided the Windows application is OLE 2.0 compliant and the Unix service has been developed to follow the standard OLE APIs.

Most interesting is that Microsoft has yet to ship the official Windows Network OLE product. It was widely expected to ship with Windows NT 4.0, but as that product nears final release it seems likely that Network OLE will slip behind. Although remote automation is possible through Visual Basic 4.0 it seems Unix boxes will be the first to have this capability incorporated into their operating systems.

Cobol vendors, already heartened by the increasing demand for their products (see Brush up your Cobol on page 12) have leapt to embrace this initiative. MicroFocus Object Cobol will support retro-fitting existing applications with Server OLE. An Evaluation Pack can be downloaded from SCO's Web site, but no pricing details have emerged.

▶ *Point your browser at http://www.vision.sco.com* ▶ *call 01923 816344*

Embedded systems developers who need to write for the Siemens **SABC16x** series of controllers can now run their software **on PCs**, thanks to Hitex's **HiSIM167**'s simulation. Tel: 01203 692066, fax: 01203 692131
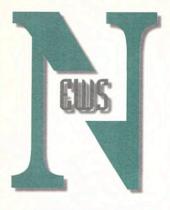
TAB/PRO 2, the latest version of the popular tab-bar widget now comes as an **ActiveX control** in addition to the usual VBX and DLL. The package includes all three at £75, and is available from **Contemporary Software**: 01344 873434, fax: 01344 872228

Testing system SQA Suite 5.0 includes features to **aid migration** of Windows 3.x applications to Win32 platforms. The same test scripts can be run on 32 and 16-bit versions of software. Call 01344 382 092 for pricing.

Digital mapping fiends can simplify their development with **MapServer 3/32**, a new map-building control from **Geosoft**. Maps built with the tool automatically feature pan, zoom, and query functions, while map features are stored in a custom object database. 0113 234 4000

OLAP and data-mining come together with **Business-Objects 4.0**, which will link into Silicon Graphics **MineSet 3D visual data-mining tool**. The aim is to allow 'non-technical users' to access corporate data. 01628 764600

Hypermedia **P-Code** interpreter from Cabot Software provides compatibility between platforms on the **Web**. 0117 958 6644

Testing times for software with QC/Advantage

CenterLine Software, longtime purveyor of automated software testing tools, believes QC/Advantage is the ultimate answer to any programmer's testing needs.

Described as a 'quality management system' by CenterLine product marketing manager Axel Kehlenbeck, QC/Advantage is designed to enhance developers' existing testing tools, not to replace them. The package is a kind of 'testing manager'. Developers define test sequences by manually executing the test once, while QC/Advantage records their actions. The sequences are stored in a custom object repository and, in true object-oriented fashion, can be inherited from, replicated, and edited.

The system is designed for use by teams of developers. Tests can be scheduled to run at predetermined times, perhaps overnight when all the workstations are free. When the developer returns in the morning the entire suite of test results is available for viewing. Kehlenbeck claims that 80% of developers do not perform sufficient testing of their code because they feel they don't have the time, and because many existing testing tools are unfriendly and need to be re-configured in order to test different functions. He says QC/Advantage completely automates the process, liberating the developer from the need to run all the tests by hand, and providing the freedom to test fully and more often. Kehlenbeck suggested this should help management attain goals – such as a guaranteed feature set and adherence to original shipping schedules – far more often.

Testing tools from Atria, Mercury Interactive, Pure, and Microsoft, and of course current CenterLine products are supported. QC/Advantage is available to run on SunOS, Solaris, HP-UX. Promised for the summer is a version for IBM AIX, and a Windows NT version should follow. In the meantime QC/Advantage on Unix is capable of running Windows testing packages on Windows machines across the network. Pricing from £1,995 per seat (for 30+ seats).

▶ *Contact CenterLine on 001 617 498 3000*

Web.sql links Sybase to the WWW

Database to Web gateways are ten-a-penny these days, but most come from third party sources rather than the major database vendors. Sybase, one of the three biggest such companies, has changed that by releasing the final code of Web.sql, its own Web gateway. The beta version has been available to download from the Internet for several months.

The software performs the usual functions required of such a gateway, including the ability to create and update HTML pages dynamically. On the server side developers have a choice of implementation using CGI or NSAPI (for Netscape servers). Web.sql can also patch into some non-Sybase databases such as DB2, Oracle, Ingres, and Informix, when used in conjunction with Sybase's Enterprise Connect. Contact Sybase direct for UK pricing information.

▶ *Sybase: 01628 597100* ▶ *http://www.sybase.com*

JavaOne: Beans means Java?

IBM's Arabica technology was previewed for the first time at the recent JavaOne conference. It is the first working implementation of the Java Beans standard. Beans is a component model for Java applets which allows re-use of Java code, and inter-operation with other object technologies such as COM and OpenDoc. In theory Java applets or applications written to the Beans standard can be used within browsers and other types of application as easily as C++, Pascal, and Eiffel components can be mixed.

Arabica is essentially a Java-based implementation of OpenDoc. With it developers will be able to write components in Java which will run on any OpenDoc-compliant platform without modification. Through the use of Java, OpenDoc will become Internet-enabled automatically. This strategy appears similar to a recently-announced Microsoft initiative which promoted the idea of writing ActiveX components (OLE Controls, in other words) in Java. Since OpenDoc will eventually be available on more platforms than COM, and barring a change in policy from Microsoft, the Beans standard should pave the way to object-oriented cross-platform applications written in Java for non-Windows platforms.
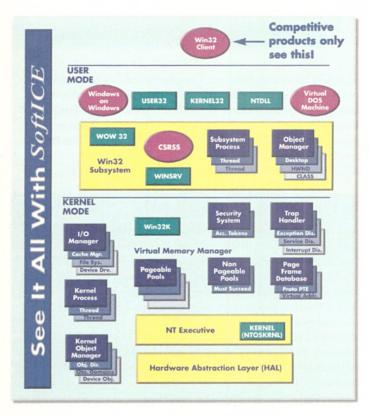
▶ *Contact IBM UK on 0171 202 3744* ▶ *Fax 0171 202 3792*

Is it a browser? Is it a class library? No, it's HotJava!

When it came to wide public attention last year, many people mistakenly confused Java the language with HotJava, the browser that JavaSoft developed to demonstrate how applets would be used. Although most casual Web surfers use Netscape with Java-enabled pages, Hot-Java has hard-core users, mostly in the Java development community. At JavaOne, to add to the confusion, JavaSoft announced that henceforth HotJava would be a technology and not a specific browser product. It is now a set of Java class libraries that provide browsing and applet-integration services to applications, and in future, to new Java-compatible operating systems and C++ development systems. One idea is to use the HotJava classes to Web-enable JavaOS devices such as phones and NCs.

Because HotJava consists of standard Java classes it can be used with any ordinary Java development environment. A new HotJava Browser built around the class libraries is available free of charge to non-commercial users. Developers interested in obtaining the source of the class libraries and of the HotJava Browser can contact SunSoft for more information.

SunSoft: 01494 472900   *URL http://java.sun.com*

KickStart and ODF in OpenDoc release 5 SDK

Although it has yet to surface for the Windows platform, OpenDoc technology has been available in beta for MacOS and OS/2 for some time. Developer release 5 brings a first look at some of the newest entries in the 'alternative' object camp. KickStart Components are a set of basic OpenDoc parts which provide primitive functionality. Button control, viewers for 2D and 3D QuickDraw images, and a QuickTime viewer component are available. From a Windows-centric viewpoint, where these kind of objects have been available for years as VBXs or OLE Controls, this may not seem a dramatic development, but Kick-Start Components have more in common with Active Controls for the Internet. Apple plans to embed many of its key technologies including video, sound, and speech recognition into KickStart components.

The OpenDoc Development Framework (ODF) is an OO framework for building OpenDoc parts and Cyberdog components (Cyberdog, for the uninitiated, is Apple's new Internet environment based on Open-Doc technology). Full sources and binaries for ODF are included in the SDK.

A minor release of OpenDoc, 1.0.4, is included within the SDK. It is available as a CD ROM or for download from Apple's OpenDoc Web site. The disk, which contains source code and example programs which are not available in the downloaded version, can be ordered direct from Apple or over the Web at the same site for a nominal duplication cost.

Call Apple on 0800 127753   *or go to URL http://opendoc.apple.com*

Yet another graphics file format

Kodak, the conventional photography people, wanted to develop a graphics file format well suited for all the stages of digital images processing, from the scan (or snapshot) through to email. The key feature required was easy image manipulation on a computer with as little memory as 8 MB (a 486/Pentium or Macintosh 68040/PowerMac). Kodak's primary target is the consumer market; a US survey shows 50% of picture taking households have PCs and 73% usually mail photos.

Kodak looked at the formats in use and couldn't find a suitable one, so it teamed up with Hewlett-Packard, Live Picture, and Microsoft to create FlashPix. The format holds images in structured storage based on OLE. The structure contains the image in several resolutions from the original size down to 64x64 pixels. At each resolution images can be compressed with JPEG. FlashPix does not support a lossless compression algorithm. Editing an image is blazingly fast, since the manipulations are done only at the required resolution and are saved as small scripts called 'viewing parameters'. FlashPix supports RGB and Photo YCC colour spaces, and colour on one screen is the same as on another, and on a printout.

Limited information is available now on Kodak's Web site, but the full specifications and the developer kit will be added in Autumn. The specifications and their use are free, but only products which pass a test suite will be allowed to use the name FlashPix. The current beta SDK for the DC20 digital camera on the Web site doesn't use FlashPix, but should in a future release.

URL: http://www.kodak.com   *Tel: 01442 61122*

Brush up your Cobol, start learning it now…

IBM has spent years trying to kill it, programmers have spent as long trying to forget it, but many hundreds of corporations still have business-critical systems that rely on Cobol. For programmers who can stomach three years of code maintenance it could be manna from heaven.

The impending year 2000 has put Cobol skills in high demand. Stuart Watkinson, General Manager of Year 2000 specialists Viasoft UK, told *EXE* he has been forced to offer Cobol pro-

(c) 1994 Right Software

grammers a 15% increase in wages and a significant loyalty package that pays out in January, 2000. Mike Kerford-Byrnes, head techy at Conversion Technologies, predicts daily pay of £500 to £700 for Cobol code specialists before the century is up.

Watkinson says 79% of code used by medium and large UK firms is date-related. From 3% to 6% will need tweaking, but all of it will have to be examined. Add problems like missing source code, and the opportunity for programmers is, well, the best of the century.

Organisations worldwide are scrambling. An unnamed US insurance company with 40,000 programs began its Year 2000 project in 1991 by hiring 35 full time programmers but is predicting it will fail to meet its self-imposed deadline of compliance by 1998. The state of Nebraska, estimating a Year 2000 cost of $31 million to update 12 million lines of code, has added two cents to the tax-hit on every pack of cigarettes sold there to help cover the cost.

Those of us willing to endure the agony of a return to Cobol could reap a great reward.

▶ *Viasoft: 01438 840833*   ▶ *Conversion Technologies: 01784 497045*

Asymetrix to SuperCede Java Virtual Machine

The next phase of Java technology has kicked off with a beta release from Asymetrix, the company founded by ex-Microsoft partner Paul Allen. SuperCede VM is an implementation of the Java Virtual Machine, the bytecode interpreter which 'runs' Java code as a virtual CPU.

SuperCede works slightly differently. SuperCede VM compiles Java bytecode to binary code as it downloads, so by the time an applet has finished downloading, a native binary file already exists on the client computer. The software then executes the binary.

As Java classes are downloaded, they are stored in DLLs which may contain code for several different applets, according to frequency of use. With these 'reference libraries' of applets saved on the local PC, SuperCede can intelligently decide to use the local copy and avoid any download overhead at all.

The SuperCede project promises to deliver a complete development environment featuring a graphical, source-level debugger, GUI builder, and class browser. It will be the first development environment to support seamless use and mixing of Java and ActiveX controls, according to Asymetrix. The initial release, SuperCede for Java, is scheduled to ship in October at a price of £395, while SuperCede Pro, which will handle ActiveX, should ship in the first quarter of 1997.

▶ *Telephone 0171 454 1061*   ▶ *Fax 0171 454 1062*   ▶ *URL http://www.asymetrix.com*

Loading down the Web with WebTest

The emergence of automated testing environments such as Centerline's QC/Advantage (see page 10) has made software testing a hot topic. According to Mercury Interactive, another leading developer of testing products, an important element is missing in traditional testing suites: the ability to test Web applications. To fill the gap the company has come up with extensions to its tools LoadRunner, XRunner, and WinRunner, under the label WebTest.

WebTest tools enhance the existing applications by adding functions to test Web response speed, data download speed, graphics integrity, and verification of HTML links. Used together with WinRunner or XRunner – testing software for Windows and X Window platforms respectively – WebTest will trawl a Web site looking for broken links or potential troublespots. With LoadRunner, WebTest runs tests to see how stable a Web server is. It can simulate multiple HTTP connections to place the maximum load on the server, giving the Webmaster an indication of how it will perform in the real-world. Like rival packages WebTest has the ability to record tests carried out manually through a browser and replay them later.

WebTest is available immediately, although some components are still in Beta and will be shipped to registered users later. For pricing information, contact Mercury Interactive.

▶ *Mercury Interactive (Europe): 0033 1 34657240*   ▶ *(USA) 001 408 523 9900*

Letters

We welcome short letters on any subject that is relevant to software development. Please write to: The Editor, EXE Magazine, St. Giles House, 50 Poland Street, London W1V 4AX or email **editorial@dotexe.demon.co.uk.** Unless your letter is marked 'not for publication', it will be considered for inclusion. Letters may be edited.

Secret admirer

Ed: This postcard refers to The future does not compute review in the March '96 issue of EXE.

A Musketeer from The Worcester Militia. Each year The Militia re-enact events from the Civil War in The Commandery.

Ed: Does this photo show how Ms Hope's correspondent looked in the sixties? Or is this how he looks now? We had been wondering what became of those famous sixties motorcycle enthusiasts...

The great debate (continued)

I think the Strong AI proponents of consciousness are missing the point, which is very simple. A simulation of consciousness on a digital computer could conceivably originate this letter, but the simulation would not be conscious.

To be conscious, in the minimal sense of experiencing raw feelings, you must first be real, that is, a physical object. And a computer simulation is not real. Real things are physical. They are made of atoms and bosons, exist in space-time, and are subject to the laws of quantum mechanics. Everything else is an *idea*. Relationships, for example, only exist in our minds. We talk of perceiving relationships but in fact we mentally construct them and disseminate them in discourse. Dis-

course is also unreal (the closest to physical it gets is as a series of relationships between perceivable differences in a medium) and a computer simulation is a mechanical form of discourse. It is a static description (the program) which could be embodied on any suitable medium, causing state changes in a discrete state machine. This resembles a book, except that the words change dynamically (suggesting the analogy of drafting a letter on a word processor).

The limited number of states in the machine is a clue that we have a problem. Consciousness cannot arise in computers as we know them. Compare this with the number of states in the brain, which is the number of particles it (or its conscious portions) contains times the number of different quantum numbers each particle can have while the brain is conscious. That's over 30 orders of magnitude greater *significant* complexity. Sure, the computer contains lots of particles, but only the number of states which make a difference to the computation (not likely to be much over a trillion including external memory) counts. But complexity is not the issue – it's the (onto)logical level of the instance of the alleged consciousness. In the case of confirmed consciousness (my own) its instance is my physical brain, but for the computer its instance is abstract – the results of computations stored in an essentially arbitrary form. The difference between the two is like the difference in logical type between something's description and its manifestation, and only the manifestation is real (James Bond never shot anyone).

OK, those are the assertions. Where's the logical rigour? The tacit assumption in the above, and in most discussions of computer consciousness, is that the world is wholly physical as opposed to being wholly or partly mental. Cognitive scientists always make this assumption since they couldn't get anywhere otherwise. Now, if the world is wholly physical then consciousness must be a physical process. Computation on a digital com-

puter is not essentially a physical process (the computer is the medium – like the ink and paper of a written message). Hence digital computation cannot produce consciousness.

If the computer model of the mind has any validity the place to look would be the analog computer. Its computations are in no way abstract. An electric circuit performing a variety of analog computations is doing its thing in the same way that a falling ball is doing its thing by falling, the sun by shining, or the brain by being conscious.

Neil Paterson
London

What?

Why must one be 'real' to be conscious? I'd agree that one has to be real to be able to exhibit one's consciousness, but the question of what reality 'is' becomes much harder to understand.

You reject the idea that a digital computer has the desired reality, even though its internal states are defined by the movements of electrons, in exactly the same way as the internal states of the analog computer are defined by the movements of electrons, or the states in your own brain are defined by the movements of more complex structures. What on earth do you think the difference is?

You say relationships exist only in our minds, and I'd guess that you'd extend that to include other hypothetical intelligent minds. Are you, then, denying physical causality as a property of the universe? What good are your quantum theories and your ideas about reality then?

Finally, an order of magnitude of 30 times is not serious. We just build 30 computers, or wait a few years for a better PC. However, ordinary computers are far more powerful than a snail's brain, and we can't model that either. Computer power is, I'm afraid, a red herring. – *Jules May*

Circles and ellipses are a new fascination for me, but back when I was a postgraduate engineering student the buzz was mechanical vibrations. I once plotted a graph showing a phase angle on one axis and amplitude of vibration on the other. On an *x, y* graph this produces a line with a peak near the resonant frequency, but on a polar graph the shape is roughly circular, as Figure 1 shows.

Determining the centre and radius of the circle revealed the resonant frequency and maximum amplitude of my vibrating system. I found the values by trial and error using tracing paper, a compass, and various bits of crockery.

Flash forward. As a born-again engineer I got hold of a survey of Woodhenge, built circa 2340 BC. As I examined the pattern of dots I wondered if they are on a circle or an ellipse, and – following others working in the same field (literally) – if they lie on arcs of different sizes joined at their ends. I worry when I have a simple-sounding question and can't find a simple answer. These two examples of the same question stayed with me until I cracked them.
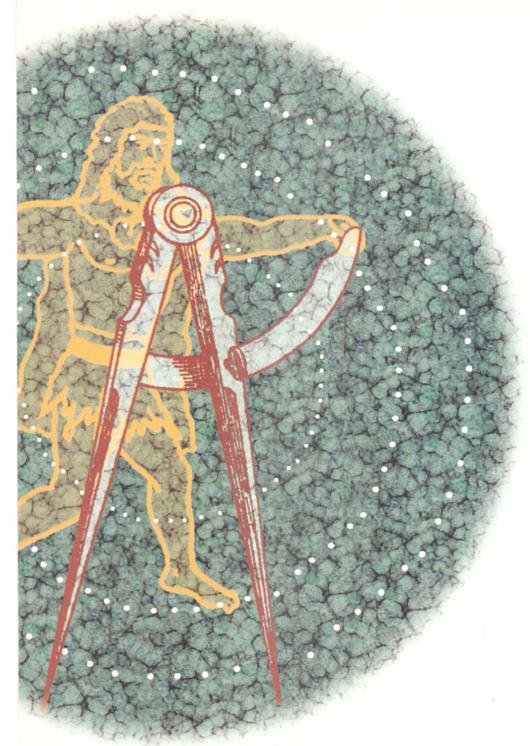
The direct problem is the easy one: given the parameters of a circle or an ellipse, where will the points on its circumference be? Any kid with a compass or a computerised turtle can solve that just by drawing them. But I needed algorithms to solve the *inverse* problem of finding the parameters of the circle or ellipse which best fits a given set of points. To learn just how useful this tool can be see the sidebar *Other inverse problems from the real world*.

The set comprises four algorithms, a pair for circles and a pair for ellipses. In each pair is one routine to make an initial

Four thousand years ago an ancient post man constructed Woodhenge, but did he plan a circle or an ellipse? Find out with **Richard Pickard's** home-made algorithms.

Circle sleuthing

estimate of the figure's parameters and another to refine the estimate. In the case of the circle the initial estimate is very good, so the refinement makes only a small improvement. For ellipses it's the other way round: the initial estimate is crude, but refinement makes a substantial difference.

A little geometry

This is plain plane geometry – ordinary straight lines and points without spheres or relativity. Any three distinct points in a plane can define either a straight line or a circle. Our focus is circles, as in Figure 2.

Draw two lines (AB and BC) to join two pairs of points. Bisect the first two lines at right angles with two more lines long enough to meet (PO and QO). Point O is the centre of the circle on which the three points lie. Doing that in Cartesian (x, y) co-ordinates is part of the algorithm's work. I've left the algebra to the sidebar *Cartesian co-ordinates* because you don't need it to get the point, and it isn't light reading.

If you have only three points in your pattern the job is over. If you have lots of points you have to take an average to get the answer. The initial-estimate algorithm takes all combinations of points three at a time, works out the x and y of the centre of a circle for the current three points, then accumulates the xs and ys. At the end it will obtain an estimate for the centre of the circle (x_c, y_c) by dividing the accumulated xs and ys by the number of points. Finally, the algorithm will get the estimated radius by averaging the distance of the points from the estimated centre.

Unfortunately the method is not totally effective. Messy-looking bits around the

This picture shows the post holes of the outermost 'circle' at Woodhenge. It is about 150 feet wide. The best-fitting true circle has a radius of 71.48 feet and a 2.71 feet measure of error. The best-fitting ellipse has semi-axes of 75.24 feet and 68.20 feet; its axis points at 39.58 degrees (roughly SW to NE). The measure of error is 1.05 feet. The line in the middle of the picture joins the foci. The four arcs, joined at their ends, fit even better and have an obvious (but nonetheless charming) symmetry. The measure of error is 0.836 ft.

Figure 1 – Phase angle vs. amplitude of vibration; x,y graphing vs. polar.

finish as an array of (x, y) co-ordinates, and the array must be accompanied by the number of points. The units used are optional, and could be microns, inches, or even Ordnance Survey grid references. The values get passed into the initial-estimate routine, and the centre and radius get passed back. The routine is written as a function; the value of the function is the measure of error in the estimate.

The measure of error is the root mean square deviation of the radii of the individual points from the estimated radius. That means taking the individual errors in the radius – one for each point – squaring and accumulating, taking the square root, then dividing by the number of points. The answer is in the same units as the original data. See Figure 5.

I do all my coding in Modula-2, but if you get hold of the code described here you shouldn't have any trouble transcribing it into Pascal, Fortran 77, or any other well-defined [sic - Ed.] language.

Refining the circle

The refinement method is quite good – particularly if you have a reasonable initial estimate – but it improves in very small steps. It adjusts the position of the centre and the radius value of the estimated circle a tiny bit at a time, reducing the measure of error. This method is very quick to describe, but you might like to take a coffee break while you absorb the idea…

edge of the circle will lead to a poor result. Look at Figure 3.

The centre will be artificially biased away from the flat spot. To get around this problem, weight the good calculations where the centre is higher than the poor ones. But what defines 'good'? What should be used as the weight? I had an 'Aha!' moment while looking at sketches of different cases. Choose the area of the triangle formed by the current three points as the weighting factor. This works extremely well. The furthest-apart and well spaced triplets of points will provide good estimates of the centre's location – and they make triangles with the largest areas. Near-together triplets can project a very distorted view of the location of the circle's centre – but they also make small triangles. See Figure 4.

The initial-estimate algorithm takes all combinations of points three at a time and works out the x and y of the centre of a circle for the current three points. It then calculates a weight, specifically the area of the triangle for the current three points. It accumulates the weighted xs and weighted ys, then accumulates the total weight. At the end, the algorithm divides the accumulated and weighted xs and ys by the product of number of points, and by the accumulated

weight. This gives an excellent estimate for (x_c, y_c). Finally, it calculates the estimated radius by averaging the distance of the points from the estimated centre.

Practically home free

At this stage two practicalities are important: setting up the data and measuring the error in the answers. The point data has to

Figure 2 – Plain plane geometry with a focus on circles.

The method is to take each point in turn and allow it to answer the question 'since the radius is currently estimated to be R, where would I, a point, like the centre of the circle to be?' The ans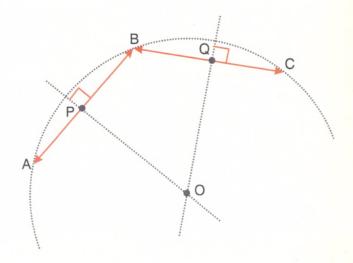wer is different for each point. Some would like the centre nearer, others further away. The algorithm averages the shifts in the x and y directions that the points would 'like' the centre to have, then calculates the centre from that. Next it can calculate a new radius by answering another question: 'since I, the centre, am currently estimated to be at (x_c, y_c), what would I like the radius to be?'

This method is an effective way of using the individual errors to shift the centre toward or away from each point. In Figure 5 the marked point at the top would 'like' the centre to be up and to the right. The marked point at bottom left would 'like' that too...

The routine which implements this algorithm has to receive the raw data and the previously estimated values for centre and radius. It returns new parameters for the circle and, as the value of the function, a new measure of error. In practice a single call to the routine is fast, but overall the convergence is slow. For circles with lots of points it may take hundreds of calls before the little improvements stop accumulating.

Between them these routines are very robust. They will handle quite small arcs or complete circles with equal ease. Test them on easy figures such as four points disposed symmetrically about (0, 0) and work up to your local ring road (or mine, the M25, when you have time).

About ellipses

In theory, you can put a pin in a board, loop a string around it, and draw a circle with a pencil. Also in theory you can put two pins in the board and draw an ellipse. The pins are at the foci. But have you ever tried? I have, and it's almost impossible. The pencil keeps slipping over the string. I raise this because the view of an ellipse being constructed in this way is what lies at the heart of the methods I am describing.

Cartesian Co-ordinates

The location of a point in the plane is shown by its distance to the right (x) and its distance up (y) from the so-called *origin*. I will represent the location of a point p by (x_p, y_p).

The equation of a line (in x, y), given two points (x_a, y_a) and (x_b, y_b) is:
$$y = (x - x_a)(y_b - y_a)/(x_b - x_a) + y_a$$

The equation of the line which bisects the line from (x_a, y_a) to (x_b, y_b) at right angles is:
$$y = (y_a - y_b)/2 + (x_b - x_a)/(y_b - y_a)[x - (x_a - x_b)/2]$$

The equations for the point where two lines meet are:
$$X = \frac{\{(y_a - y_b)[(y_c - y_b)^2 - (x_c - x_b)^2] - (y_c - y_b)[(y_b - y_a)^2 - (x_b - x_a)^2]\}}{2[(x_c - x_b)(y_b - y_a) - (x_b - x_a)(y_c - y_b)]}$$

and, dare I say it, similarly for y.

Putting these three together is what we did – so easily – in Figure 2.

The area of a triangle whose corners (posh *vertices*) are at (x_a, y_a), (x_b, y_b) and (x_c, y_c) is given by:
$$A = |(x_a y_b - y_a x_b + x_b y_c - y_b x_c + x_c y_a - y_c x_a)/2|$$

Watch out! The intermediate result could be negative because of going the wrong way around the angles.

The ellipse-solving routines work in a way similar to the circle-solvers. They make an initial estimate then refine it. This time the given data, an array of points, has a constraint: the values must be presented in sequence of angle measured from the centre of the circle. The circle must be solved before the angle of each point from the centre can be found. Only after all that can the points be sorted.

Initial ellipse

The routine accepts the array of points in angle sequences and looks at portions of the figure. It is looking for the arc (from the neighbouring points) that has the smallest radius, to find one of the sharp ends of the ellipse.

To solve for an initially well-fitting circular arc the routine starts at the first point and takes the neighbouring points clockwise (about 1/3 of the points in the whole figure). It then moves to the second point and takes the same number of neighbouring points clockwise. At each step the arc with the smallest radius is remembered.

The routine finishes by finding both foci. The first focus is taken to be the centre of the circle from which the arc with the smallest radius came. The second is found by symmetry, across the centre of the previously estimated circle. This process is illustrated in Figure 6.
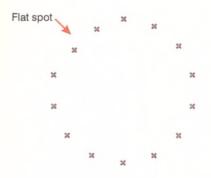
Figure 3 – Messy-looking bits around the edge of the circle will lead to a poor result.

A poor estimate but the small triangle E_1 means weighting is low

A good estimate and the large triangle E_2 means weighting is high
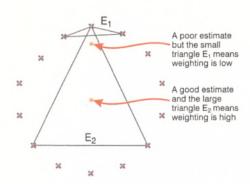
Figure 4 – The furthest-apart and best spaced triples of points will provide good estimates of the centre's location.

e_2 - an individual error

Radius

e_{10} - another error
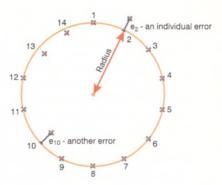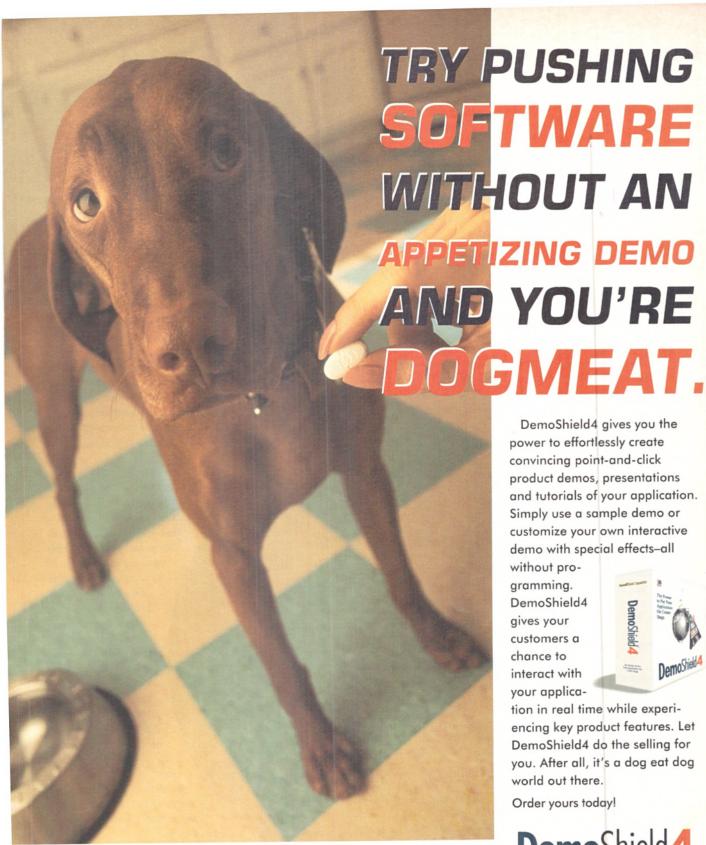
Figure 5 – Individual errors shift the centre toward or away from each point.

TRY PUSHING SOFTWARE WITHOUT AN APPETIZING DEMO AND YOU'RE DOGMEAT.

DemoShield4 gives you the power to effortlessly create convincing point-and-click product demos, presentations and tutorials of your application. Simply use a sample demo or customize your own interactive demo with special effects–all without programming. DemoShield4 gives your customers a chance to interact with your application in real time while experiencing key product features. Let DemoShield4 do the selling for you. After all, it's a dog eat dog world out there.

Order yours today!

DemoShield4
Show and Sell

CIRCLE NO. 700
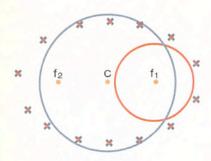
c = centre of circle which roughly fits the points

f_1 = centre of circle at sharp end of ellipse

f_2 = symmetrically placed "other" focus

Figure 6 – The algorithmic routine finishes by finding both foci.

This part of the algorithm has a tendency to put the foci too close together. It can be remedied by taking 1/4 of the points and getting a better fit to the sharp end of the ellipse, but at the risk of getting the wrong answer completely, particularly if the pattern of points has bad irregularities or gaps. Regardless, the refinement soon sorts out the initial estimate.

In contrast to the circle solver the ellipse solver does need points all round the figure. The circle solver can deal with small-angle arcs, and very well, but the ellipse solver really needs points around 2/3 or 3/4 of the circumference.

The results of the call to the routine are the locations of the foci and the length of the ellipse's major axis (overall length, a).

Refining the ellipse

This method is based on that for circles. Remember the image of points, two pins, and a loop of string.

The algorithm takes each point in turn and allows it to answer the question 'since the length of the string is currently estimated to be S, where would I, a point, like the foci of the ellipse to be?' The answer, as with our inquisitive points on the circle, is different for each. The algorithm averages the shifts in the x and y directions that the points would 'like' the foci to have, then calculates a new string length by answering the question 'since we, the foci, are currently estimated to be at (x_{f1}, y_{f1}) and (x_{f2}, y_{f2}), what would we like the string length to be?'

Shifting the foci is itself a two-stage process. Each point actually answers the double question 'since the length of the string is currently estimated to be S and the

first focus is at (x_{f1}, y_{f1}), where would I, a point, like the second focus to be?' and 'since the length of the string is currently estimated to be S and the second focus is at (x_{f2}, y_{f2}), where would I, a point, like the first focus to be?'.

As for the circle, so for the ellipse. Each call to the routine produces a small shift in

Other inverse problems from the real world

If we had a sharply focused photo and a dirty lens, we could predict what the picture would look like through the lens. But suppose we only had a fuzzy photo and we wanted to see the image clearly. We would have to solve the inverse problem of discovering what lens would produce that picture *and* what the picture is. Similar questions, tackled in all the 'ologies', are examples of *inverse* problems.

Seismology tries to 'see' and build a model of the interior of the Earth from readings taken at the surface. The model has to explain the readings. Deviations by the actual from the theoretical change the model.

The midwife visualises the growing embryo using a device based on the reflection and scattering of sound waves. In this inverse problem the question is about the location of reflective surfaces inside the human body based on some prior knowledge and on soundings taken at the surface.

In x-ray crystallography the measurement process starts with photographing x-ray diffraction patterns from clean and from doped crystals. The positions and intensities of the points are then measured. Finally, the hard bit: decoding the raw data to reveal the locations of atoms within molecules.

In all of these cases the influence of small errors of measurement can get in the way. It takes a careful mixture of math and intuition to solve these problems.

the locations of the foci and a new value of the string length (and hence the major axis). This algorithm is a bit unstable. For a badly formed set of points, the answer may be wildly wrong, and look like a pair of vapour trails plotted across the Pleiades. If the points form a reasonable-looking ellipse in the first place, these two routines will certainly produce a good answer quite rapidly.

D-I-Y shape solving

My 'Aha!' – using weights in first estimating the circle – is just one possible method. You may devise some other schemes, even ones which progressively alter weights associated with particular points. For example, one refinement scheme progressively ignores points which are 'obviously wrong'. Another allows the circle to go 'near' the majority of points rather than trying to put it through them. I have tried varieties similar to both, but they didn't work for me.

I have collected the two pairs of routines – with test program, data, and results – into a small package which you can get by following the instructions which appear at the end of this article.

I have not offered the I/O routines or graphical display routines I used. I wrote my own little library of procedures to draw true circles, ellipses at any angle, lines, and blobs – all with clipping at the edge of the frame. I did this partly to help me develop the shape-solver routines and visualise the results, and partly to investigate the circular-looking shapes formed by the patterns of post holes at prehistoric Woodhenge. Both endeavours have succeeded. I am now convinced the builders of Woodhenge had no idea what an ellipse is, but they certainly knew their arcs and circles. ∎

Richard H. Pickard – a former student of mechanical engineering – continues to take an interest in the methods of computing and software engineering and the application of those methods to solving problems in business and technology. He can be contacted by phone at 0181 993 1245, but is not netted.

The routines referred to above are available on EXE's FTP site: ftp://ftp.exe.co.uk/pub/exestuff/circles/ and on Cix, in the EXE conference.

Spritely optimisation

The on-screen sprites of computer games must be extremely agile. **Gavin Smyth** has figured out how to make the little devils go lightening fast. In this, the first article of a two-part series, he explains how.

Since *Space Invaders* displaced *Pong* as the archetypal computer game, drawing sprites on the screen has been fundamental to most programs of the genre. This reality has turned many programmers to the challenge of making a drawing mechanism for the nimble creatures of lore that is flexible and as fast as possible. Optimising code is the key.

To draw a sprite obviously requires the sprite data. An array of pixel values with associated width and height will suffice, and herein each image is encoded as a C++ class with a method to draw the sprite on the screen. Next a place for the sprite is required. With a PC VGA display adapter operating in mode 13h (320 ˜ 200 pixels, with one byte per pixel) it is the screen memory block starting at location A0000h. This resolution is one of the easiest to work with on the PC. The pixel at (0, 0) – top left – is the byte at address A0000h. Pixel (0, 1) is at A0001h; (1, 0) is at A0140h (A0000h + 320). Colour palette information should be included in the sprite definition; for this article it is the default palette for mode 13h.

Figure 1 contains type definitions used for brevity. Sprites are managed as objects of class `Image`, declared in Listing 1, which contains only the constructor and drawing functions. Class definitions should really include a copy constructor, assignment operator, and destructor, but in this case the defaults supplied by the language are sufficient because the class is so simple. Note the double `const` in the `_pixels` declaration states that both the pointer and the data it points to are treated as constant.

Sprite-building

The constructor (Listing 2a) initialises – rather than assigns – the internal variables with the values passed in. With this mecha-nism, for added safety, the variables can be marked as constant within the class declaration. If the constructor was coded as shown in Listing 2b, the `const` keyword in the member declarations would have to be omitted to prevent the compiler from spotting subsequent accidental assignments to these variables as errors. Correctness is not guaranteed by `const`, but it helps trap many careless mistakes and should be used wherever possible to reduce the likelihood of accidentally altering something that ought to be invariant.

The pixel data vector passed into the constructor is not copied. Instead the `Image` class simply references it. The data is intended to originate from a C++ vector defined within the program, so there is no need to copy it. Alternatives include reading it from a file, but for the current implementation this option is marginally less efficient, since the time and space required would outweigh the advantage.

The vector defines the pixel values of the sprite starting at top left, moving along and down the sprite. The data could not be defined as a two dimensional array in C or C++ because different sprites are different sizes, and C++ does not store information about dimensions in the array. An array type for the largest sprite could be defined, but would be a waste of storage for smaller ones; a clever C++ matrix class could be employed, but would add unnecessary inefficiency to what must be a very quick operation. Instead the sprite pixels are arranged as a simple one dimensional array with the pixel for row x, column y being the value at `_pixels[ x * _width + y ]`. The multiplication can very easily be optimised away.

The sprite used later in the speed comparisons appears on page 32. Part of its data

typedef unsigned short Coord	Position on screen or screen dimension.
typedef unsigned int Count	Any counter (as long as it never goes negative).
typedef unsigned char Pixel	Value put into the screen memory (strictly, the index into the colour palette).

Figure 1 – Type definitions used for brevity.

	Slow C	Fast C	Slow assembler	Fast assembler
25MHz 386 SX, compiler optimisation enabled	120	180	720	890
33MHz 486 DX, no compiler optimisation	380	1100	4400	5200
33MHz 486 DX, compiler optimisation enabled	600	1200	4400	5200
90MHz Pentium, compiler optimisation enabled	2000	3400	6200	8700

Figure 2 – The results in sprites per second.

FEATURES

```
class Image
{
public:

    Image( const Pixel* data,
           Coord width, Coord height );

    void plot( Coord x, Coord y );

private:

    const Pixel* const _pixels;
    const Coord  _width, _height;
```

Listing 1 – Sprites are managed as objects of class Image.

definition is shown in Listing 3a. Constructing an alien sprite like this is as simple as the single line in Listing 3b.

Draw!

The `plot()` function puts the sprite on the screen, and optimising the code helps. The following function definitions have long names to indicate the algorithm employed, but are really just variants of the `plot()` member function declared in Listing 1.

My first attempt to put the sprite on the screen looked like Listing 4a, with the `Screen_XXX` values defined in Listing 5. The first two values are the width and height of the display. The last defines a pointer. Because of the standard PC 16-bit operation, location A0000h cannot be referenced directly. It must be accessed via segment and offset. Fortunately the entire mode 13h screen memory fits within one segment, so once the segment base is set, it need not be changed. The macro `MK_FP(0xA000, 0)` creates a far pointer from the specified segment and offset. This code is built with the large memory model, which means all the pointers are far pointers. The code could be made more efficient by using near pointers, but the optimisations below remove most of the effects of this inefficiency anyway.

My first effort is hideously slow, and I'm almost ashamed to make it public. Nevertheless it works, and it illustrates the logic of the operation, but it needs to be faster.

The program always steps through the sprite data in sequence. When it has completed one row it immediately starts on the next. For example, the last pixel on the third line is `_pixels[ 2 * _width + (_width − 1) ]` and the first on the fourth line is the very

next location in the vector, `_pixels[ 3 * _width + 0 ]`. The multiplication and addition on the right side of the assignment can be eliminated. A pointer to the start of the sprite data can be created and stepped along for each pixel, regardless of its place in the line.

The same can be done for the left side – almost. A pointer can be incremented in the same way, but at the end of each row it must skip several pixels to get to the equivalent position on the screen, `Screen_width − _width`. Combining these two gives the function in Listing 4b, without multiplication in the loop. Although it introduces a bundle of local variables, this is much faster.

A few more tweaks will make the C a little faster. Counting down and comparing

with zero is marginally faster than counting up. Placing heavily used variables in registers can speed things up, but since the processor has a limited number of registers the `register` keyword is really only a hint to the compiler that the associated variable is heavily used. Some compilers ignore the keyword, particularly when compiler optimisation is disabled. Another minor optimisation concerns the `for` loops, which checks the test condition *before* processing the loop body (so that `for(i = 0; i < 0; i++)` correctly executes zero times). Our sprites have non-zero height and width, so the test can be moved to the end of the loop with a

`do...while` structure, and performed once less. Listing 4c shows the best general sprite copying C++ routine I could come up with, and is roughly twice as fast as the first example. The only way to improve this in C++ is to turn on all compiler optimisations and fiddle with near pointers. However, by delving into assembler, it can be made faster again, by a factor of almost four.

Only if I must...

Assembly language programming is not easy or fun. I try to steer clear of it, but in sprite drawing, where speed is very important, it would be a mistake to ignore the detailed low level control it offers. For the lazy method, have the compiler produce the assembly output file (`-S` option on most compilers), then examine it to get the general structure of the code. This, at least, shows where the variables are on the stack. Much time will be spent translating the compiler's rather dumb register and variable assignments into more efficient ones. Even if you don't intend to write assembly code it is worthwhile to peruse the compiler-generated assembler occasionally. It can suggest improvements to the C code, mainly to compensate for the compiler's lack of intelligence.

Listing 6 shows what the Borland compiler, with all speed optimisations enabled, did with the loops of the last C++ function above. The source code appears as comments in the assembler, and the odd-looking label names are the ones generated by the compiler. The numbers at the left of the listing refer to these notes:

1. The compiler is using ES:BX as the C++ `this` pointer. The assembler produced for a number of C++ member functions uses BX for this purpose, unless the `fastthis` optimisation – available in recent versions of the compiler – is used, but then the resulting code is implementation dependent. Other compilers may use different register conventions. According to the declaration of the `Image` class (Listing 1) the first member variable is the `_pixels` pointer, obvious given the long word at address ES:[BX]. The compiler has ignored the register hint: it immediately copies the pointer into memory on the stack. The long word at address BP−4 is pointer `p`.

```
Image::Image( const Pixel* data, Coord w, Coord h ) :
        _pixels( data ),
        _width( w ),
        _height( h )
{}
```

Listing 2a – The constructor initialises the internal variables.

```
Image::Image( const Pixel* data, Coord w, Coord h )
{
    _pixels = data;
    _width  = w;
    _height = h;
}
```

Listing 2b – Constructor definition using member assignment.

2. Once again, `register` has been ignored, and BP–8 is used as the pointer `s`.

3. `offset` is stored in BP–10.

4. `ES:[BX+6]` contains the `_height` field in the Image object, indicating the CX register is the counter for the outer loop.

5. The DX register is the counter for the inner loop.

6. This rather convoluted bit of code loads BX with the source pointer (from the memory located at [BP–4]), loads AL with the byte addressed by this pointer, reloads BX with the destination pointer (from [BP–8]), shoves AL into the destination location, then increments both of the pointers in memory.

7,8. Since the decrement instruction sets the zero flag, these statements, which move data into AX and 'or', the AX register with itself, are totally unnecessary.

Some of the faults are obvious. The working pointer variables don't need to be kept in memory. The end of loop tests can be simplified by removing the redundant operations, and the 8086 MOVS instructions can be used in the innermost loop.

Listing 7a is a faster and considerably smaller variant. I'm too lazy to write a complete assembler routine and do only the critical bits. C++ handles the function stack

```
const int alienWidth  = 27;
const int alienHeight = 31; static const Pixel alienData[] = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, ...
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, ...
  0x00, 0x00, 0x00, 0x00, 0x20, 0x00, ... ...  0x00, 0x28, 0x28, 0x28, 0x28, 0x28, ...
  0x28, 0x28, 0x28, 0x28, 0x28, 0x28, ...
  0x28, 0x28, 0x28, 0x28, 0x28, 0x0F, ...
  0x28, 0x28, 0x28, 0x28, 0x28, 0x0F, ... ...
};
```

Listing 3a – The classic alien sprite...

```
const Image alien( alienData, alienWidth, alienHeight );
```

Listing 3b – ...is constructed with a simple single line.

```
void Image::plotSlowC( Coord x, Coord y )
{
  for( Coord row = 0; row < _height; row++ )
    for( Coord col = 0; col < _width; col++ )
      Screen_base[ ( y + row ) * Screen_width + x + col ] =
                  _pixels[ row * _width + col ];
```

Listing 4a – A first attempt to put a sprite on the screen.

```
void Image::plotFasterC( Coord x, Coord y )
{
  const Pixel* p = _pixels;
  const Pixel* s = Screen_base + y * Screen_width + x;
  const Coord offset = Screen_width - _width;

  for( Coord row = 0; row < _height; row++ )
  {
    for( Coord col = 0; col < _width; col++ )
      *s++ = *p++;
    s += offset;  // Back to the right place for the next row
  }
}
```

Listing 4b – A pointer function without multiplication in the loop.

```
void Image::plotFastC( Coord x, Coord y )
{
  const register Pixel* p = _pixels;
  register Pixel* s = Screen_base + y * Screen_width + x;
  const register Coord offset = Screen_width - _width;

  register Coord row = _height;
  do
  {
    register Coord col = _width;
    do
      *s++ = *p++;
    while( --col > 0 );
    s += offset;  // Back to the right place for the next row
  } while( --row > 0 );
}
```

Listing 4c – My best sprite copying C++ routine.

```
const Coord Screen_width = 320;
const Coord Screen_height = 200;
Pixel* const Screen_base = (Pixel*)MK_FP( 0xA000, 0 );
```

Listing 5 – Screen_XXX values.

frame and allocation of variables on the stack. The code would be faster if it was entirely in assembler, but the writing would not be as enjoyable! A good policy is to get it working, and then get it working fast by concentrating on maximum gain for minimum effort. A development time versus efficiency trade-off comes into play: I prefer to expend effort on the areas that matter most, like speeding up the tight loops. I will worry about the rest later (or never, if I'm lucky). This particular brand of coding trickery is very sensitive to the compiler chosen and compiler options selected. Changing the memory model will almost certainly break the code. I used a Borland compiler, but similar techniques will work with any other 16-bit compiler. The syntax for embedded assembler should be similar.

Sometimes the inner loop should not be coded as shown in Listing 8a. Programmers may have to go to the effort of writing it as in Listing 8b, because when programming for any machine lower than a 386 an interesting 'feature' of the processor must be considered: if a repeated string instruction is interrupted it may not complete properly. REP is not an instruction in its own right; it is a prefix byte for string oper-

ations. If a string operation on a 386 or above is interrupted before the count hits zero, the address pushed on to the stack when the interrupt is processed is that of the complete instruction including prefixes.

However, prefix bytes were not taken into account by the older processors, so upon return from interrupt the REP is ignored. The processor executes a single string operation then continues. In the longer form the CX register is checked at the end of the REP loop. If it is not zero the loop was interrupted and must be re-entered. The check can be dispensed with when coding only for 386 and above.

```
1 ;   const register Pixel* p = _pixels;
      les   bx,dword ptr [bp+6]
      mov   dx,word ptr es:[bx+2]
      mov   ax,word ptr es:[bx]
      mov   word ptr [bp-2],dx
      mov   word ptr [bp-4],ax

2 ;   register Pixel* s = Screen_base + y * Screen_width + x;
      mov   ax,word ptr [bp+12]
      imul  ax,ax,320
      mov   dx,word ptr DGROUP:Screen_base+2
      mov   bx,word ptr DGROUP:Screen_base
      add   bx,ax
      add   bx,word ptr [bp+10]
      mov   word ptr [bp-6],dx
      mov   word ptr [bp-8],bx

3 ;   const register Coord offset = Screen_width - _width;
      mov   bx,word ptr [bp+6]
      mov   ax,320
      sub   ax,word ptr es:[bx+4]
      mov   word ptr [bp-10],ax

4 ;   register Coord row = _height;
      mov   cx,word ptr es:[bx+6]

 @4@226:
 ;   do
 ;   {
5 ;     register Coord col = _width;


      les   bx,dword ptr [bp+6]
      mov   dx,word ptr es:[bx+4]

   @4@254:
 ;     do
6 ;       *s++ = *p++;
      les   bx,dword ptr [bp-4]
      mov   al,byte ptr es:[bx]
      les   bx,dword ptr [bp-8]
      mov   byte ptr es:[bx],al
      inc   word ptr [bp-4]
      inc   word ptr [bp-8]

7 ;     while( --col > 0 );
      dec   dx
      mov   ax,dx
      or    ax,ax
      ja    short @4@254

   ;     s += offset;
      mov   ax,word ptr [bp-10]
      add   word ptr [bp-8],ax

8 ; } while( --row > 0 );
      dec   cx
      mov   ax,cx
      or    ax,ax
      ja    short @4@226

   ; }
```

Listing 6 – *The Borland compiler's work, with all speed optimisations enabled, on the C++ function.*

The alien sprite

```
void Image::plotSlowAsm( Coord x, Coord y ) const
{
  assert( x + _width <= Screen_width );
  assert( y + _height <= Screen_height );

  Pixel* s = Screen_base + y * Screen_width + x;
  const Pixel* p = _pixels;

  _asm {
    push ds;                    // Preserve!

    les   bx,this;              // Use ES:BX as "this" pointer
    mov   ax,es:[bx]._width;    // AX contains "width"
    mov   dx,es:[bx]._height;   // DX contains "height"
    mov   bx,Screen_width;
    sub   bx,ax;                // BX contains "offset"

    les   di,s;                 // ES:DI is "s"
    lds   si,p;                 // DS:SI id "p"
    cld;

  }

rowloop:              // Have to come out to C(++) to define label
  _asm mov   cx,ax;             // Get the row width
movint:
  _asm {
      rep   movsb;             // Copy a single row
      jcxz  short movdone;     // Done, or interrupted?
      loop  movint;
  }
movdone:
  _asm {
    add   di,bx;               // Move to the next row
    dec   dx;
    jnz   rowloop;             // Finished all the rows?

    pop   ds;
  }
}
```

Listing 7a – *A modified, faster, and considerably smaller variant, with slow assembler.*

```
void Image::plotFastAsm( Coord x, Coord y ) const
{
  Pixel* s = Screen_base + y * Screen_width + x;
  const Pixel* p = _pixels;

  _asm {
    push ds;                    // Preserve!

    les   bx,this;              // Use ES:BX as "this" pointer
    mov   ax,es:[bx]._width;    // AX contains "width"
    mov   dx,es:[bx]._height;   // DX contains "height"
    mov   bx,Screen_width;
    sub   bx,ax;                // BX contains "offset"

    les   di,s;                 // ES:DI is "s"
    lds   si,p;                 // DS:SI id "p"

    cld;
    shr   ax,1;                 // CX is width / 2
    jc    oddrowloop;           // Was width odd?
  }


rowloop:              // Have to come out to C(++) to define label
  _asm {
    mov   cx,ax;                 // Get the row width
    rep   movsw;                 // Copy a single row
    add   di,bx;                 // Move to the next row
    dec   dx;
    jnz   rowloop;               // Finished all the rows?
    jmp   endroutine;
  }
oddrowloop:           // Have to come out to C(++) to define label
  _asm {
    mov   cx,ax;                 // Get the row width
    rep   movsw;                 // Copy a single row
    movsb;                       // ... and the last pixel
    add   di,bx;                 // Move to the next row
    dec   dx;
    jnz   oddrowloop;            // Finished all the rows?
  }
endroutine:
  _asm pop   ds;
}
```

Listing 7b – *The MOVSW instruction moves two bytes at a time.*

develop to advantage

Build better applications
Add performance, functionality & versatility

BoundsChecker Professional
BoundsChecker Professional redefines automatic error detection for C/C++ developers using Windows 95 or NT. Professional Edition has breakthrough technologies to capture even more information, with extended API compliance checking. Integration into the VC++ environment, enables BoundsChecker to be used at all stages of development. *New version 4.0* has Delphi 2 support and many new features. **from £235**–PS Soft-ICE/NT now shipping. **£495**

Sybase SQL Anywhere
The powerful new version of **Watcom SQL,** delivers a high-performance DBMS for mobile, desktop and workgroup environments. Ideal for broad deployment, SQL Anywhere requires minimal DBA support and offers interoperability with Sybase System 11™. Replication technology facilitates database access for occasionally connected users. Features small footprint and easy-to-use GUI tools. New flexible license options. **from £195**

Borland C++ 5.0 Dev. Suite
The Ultimate in Programming Power. New Borland C++ Development Suite for Win95 /NT saves developer time with five essential tools useful from coding to installation. Edit/compile/debug 32- and 16-bit code, automatically detect bugs, use version control, compile and debug Java Apps and use App Accelerator to speed Java Apps. **£315**

MKS Toolkit ver 5.1
MKS Toolkit gives Windows NT3.5+ and Windows 95 developers a full suite of powerful UNIX tools including KornShell, awk, awkc, vi and visual diff for Windows, make, a windows scheduler, grep, sed, tar, cpio, and pax - more than 190 utilities and cammands for performing a variety of computing tasks, with support for NT & 95 long filename. For Win 95 & NT–Intel, Alpha, Mips on one CD. **£239**

Amazon
Amazon is the fast way to build and maintain live, interactive business applications by integrating corporate databases into the World Wide Web, enabling fast, friendly customer interaction. Amazon enables a new generation of low-cost, high-impact customer-focused applications for Inter/Intranet delivery. With Amazon go beyond static "dead web" pages, to create "live" services with text and graphics, data and logic, by enabling access to multiple information sources and the inclusion of business rules and data processing.

Optima++
Optima++ Developer is the first RAD tool to combine the power of an industry standard O-O language, C++, with the productivity of a component-centric client/server and Internet development environment. Includes a built-in copy of 32-bit Sybase SQL Anywhere RDBMS. OCXs are automatically integrated, and accessed with drag-and-drop programming, dynamically generated wizards, and online reference information. **£139**

CodeWright Professional 4.0
The programmer's favourite programmer's editor, now has a user interface polished for Windows 95. Favourite features, such as multi-file, multi-window editing, including search & replace, chroma-coding and more.. are enhanced with the new API Assistant and Button Links to tag notes etc. Available for Windows 3.1 and Windows 95 & NT in a single pack. *Now available on CD or 3.5" – £179*

Btrieve 6.15
Btrieve 6.15 for Netware, Windows, 3.1, NT and 95 is the de-facto standard database used by developers working with 3rd, 4th or 5th generation languages to build mission critical client/server applications. Features include multi-tasking, registry support, dynamic file expansion, true WinDLL requesters, dual mode requesters (NT and Netware), improved memory use, SFT III support, and more... Btrieve Developers Kits for Win, Win 95/NT & DOS are available. **from £345**

TO ORDER CALL
0171-833-1022
FAX 0171-837-6411

SYSTEM SCIENCE

System Science, 1 Bradley's Close, White Lion St. London N1 9PN

your first choice for better development tools

```
        mov  cx,limit
        rep  movsb
```

Listing 8a – The naive rep loop.

```
        mov  cx,limit
restart: rep  movsb
        jcxz short done
        loop restart
done:
```

Listing 8b – The same code for machines older than the 80386 microprocessor.

More can be done, but not much. The code uses the MOVSB instruction to copy a byte at a time, but all of the 80x86 family members have the MOVSW instruction, which moves two bytes at a time. Listing 8b does so, and checks for an odd number of bytes (the REP loop restart was omitted for brevity). When writing only for 80386 and above MOVSD will move four bytes at a time. The only ways to speed up this particular bit of code are to replace the C++ entry and exit code with more assembler and juggle the instruction ordering to assist the processor pipelines in the higher end processors, but that may be more effort than it's worth.

Pulling it together

Listing 9 is a short program that blasts lots of Images on to the screen in random positions and times the operations. Drawing a single sprite is much too quick to measure accurately, so a large number – numIterations – are drawn for a more accurate result. The assembly code versions are much faster than the C ones, so a further multiplication factor was added to the number drawn. The PC clock operates at about 18.2 ticks per second, so the timings will not be precise. They also include the overhead of the loops and random number generation, but the results will give a good idea of the relative performance of the different implementations of the drawing routine.

The vidInterrupt() routine is a short and not particularly efficient way to change video modes and perform other screen based operations via the BIOS. It is only used three times in the program (to get the current mode, to set mode 13h, and finally to restore the original), so its speed does not have great impact overall.

I ran the code on a few machines. The results (to two significant figures) are given in Figure 2. However, it is not only the processor that affects the final numbers. Memory speed and the video card make the most difference, so the inter-machine comparisons may not be very valid.

The compiler optimisations do not make nearly as much a difference as using a good algorithm in the first place. The largest difference occurs when the starting point is a very sloppy implementation. Having a faster processor helps, but once again, the change is swamped by the effect of selecting a better algorithm. The tuned version on the slow 386 outperforms the sloppy version on the higher speed 486, and the tuned 486 code outperforms even the best C on the Pentium! These days many programmers are becoming too lazy and assume that throwing money (more memory and a faster processor) at a problem is a solution, when the application of common sense and programming skills should be the preferred option.
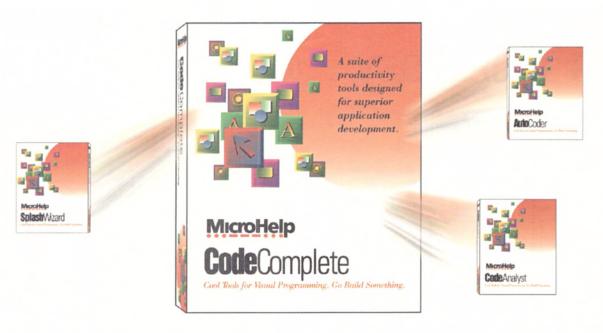
But does the code work? Before getting too excited, this fairly fundamental question must be considered. The answer is 'yes and no', since it only does part of what was wanted. I'll explain in the second part of this article next month. ■

Gavin Smyth is a real time software engineer and a part time Linux hacker.

```
const int vidInt   = 0x10;  // Interrupt num for video fns
const int setMode  = 0x00;  // Function code to set video mode
const int getMode  = 0x0F;  // Function to read video mode

int vidInterrupt( int fn, int al = 0 )
{
  union REGS r;
  r.h.ah = fn;
  r.h.al = al;
  int86( vidInt, &r, &r );
  return r.h.al;
}

const long numIterations = 10000;
const long asmScale      = 10;

int main()
{

  int oldMode = vidInterrupt( getMode );

  vidInterrupt( setMode, 0x13 );

  long i;

  clock_t start = clock();

for( i = 0; i < numIterations; i++ )
    alien.plotSlowC( random( 200 ), random( 150 ) );

  clock_t first = clock();

for( i = 0; i < numIterations; i++ )
    alien.plotFastC( random( 200 ), random( 150 ) );

  clock_t second = clock();

for( i = 0; i < numIterations * asmScale; i++ )
    alien.plotSlowAsm( random( 200 ), random( 150 ) );

  clock_t third = clock();

for( i = 0; i < numIterations * asmScale; i++ )
    alien.plotFastAsm( random( 200 ), random( 150 ) );

  clock_t last = clock();

vidInterrupt( setMode, oldMode );

const long mult = CLOCKS_PER_SEC * numIterations;

  cout << "Slow C: " << mult / ( first - start )
       << " per second\n"
          "Fast C: " << mult / ( second - first )
       << " per second\n"
          "Slow assembler: "
       << mult * asmScale / ( third - second )
       << " per second\n"
          "Fast assembler: "
       << mult * asmScale / ( last - third )
       << " per second\n";

  return 0;
}
```

Listing 9 – A short program to blast lots of Images on to the screen in random positions.

The Whole
is greater than
the sum of its Parts!

A suite of productivity tools designed for superior application development.

MicroHelp
CodeComplete

Cool Tools for Visual Programming. Go Build Something.

But when they're from MicroHelp, even the parts stand apart — like these hot new development tools.

Splash Wizard will let you do more than just add splash screens to your applications! Use it to add complete version checking to your applications, which is done before it loads. Check for correct versions of OCXs, VBXs, DLLs, even the version of your own EXE. You can also check for a multitude of other items such as licensing requirements, a previous instance of your application or a pre-defined required level of system resources. All exceptions are optionally logged to a file of your choice to dramatically simplify your technical support and conflict troubleshooting.
List price: £75

Code Analyst dissects your Visual Basic® code and gets down to business. Use its powerful cross-referencing engine to generate detailed reports about your projects, variables and functions. Or use the innovative code review feature to compare code to a user defined set of standards and generate a report of all exceptions. And when you are ready to generate that EXE, use the Code Compression Wizard to remove all dead (unused) code to tighten up the shipping executables.
List price: £99

AutoCoder is a versatile Visual Basic 4 Add-In that enables you to insert fully customizable templates into your code. Simplify your team coding efforts by standardizing module and function headers. Create new templates to do everything from comment blocks to profiling! AutoCoder will also implement global error trapping across your applications automatically.
List price: £55

Add them all together, and you've got **Code Complete**: the not-to-be-ignored suite that includes this versatile trio **plus** three special assistants: *Message Assistant* — create message boxes in a snap with our Message Box Assistant. Specify message box parameters such as styles, icons, and buttons. *Common Dialog Assistant* — simplifies building Common Dialog Boxes by allowing you to specify the dialog type and parameters. It then generates all the Visual Basic code for you. *HelpContextID Assistant* — scans your Visual Basic 4 project and generates a list of HelpContextIDs to simplify connecting your application to its Windows Help file for context sensitive help.
List price: £195

That combination makes Code Complete greater than the sum of its parts in every way - except price. You'll save nearly £40 when you buy Code Complete! To order or for more details, call today.

To order call: **07000 422 224**

CONTEMPORARY
software

Contemporary Software
Tel: 07000 422 224. Fax: 07000 422 225 E-mail: cssales@contemporary.co.uk
Contemporary Software Limited, Kingswick House, Sunninghill, Berkshire SL5 7BH

MicroHelp
Component Products Division
Windows® and Visual Basic® are registered trademarks of Microsoft Corporation.

cool tools for Visual programming.
go Build something.

CIRCLE NO. 708

M⊙ving targets
AIMING FOR AN ANIMATED WEB

Serendipity is one of the really enjoyable features of the Web. Last month I went on an extensive tour while researching HTML 3.2 for my June column, and en route saw many pages barely related to my topic. The search engine had pointed them out for some reason so I had a look.

I discovered an explosion of pages that contain some form of moving object. Some were generated by Java, but as a user I find Java applets a pain. They often give me an error about an inability to assign a colourmap on my Solaris 2.5 system using CDE. I have to click the OK button in the dialog that tells me this, to say, yet again, 'fine, I know about that'. After this little action the beast is appeased and continues with no obvious problems.

I remain very queasy about the general safety of Java. Several stories have been posted in the `comp.risks` newsgroup about how the much-vaunted system's security has been breached. Netscape had to rush out a new version of its browser because security was simply not implemented. The spectre of the unscrupulous programmer, remorselessly introducing nasty programs into my system, has appeared despite the assurances of Sun.

I freely admit that my system is not the fastest machine available, but the biggest problem with Java applets is that they take a long time to start running. I'm not sure I benefit from this Java class. It makes me wait for ages, and all it does afterwards is animate a little picture.

The majority of Java applets I download are based on examples available on the 'net. The page author has pulled the compiled code, and sent it back up with a bit of his own tailoring. The same prescriptive process is true of the dreadful JavaScript application that loads a text string into the bottom strip of the Netscape page. It would not be so bad if the image produced scrolled smoothly without flickering and juddering – but it doesn't. The tiny and generally unreadable text flashes and flickers across the screen, getting in the way of the normal feedback of the URLs the page contains.

Remember GIFs

Most animations on the Web are not esoterica based on Java, but are animated GIFs, a series of frames encapsulated in a single graphics image. The GIF file format was invented by CompuServe, and was intended to create an image storage format that could be used to ship multiple images between

Web authors everywhere are pouring Java into their pages, but **Peter Collinson** prefers to serve up a round of traditional GIFs to add some life to the 'net.

machines in a single file. Successful in that goal, GIF is probably the most widely used format for graphics interchange.

There is, however, a snag. GIF uses the Lempel-Ziv-Welch (LZW) data compression algorithm to reduce the size of the file. The algorithm has been the subject of much legal battle [see *Patent grief in GIF city*, *EXE* May '95]. Although LZW was already widely used, Unisys, the patent owner, decided in 1994 to charge licence fees for software incorporating the algorithm. CompuServe was granted a licence to sublicense late that

Figure 1 – Alchemy Mindworks GIF Construction set.

Early Web programs supported inline GIFs specified to the GIF87a standard. Other graphics formats tended to be supported in 'helper applications', so an extra window would pop up to show photographs encoded with the JPEG standard. Netscape initially followed this lead and supported only GIF, but the interlace feature of early Netscape browsers used the original GIF87a format to advantage.

GIF87a uses a four pass interlace scheme to split the image into raster sections. The first pass starts at row zero of the image, then uses every eighth row. The second pass starts on the fourth row and uses every eighth row of the data. The third starts on row two using every fourth row. The final pass starts on row one and uses every second row. Netscape implemented GIF interlace so that an image could come in from the net and be painted in progressively greater detail onto the screen, but its efficacy when applied to Web pages has caused debate. Some designers prefer to see nothing displayed until the image is completely present. Others argue we are still at a stage where it is important to reassure Web travellers that something is happening while they are waiting. I usually choose interlaced GIF images for large bitmaps.

Enter GIF89a

New in GIF89a were specified blocks of information to add, optionally, to GIF files. Functionality increased with the introduction of extensions for graphics control blocks, plain text, comments, and applications.

The graphics control block extension allows the definition of a transparent background, a feature used to great effect on the Web. The block can contain information describing what should happen to the image when it is removed from the screen, just before being replaced by another image. For example, this option allows the overlaying of graphics. The graphics control block also
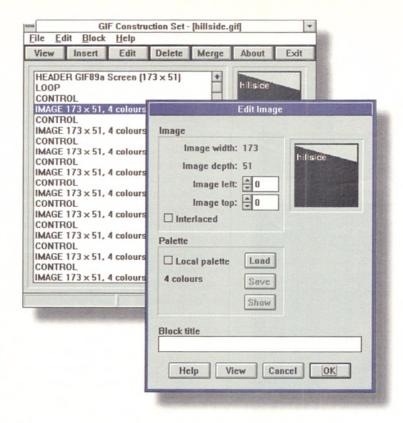
year. End users and non-profit organisations are exempt, but many software development companies were required to pay the fee. Since several of them chose not to, some graphics programs are unable to generate bitmaps using GIFs.

The GIF standard comes in two flavours. GIF87a is the original definition, which was later augmented by the GIF89a specification (I'll leave it to you to deduce the significance of the numbers). GIF87a specified a file format with a header followed by one or more image definitions. The format allowed the storage of 8-bit images supporting 256 colours, which generate reasonable pictures on many display systems.

Most workstations support a maximum of 256 distinct colours. Common PC displays have the same limit. However, it is not the monitors that prevent a greater variety of colours; it is the underlying graphics systems that store the pictures. On my Windows 3.1 system I prefer to use the memory on my graphics card for displaying high resolution images. It shows more on the screen – at the expense of the number of distinct colours displayed simultaneously. The emergence of 'true colour' monitors introduced full 24-bit colour display. This equipment, with more memory in its graphics cards and faster hardware to move the additional bits, is becoming widespread.

Paul Phippen

contains a delay timed in hundredths of a second which can retain an image on the screen for the specified period while the next image from the file is processed.

The plain text extension allows ASCII stored in a GIF file to be rendered on top of a bitmapped image, and was intended to enable captions to be placed over graphics. The comment extension permits a block containing human readable ASCII text to be embedded in the file. This can be used to comment the GIF file, often to assert ownership and copyright.

Finally, the application extension facilitates the storage of data understood only by the software reading the file. Netscape employed this feature to add a definition for a looping image, permitting cyclic animations to run from a cached copy of the GIF. It is possible to supply a loop count to limit the

number of cycles the animation will make, but Netscape browser versions after Beta 5 of release 2 simply ignore the count. Animations will loop endlessly.

When a browser finds several images in a single GIF file, they are loaded in the fashion dictated by the graphics control block... or nearly. Each image is placed on the screen in turn, creating animation. Initial loading speed is dictated by the network connection, so animations may not be completely satisfying on the initial load. However, the animation effect may improve once the GIF is resident in Netscape's cache.

A defined processing delay is not a good way of expressing frame rates. The same image iterates much more quickly on my Sun Sparc 2 running Solaris than it does on my 486/66 Windows 3.1 machine. Even on the Sun the Netscape image is slow com-

pared to the image displayed by an animation viewer such as `xanim`. Although all three have the same delays, the graphics update mechanisms in the different environments cause speed differences.

To animate you need to create a set of individual GIF frames that will become the animation when glued together into a single file. I used the `xpaint` program on UNIX to make objects move from frame to frame in my sample animation, but Paintshop Pro or Corel Photo-Paint could be applied. The Corel program cannot generate GIF files, so images must be passed through a program that will take a convenient intermediate format and create a GIF for each image (perhaps the Portable Bitmap suite – `pbm`).

Alternatively a program can be written to generate the images. My son wanted to move an object across the screen, and was quickly bored by the repetition of doing so by hand. However, with Thomas Boutell's excellent GIF creation graphics library `gd1.2` it was not difficult to write a program that executed a series of circular shifts on the image. The tool is exactly what's needed to create the animated scrolling text strips many people create in Java. It makes a file for each frame and names it sensibly, so shell wildcards can be used to ensure that the correct file sequence is followed in the final animated GIF. The program is available on my FTP server; see the box to the left.
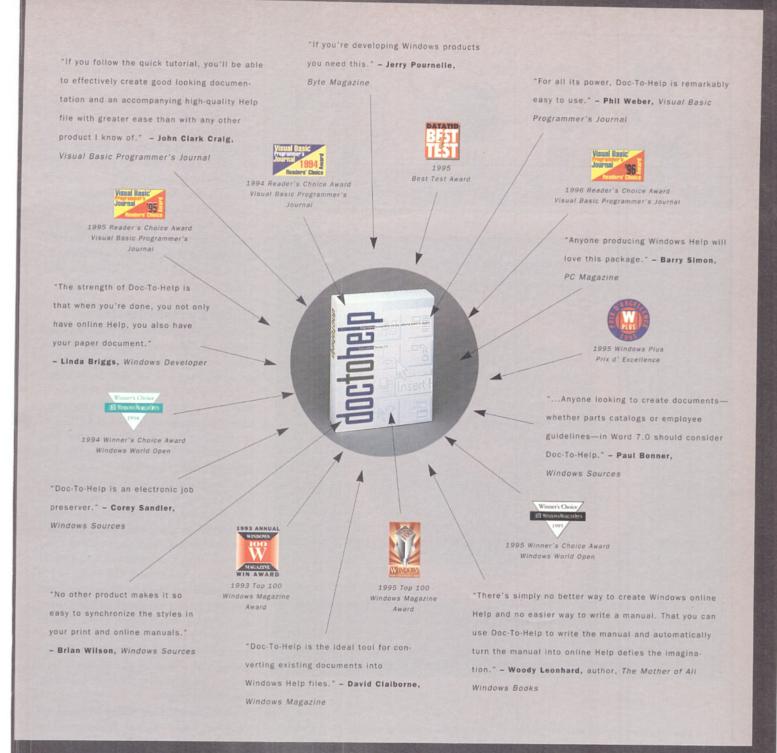
GIF89a tools

The Netscape browser implements nearly all of the GIF89a standard. Given the correct tools to set up a GIF file, GIF features can be used to obtain the various effects inline on the displayed Web page. The first tool widely available on Unix platforms was `giftrans`, written by Andreas Ley from the University of Karlsruhe in Germany [see *Pictures on the Web, EXE*, June '95]. With `giftrans` colours for a transparent background can be set.

Several tools are available to take a collection of GIF images and glue them together into a single animated package. On Unix I use `WhirlGif`. This free tool was written by Kevin Kadow based on the program `txtmerge`, originally part of Mark Podlipec's `xanim` distribution. Note the `WhirlGif` tool doesn't provide control over what happens when a particular graphic image is destroyed. It expects you to line up a sequence of GIFs that will occupy the same space on the screen, each image completely overwriting the previous one.

Control animations with options set through `WhirlGif` arguments:

`-loop` supplies the application control block, making Netscape restart the animation automatically. The argument can be followed by a count, which Netscape will ignore;

Thomas Boutell's GD library

Boutell's library, originally designed to create dynamic GIF images on the fly for HTML applications, is available on the net. The library is written in ANSI C and compiles on my Sun without problems. It should do the same in a DOS/Windows environment, although I have not checked. Versions of the library can be embedded into Perl and Tcl, if C is not your métier.

The gd library provides a reasonably well featured set of graphic primitives, with lines, arcs, text, multiple colours, cut and paste from other images, and flood fills. It interfaces to the standard I/O library, making input and output of images trivial. Here's a fragment of code that reads in a GIF and obtains its size:

```
/* fin is a FILE that is open */
im = gdImageCreateFromGif(fin);
/* get size */
width = gdImageSX(im);
height = gdImageSY(im);
```

The im value is a gdImagePtr, defined in the standard header file. All the routines and macros start with the letters 'gd', so there is little danger of name-space pollution.

The graphics routines are not surprising. You can set pixels to specific colours, draw lines, rectangles, polygons, arcs, and more. You can specify a fill for a closed image. With style setting routines you can select a drawing style for several operations.

The library, recommended if you plan to do image manipulation for the net, is documented by a single HTML file. You can point your browser at the file and use hyperlinks to look up routines.

-time is followed by the number of 0.01 second intervals to elapse before each image is replaced by the next one; and

-trans is used to specify the background transparency value of the GIF by invoking either a 24-bit colour value or a colour map index value.

The remaining parameters are used to specify input and output files. The facility to optionally give WhirlGif the name of a sequence file containing a list of input files is particularly useful. It has allowed me to control the sequencing of GIF files that make up the animation with ease.

The good thing about WhirlGif is you can set up an animated GIF using a batch script or a Make file. Even a small animation can consist of a great many frames, and dealing with these one at a time is very tedious. With WhirlGif you can re-edit the control file to set a new animation sequence. This lets the programmer concentrate on each frame to try different effects until the final sequence is correct.

Animated GIF construction for Windows 3.1 and NT is supplied by a program from Alchemy Mindworks called the *Gif Construction Set*. It offers much greater control over individual blocks in the GIF than the WhirlGif program. The basic window displays the contents of the Gif. You can add or delete image sections or control blocks, and the contents of each block in the image can be edited to change their characteristics.

In Figure 1 I loaded the output from the WhirlGif program into the editor for demonstration purposes. The separate sections in the GIF file appear clearly. I've clicked on the IMAGE block, and could change its values using the smaller edit win-

dow. The GIF is too small for the display window within the edit frame, and has been extended to make it square.

The edit window allows me to change the positioning of the individual GIFs within the display frame. This feature is not used by WhirlGif, so the offsets, with which I could animate a small object moving in a larger frame, are set to zero. It would be a long and arduous task to move one small image across a larger background while specifying offsets for each step of the way. However, overlaying new images on old ones using small additional animated images could significantly reduce the size of the final GIF, making the completed animation much more viable.

Efficiency considerations

You can easily generate huge GIF files when animating. When creating pages for my server I always consider how much data I am sending to users, so I spend time reducing the sizes of graphics files to help to deliver the images in a reasonable time. One simple way to reduce the size of a GIF is to cut down on the number of colours used. The pbm package includes the handy tool ppmquant which does this.

The small animation on my server comprises a simple four colour image of 48 frames. I didn't reduce the colours; the image started life as a piece of PostScript that I converted to a monochrome bitmap some time ago. It uses an additional colour for its transparent background. The image is 173 by 51 pixels, and each frame is around 400 bytes. The total animated GIF is around 18,000 bytes, which seems about right in terms of the load it imposes for something purely decorative. It may be too big. Most people are accessing the Web via 14.4 modems which require about ten seconds to transfer the image.

It is important to remember that not everyone uses Netscape. Many browsers support the animated GIF features, but your pages will always have visitors whose browsers cannot cope. Some will display only the first frame of your animation, and some, perversely, will show the last. Because of this I've made sure that the first and last frames of my animation contain an image that works when displayed statically.

GIF animations are ideal for adding life to Web pages. With the format's own extensions, a few extra tools, and a dollop of creativity, you may even catch the occasional wayward surfer-by. ∎

Peter Collinson is a freelance consultant specialising in Unix. He can be reached electronically as pc@hillside.co.uk, by phone on 01227 761824, or on the Web at http://www.hillside.co.uk

Portability everyone can agree on

"Zinc's portability is transparent. And only Zinc has a full set of UI objects and real extensibility to develop commercial-grade applications. Full source code is a big plus, too."

DEVELOPMENT MANAGER

"We're now shipping our application on every major platform and in every major world market. Our sales force loves it! With Zinc we're beating the competition."

MARKETING MANAGER

"Zinc cost us very little up front and saved us years of development expense. Revenue is up, thanks to the new markets [Zinc enabled us to enter]. I'm very satisfied."

FINANCE MANAGER

When was the last time Development, Marketing, and Finance all agreed on anything? Now's your chance to make it happen. With Zinc you'll build better applications, on more platforms, in less time, and with less money—and that's a promise.

Only Zinc offers complete portability.

Since Zinc Application Framework is the only cross-platform tool that delivers 100% portability, you'll have your applications on other platforms as fast as you can recompile. And no one supports as many platforms as Zinc does. It's all part of what makes Zinc the most productive—and affordable—tool you can own.

Productivity that leads to opportunity.

Zinc zips through tedious tasks with C++ object orientation and a unique visual development tool. And, enabling your application for international markets is already done—just translate your text. Plus, Zinc is the only tool that supplies 100% of the source code.

It all adds up to productivity. Which means more profitability. Which means everybody's happy—especially you.

PCWEEKLABS
ANALYST'S CHOICE
JAN. 30, 1995

"Developers seeking easy delivery of GUI applications... will find Zinc their best option by far."

"This product is absolutely the best development environment I have personally seen for the international engineer."

Multilingual Computing

EDITOR'S CHOICE '95
MLC

"Best Portability"
INFOWORLD
February 6, 1995

"Zinc came closest of all the products we tested to our ideal of portability... In short, Zinc did a great job."

For free demonstration software and an information packet, please call:

+ 44 (0) 181 855 9918

USA: +1 801 785 8900 or fax +1 801 785 8996
Europe: +44 (0) 181 855 9918 or fax +44 (0) 181 316 2211
Asia: +81 (052) 733 4301 or fax +81 (052) 733 4328
Electronically: Info@zinc.com or GO ZINC on CompuServe.
Web: http://www.zinc.com/ Ftp: ftp://ftp.zinc.com/

z i n c

NO LIMITS

CIRCLE NO. 714

How much **time** are you spending learning your configuration management system......

instead of actually using it?

You've made the decision to use software configuration management (SCM). And you know that a streamlined development process can help you meet your deadlines and make your developers more productive. So why are you still figuring out how to use the software? You need an SCM tool that produces results instantly.

MKS Source Integrity

At last, there's a software configuration management system that eliminates the learning curve. MKS Source Integrity orchestrates your entire development process, providing your team with a choice between a stand alone GUI and command line interface, across 20 different development environments. Color visual merging and differencing show changes to files at a glance. Sandbox environments provide each developer a safe place to do their daily work.

Thinks and works like development teams do

MKS Source Integrity is a complete SCM system designed to maximize your team-based development. More than 45,000 developers worldwide use MKS Source Integrity's complete suite of tools to help them increase productivity, protect software assets, and guarantee overall source code integrity.

To find out how MKS Source Integrity can accelerate your team's development, call today for your FREE demonstration copy, or download it from our website.

Your challenges are our challenges. All people depicted in this ad are MKS personnel.
30-day unconditional money back guarantee.
MKS, MKS Source Integrity and Sandbox environments are registered trademarks of Mortice Kern Systems Inc. All other trademarks acknowledged.

"Would you like to know how MKS Source Integrity integrates with Microsoft C++? One word: seamlessly."
- LAN Times, August 95

Source Integrity provides full integration with leading development environments including:
- Borland C++
- Borland Delphi
- Microsoft Visual Basic
- Microsoft Visual C++
- PowerBuilder
- Watcom C++

Call today!
0171 624 0100
http://www.mks.com
Email: uk@mks.com

MKS
MORTICE KERN SYSTEMS INC.

Mortice Kern Systems (UK) Ltd
239 Kilburn Park Road
London NW6 5LG
Fax: 0171 624 9404
Alternatively, contact:
System Science
Tel: 0171 833 1022
Grey Matter
Tel: 01364 654 100
Admiral
Tel: 0276 692 269

CIRCLE NO. 715

Task bar à la carte

With QuickStart the functionality of Windows 95 reaches new heights. The small utility, created by **Dave Jewell,** gets around some of the niggling shortcomings of the TaskBar by letting users keep their windows open.

Hulton Getty

Not long ago, when the world was young and the ink was still drying on the Windows 3.1 documentation, I wrote a little program to give me a bit of added convenience when working with Ye Olde Program Manager. The program was published in certain computer magazines in the UK, and given the names Pop-Group and DeskPop by various editors.

The idea was simple. Basically it scanned all the Program Manager Group (.GRP) files in the Windows directory, looked inside the group files to determine the contents of each group, then created a simple hierarchical popup menu activated by clicking the mouse button anywhere on the Windows desktop. The program was implemented by sub-classing the desktop window and looking for right-button mouse clicks.

I was very flattered when the author of the popular DeZkTop shareware program told me it was DeskPop/PopGroup which inspired his initial efforts. The rest, as they say, is history...

Wherefore art thou, desktop?

Just as the Windows 3.1 shell had deficiencies, the Windows 95 Explorer is not perfect. I will even go so far as to suggest that the more time and money Microsoft claims to spend on usability studies the less useable Explorer becomes. Consider the much-criticised TaskBar. It's cool that it slides into view when the mouse moves to the edge of the screen. It's certainly a cool way of switching to another application – but less convenient than Alt-Tab key press, which does the same far more quickly, without having to reach for the mouse.

Despite your preference for the TaskBar or Alt-Tab, switching away from a maximised application doesn't automatically uncover the desktop. When any application is maximised the desktop is completely obscured. Even if the active application is running in a small window, any other maximised window will obliterate the desktop. This is an unfortunate shortcoming. A nice feature of Windows 95 is the ability to place folders and shortcuts on your desktop, but the current user interface makes it difficult to get at these shortcuts in a hurry.

If an application is iconised, switching *to* it will automatically restore its window to the desktop. Yet switching *away* from an application does not automatically re-iconise it. Many people would find such a default convenient. Microsoft could easily have arranged things that way, for example, Alt-

Tabbing to an application could keep it on the desktop only as long as it's active, whereas selecting the same application from the TaskBar could ensure it would 'stay put' even when inactive. Users would have the best of both worlds. Alternatively a new set of properties could control the behaviour of an application relative to the desktop and TaskBar. Microsoft did neither.

Enter QuickStart...

Enough philosophical mutterings – it is easy to be wise after the event. Figure 1 shows my approach to circumventing the problem. It is a close-up view of the 'tray' area of the TaskBar. The installed applets are the Microsoft Volume Control, the Microsoft QuickRes utility, and – ready and able at all times – QuickStart. When the QuickStart

Figure 1 – The QuickStart icon, third along, installed in the tray area of the Windows 95 TaskBar.
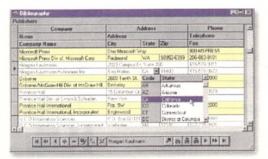
NOW DESIGNING DATABASE FRONT ENDS IS SO EASY IT'S EMBARRASSING.

Introducing Data Widgets 2.0

Data Widgets 2.0

Dramatically increase the productivity of your Microsoft® Visual Basic® 4.0 application development with Data Widgets 2.0 from Sheridan Software. Data Widgets 2.0 is a set of bound OLE controls that lets you design front ends for database applications in the shortest amount of time with all the ease and power you've come to expect from Visual Basic.

Data Widgets 2.0 is designed to optimize your Visual Basic 4.0 database front ends, supplied as 16- and 32-bit OLE Controls. Included are a fully editable DataGrid, a DataCombo (the dropdown and edit portions can be bound to different databases), Enhanced Data Controls, a DataDropDown, a Data Command and a DataOptionSet Control.

Drop-In Replacement for Visual Basic 4.0's grid.

DataGrid™ *(SSDBGrid)*
- Functionally and visually consistent with data grids in Microsoft Access and Visual Basic 4.0.
- Support for movable groups and columns.
- Optional dropdowns in headings allow users to select from a list of available fields and/or groups at run time.
- Additional cell types include checkbox, button, label, and combo box.
- Multiline row formats.
- Pictures and text in cells and headings.
- Individual fonts and colors for columns, rows, and cells.
- Full design time capabilities.
- Supports multiple data modes including bound, unbound, and AddItem (design and run time).

Data Combo *(SSDBCombo)*
- Variable edit area height a la Microsoft Access.
- Multiline edit area.
- DropDown display is not limited to the width of the edit area.
- Same formatting capabilities as the SSDBGrid control.
- Full design time capabilities.

DataDropDown *(SSDBDropDown)*
- Used in conjunction with the DataGrid.
- DropDown display is not limited to the width of the edit area.
- Same formatting capabilities as the SSDBGrid control.
- Full design time capabilities.

Data Option Set *(SSDBOptSet)*
- One control creates an unlimited number of option buttons that are bound to the same data field.
- Multi line captions and pictures.
- Automatic and manual button positioning.
- Custom color options.

Enhanced Data Control *(SSDBData)*
- Re-position recordset by selecting from a drop down list of up to 100 previously marked rows.
- Store and sort multiple bookmarks in the bookmark drop down list.
- Buttons that perform database actions such as add, delete, update, plus bi-directional nth record paging.
- Conditional and Soundex searching.
- ToolTips

Data Command *(SSDBCommand)*
- Multiline captions and pictures.
- Add, Delete, Refresh, Bookmark, and Auto-Positioning functions.
- Click and AfterClick events.
- Auto-Repeat functionality.
- Custom color options.
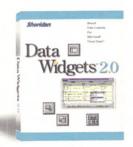
Tel: 07000 422 224

Sheridan

Reusable Components and Productivity Tools for the Visual Developer

Visual Basic Programmer's Journal '95 Readers' Choice Award

Thunderbolt Award Winner

Data Widgets 2.0

List Price:
£99.00

✎ CIRCLE NO. 716

CONTEMPORARY
software

Comtemporary Software Ltd
Kingswick House, Sunninghill, Berkshire SL5 7BH
Tel: 07000 422 224 • Fax: 07000 422 225
E-Mail: cssales@contemporary.co.uk

icon is clicked a hierarchical menu appears, like the one shown in Figure 2. Desktop folders appear first in the menu, followed by top-level menu items. Next is a pair of options to kick off Explorer and open the desktop as an Explorer window. Last are a menu item for an About box (which I've left empty) and an option for exiting QuickStart.

Desktop folders are grouped in a sub-menu containing a series of options relating to the folder contents, which may of course include other folders. The original DeskPop program merely reflected Program Manager's group file setup, which does not allow nested groups. Because Explorer recognises nested folders QuickStart had to be more flexible than its predecessor. It uses a recursive routine to build the menu.

I run a relatively fast machine, so I've designed QuickStart to rebuild the popup menu each time the QuickStart icon is clicked on. No matter how often items are added or removed from the desktop, the popup menu will always be current. However, if you have a slower machine, or complex, deeply-nested desktop contents, you may prefer to have the menu built only when QuickStart is initialised. The change is very simple to implement.

How does it work?

The source code for QuickStart is given in Listing 1. I built the code with the Delphi 2.0 compiler. Because of the various API calls that are used, notably `Shell_NotifyIcon`, it must be a 32-bit application. However, by using Delphi in 'barefoot mode' and ignoring the VCL library, the size of the executable file is a mere 16 KB.

The source code is available on *EXE's* FTP site at *ftp://ftp.exe.co.uk/pub/exestuff/quickstart/*. You should have few problems porting it to C or C++, but bear in mind the source code for the StrRScan and StrLI-

Comp routines is not included because these routines form part of Borland's copyrighted run-time library. If you're using Delphi 2.0 you can extract them from the SYSUTILS unit. To keep the size of the program down I physically copied the necessary code into the source file, rather than referencing the unit in the normal way. For the benefit of C/C++ programmers StrRScan is a reverse string

```pascal
program DeskPop;

uses
  Windows, Messages, ShellAPI;

{$R *.RES}

const
  AppName = 'QuickStart';

var
  CmdCount: Integer;
  tid: TNotifyIconData;
  Commands: array [0..255] of PChar;
  szDesktopFolder: array [0..Max_Path - 1] of Char;

function GetDeskTopFolder (szDest: PChar; DestLen: DWord): Bool;
const
  ShellFolders = 'Software\Microsoft\Windows\
                  CurrentVersion\Explorer\Shell Folders';
var
  key: hKey;
begin
  GetDeskTopFolder := False;
  if RegOpenKeyEx(hKey_Current_User, ShellFolders, 0, Key_Read, Key) = 0
  then
  begin
    if RegQueryValueEx (key, 'Desktop', Nil, Nil, Pointer(szDest),
                        @DestLen) = 0 then
    begin
      lstrcat (szDest, '\');
      GetDeskTopFolder := True;
    end;

    RegCloseKey (key);
  end;
end;

procedure Panic (szMessage: PChar);
begin
  if szMessage <> Nil then
    MessageBox (0, szMessage, AppName, mb_ok);
  Halt (0);
end;

function StrRScan (Str: PChar; Chr: Char): PChar; assembler;
asm
  { See Text }
end;

function StrLIComp (Str1, Str2: PChar; MaxLen: Cardinal)
         : Integer; assembler;
asm
  { See Text }
end;

procedure AddCommand (Menu: hMenu; Folder, fName: PChar);
var
  p: PChar;
  szBuff: array [0..Max_Path - 1] of Char;
begin
  // Build the full pathname
  lstrcat (lstrcpy (szBuff, Folder), fName);
  // Copy the full pathname into the Commands array
  GetMem (Commands [CmdCount], lstrlen (szBuff) + 1);
  lstrcpy (Commands [CmdCount], szBuff);

  // Now cleanup the command name for the menu
  p := StrRScan (fName, '.');
  if p <> Nil then
  begin
    if StrLIComp (p, '.LNK', 4) = 0 then p^ := #0;
    if StrLIComp (p, '.PIF', 4) = 0 then p^ := #0;
  end;

  AppendMenu (Menu, mf_String, CmdCount + $1000, fName);
  Inc (CmdCount);
end;

procedure AddDesktopItems (Menu: hMenu; szFolder: PChar);
var
  p: PChar;
  h: THandle;
  hSubMenu: hMenu;
  dirs, scanCount: Integer;
  findData: TWin32FindData;
  szBuff: array [0..Max_Path - 1] of Char;
  szBuff2: array [0..Max_Path - 1] of Char;
begin
  dirs := 0;
  with findData do
  begin
    lstrcat (lstrcpy (szBuff, szFolder), '*.*');
    for scanCount := 0 to 1 do
    begin
      h := FindFirstFile (szBuff, findData);
      if h <> -1 then
      begin
        repeat
          if cFileName [0] <> '.' then
          begin
            if (scanCount = 0) and
               ((dwFileAttributes and file_Attribute_Directory) <> 0) then
            begin
              Inc (dirs);
              hSubMenu := CreateMenu;
              lstrcat(lstrcat(lstrcpy (szBuff2, szFolder), cFileName),'\');
              AddDesktopItems (hSubMenu, szBuff2);
              InsertMenu (Menu, -1, mf_ByPosition or mf_Popup,
                          hSubMenu, cFileName);
            end
            else if (scanCount = 1) and
                    ((dwFileAttributes and file_Attribute_Directory) = 0) then
            begin
              AddCommand (Menu, szFolder, cFileName);
            end;
          end;
        until not FindNextFile (h, findData);
        Windows.FindClose (h);
        if (scanCount = 0) and (dirs <> 0) then
```

Listing 1 – The source code for QuickStart (continued on next page).

scan (like `strrchr`); `StrLIComp` performs a case-insensitive string comparison for a specified maximum number of characters.

Windows 95 Explorer's implementation of the desktop is essentially disk-based. The desktop itself corresponds to a single sub-directory, the desktop folder. Desktop short-cuts correspond to .LNK files placed in this sub-directory, and sub-directories which hang off the desktop folder will likewise appear as folders on the desktop. Certain items such as 'My Computer' and 'Recycle Bin' are implemented differently, and do not correspond to files in this directory structure.

Most machines have Windows 95 installed on drive C:, and in most cases the desktop folder will be C:\WINDOWS\DESKTOP, but obviously it is a bad idea to assume this is true in every case. The program contains the `Get-DeskTopFolder` routine, which interrogates the registry to determine the current desktop folder location by looking at the value of the

'Desktop' sub-key in the following location:

```
'\HKEY_CURRENT\USER\Software\
Microsoft\Windows\CurrentVersion\
Explorer\Shell Folders'
```

Once the desktop folder is located it is amusing to sit there in a windowed DOS box, adding and deleting sub-directories and files, while watching the Windows 95 desktop adjust to reflect the changes. A delay of a second or so before Explorer notices them is usual – perhaps Explorer receives file-change notifications in a manner similar to Windows 3.1's File Manager.

By tweaking this value under program control it is probably possible to write a utility capable of maintaining multiple desktops. However, writing such a program would require a way of getting Explorer to rebuild it's 'view' of the desktop once such a change has been made. It would be far sim-

pler to leave the current desktop folder location alone and instead replace its contents with an alternative desktop arrangement as needed. Explorer would automatically take care of redisplaying the new desktop.

Creating a dummy window

In common with similar utilities QuickStart creates a hidden dummy window. It can't

```
        AppendMenu (Menu, mf_Separator, 0, Nil);
    end;
  end;
  end;
end;

procedure HandleCommand (Wnd: hWnd; Cmd: Word);
begin
  case Cmd of
    Ord ('Q'): PostMessage (Wnd, wm_Close, 0, 0);
    Ord ('D'): ShellExecute(0,Nil,szDesktopFolder,Nil,Nil, sw_ShowNormal);
    Ord ('E'): ShellExecute (0, 'Explore', 'C:\', Nil,Nil, sw_ShowNormal);
    else if Cmd >= $1000 then
      ShellExecute (0, Nil, Commands[Cmd and $ff], Nil,Nil,sw_ShowNormal);
    end;
end;

function DummyWindowProc (Wnd: hWnd; Msg, wParam: Word; lParam: LongInt)
          : LongInt; stdcall;
var
  i: Integer;
  pm: HMenu;
  pt: TPoint;
begin
  DummyWindowProc := 0;

  case Msg of
    wm_Create:        // Program initialisation - just set up a tray icon
    begin
      tid.cbSize        := sizeof (tid);
      tid.Wnd           := Wnd;
      tid.uID           := 0;
      tid.uFlags        := nif_Message or nif_Icon or nif_Tip;
      tid.uCallBackMessage := wm_User;
      tid.hIcon         := LoadIcon (hInstance, 'MAINICON');
      lstrcpy (tid.szTip, AppName);
      Shell_NotifyIcon (nim_Add, @tid);
    end;

    wm_Destroy:       // And now we face the final curtain...
    begin
      Shell_NotifyIcon (nim_Delete, @tid);
      PostQuitMessage (0);
    end;

    wm_Command:       // Command notification
    begin
      HandleCommand (Wnd, LoWord (wParam));
      Exit;
    end;

    wm_User:          // Had a tray notification - see what to do
    if (lParam = wm_LButtonDown) or (lParam = wm_RButtonDown) then
    begin
      GetCursorPos (pt);
      for i := 0 to 255 do
        if Commands [i] <> Nil then
          FreeMem (Commands [i], lstrlen (Commands [i]) + 1);
      pm := CreatePopupMenu;
      CmdCount := 0;
```

```
      AddDesktopItems (pm, szDesktopFolder);
      AppendMenu (pm, mf_Separator, 0, Nil);
      AppendMenu (pm, 0, Ord ('E'), 'Open Explorer...');
      AppendMenu (pm, 0, Ord ('D'), 'Open Desktop as a window...');
      AppendMenu (pm, mf_Separator, 0, Nil);
      AppendMenu (pm, 0, Ord ('A'), 'About ' + AppName + '...');
      AppendMenu (pm, 0, Ord ('Q'), 'Exit ' + AppName);
      SetForegroundWindow (Wnd);
      if TrackPopupMenu (pm, tpm_BottomAlign or tpm_RightAlign,
                              pt.x, pt.y, 0, Wnd, Nil) then
        SetForegroundWindow (Wnd);
      DestroyMenu (pm);
    end;
  end;

  DummyWindowProc := DefWindowProc (Wnd, Msg, wParam, lParam);
end;

procedure WinMain;
var
  Wnd: hWnd;
  Msg: TMsg;
  cls: TWndClass;
begin
  { Previous instance running ? If so, exit }
  if FindWindow(AppName, Nil) <> 0 then
    Panic (AppName + ' is already running.');

  { If we can't get the Desktop location, get out now }
  if not GetDeskTopFolder (szDesktopFolder, sizeof (szDesktopFolder)) then
    Panic ('Can''t find Explorer''s Desktop folder.');

  { Register the window class }
  FillChar (cls, sizeof (cls), 0);
  cls.lpfnWndProc := @DummyWindowProc;
  cls.hInstance := hInstance;
  cls.lpszClassName := AppName;
  RegisterClass (cls);

  { Now create the dummy window }
  Wnd := CreateWindow (AppName, AppName, ws_OverlappedWindow,
                       cw_UseDefault, cw_UseDefault, cw_UseDefault,
                       cw_UseDefault, 0, 0, hInstance, Nil);
  if Wnd <> 0 then
  begin
    ShowWindow (Wnd, sw_Hide);
    while GetMessage (Msg, 0, 0, 0) do
    begin
      TranslateMessage (Msg);
      DispatchMessage (Msg);
    end;
  end;
end;

begin
  WinMain;
end.
```

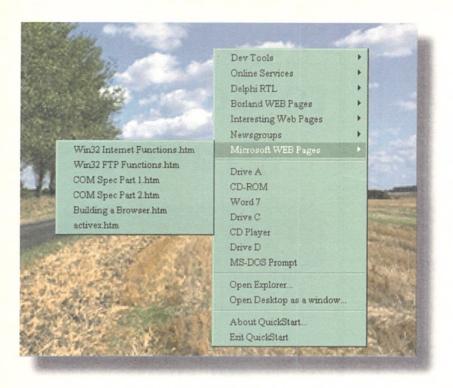*Listing 1 – The source code for QuickStart – continued.*

TECHNIQUES

Figure 2 – Invoking the QuickStart icon with the left or right mouse button will bring up a hierarchical menu similar to this one. Desktop folders appear as additional levels of sub-menu.

process messages in the normal way without it. When the program starts running it checks for the presence of a dummy window and – if it finds one – terminates. When the application window receives its `WM_CREATE` message it calls `Shell_NotifyIcon` to add an icon to the tray area of the TaskBar. This includes setting up a 'Tip' string (in this case just the name of the application) which tells Explorer what callback message should be used if the icon is clicked, and specifies the name of the window to which the callback notification should be sent.

At this point the program goes to sleep until a mouse click is detected, which prompts Explorer to send a `WM_USER` notification to the dummy window. If the notification message corresponds to a left or right mouse button click the program calls `CreatePopupMenu` to build an empty pop-up menu, calls the `AddDesktopItems` to add the contents of the desktop folder to it, then appends a set of standard menu items as described above. Finally it calls `TrackPopupMenu` to handle menu processing. Unless the user dismisses the pop-up menu with the Escape key, a `WM_COMMAND` message will be sent to the window procedure.

The `AddDesktopItems` routine is the most complex in the program. It must be recursive, because it is called each time a new level of folder is encountered on the desktop. If this had been a 16-bit application, allocating two large string buffers as part of the routine's stack frame might be risqué, but since it

is a 32-bit program this is not a concern. Realistically, if a desktop with folders nested 50 levels deep was encountered, Explorer would probably have curled up and died long before QuickStart got an opportunity to run!

Another problem is the need to make two passes through each folder. The first pass examines sub-directories and initiates a recursive call. The second examines files and calls the `AddCommand` subroutine to add each encountered file to the menu. Because of this, each level of folder has its sub-folders added first, followed by a menu separator, and finally the ordinary items. I don't understand why Microsoft hasn't added functionality to the `FindFirst/Next/File` routines allowing them to return file entries in this order. Oh well...

The `AddCommand` routine is relatively straightforward. It simply creates a string containing the complete path-name of the current item, allocates memory for it, and stores the string as the next entry in the `Commands` array. It also cleans up the command name by stripping .LNK and .PIF extensions. This could probably be improved by getting QuickStart to strip all file extensions which Explorer has been told to hide, but again, much poking around in the registry would be required to locate the information, a distasteful proposition.

When a `WM_COMMAND` message is received by the dummy window, control is transferred to the `HandleCommand` routine. It looks for the 'Q', 'D', and 'E' commands, call-

ing `ShellExecute` in the second two cases to invoke Explorer. When telling QuickStart to open the desktop as an Explorer window the initial window size is rather strange – it is only as high as the caption bar and the full width of the screen. I don't know why it does this, but it is only a minor irritation. Once the window is resized it will stay at its new size next time it is invoked, and will remain so until Windows is restarted. *(QuickStart will also work with NT4.x build 1314 or higher – Ed.)*

Installing QuickStart is dead easy. Simply copy it to the directory of your choice and add it to the startup folder so it starts running whenever Windows is booted.

In addition to the menu items described above you could experiment with others, and even add a fancy configuration dialog to allow the user to set things up 'custom'. Access to the desktop properties requires a right-click on the desktop before selecting properties from the pop-up menu, but as established above it is rather tricky clicking on the desktop when it's completely obscured. Fortunately it is very simple to invoke the desktop properties under program control. This is all you need:

```
WinExec ("Control DESKTOP", SW_SHOWNORMAL);
```

If you want to get really fancy, you can even invoke these Control Panel dialogs and ask to make a specific page active:

```
WinExec("rundll32.exe Shell32.dll,
Control_RunDLL desk.cpl,,3",
SW_SHOWNORMAL)
```

The statement above, though a bit of a mouthful, will invoke the desktop properties dialog and make the Settings page active.

After dining on QuickStart for a while, you may concoct ideas for some enhancements. Please let me know about them. Meanwhile, *bon appétit!* ∎

Dave Jewell is a freelance consultant, programmer, and technical author. You can contact him at DS.Jewell@aol.com, 102354.1572@compuserve.com, or DaveJewell@msn.com.

NEW VERSION Btrieve v6.15
Opening up to other environments

MS DOS

WINDOWSNT

OS/2 WARP

Microsoft® Windows 95

NetWare®

WINDOWS 3.x

Across the board

Developers, Analysis, Project Managers, you are already aware that Btrieve, the leading navigational client-server database offers high performance, maximum programming control, seamless scalability and maintenance free operations, Btrieve v6.15 has more to offer...

- Development scalability from palm-top to client-server.
  - Client : DOS, OS/2, Windows and Windows NT, Windows 95, Windows 3.x.
    - Server : NetWare, Windows NT, OS/2 LAN Server, WARP Server.
      - MicroKernel Architecture : Compatibility with the Scalable SQL relational database.

With the new version, Btrieve v6.15, you can expect a new surge in terms of performance, reliability, and extension capacity.

- Opening up to multiple platforms, total and definitive.
- 32 bit performance of Windows NT/Windows 95 applications.
- Coexistence and file-sharing with 16 bit applications.
- True Windows DLL "requester" loaded dynamically on demand.
- Flexibility and security.
- ODBC interface, retrieval of a set of records in a single function, referential integrity dynamic allocation of disk space, logging of events and transactions.
- Support for NetWare SFT III.

Get off to the best of starts ! Grab your chance today to join the leading developers, by asking for your "Update and Configuration Guide", which we have reserved for you. Simply call us, free of charge, at the number listed below.

Btrieve v6.15

It simply works

CIRCLE NO. 721

BTRIEVE
TECHNOLOGIES
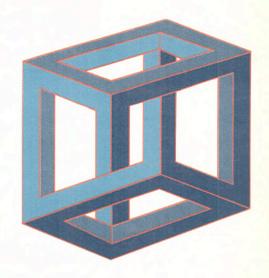
Les Collines de L'Arche, Opéra E - 92057 Paris La Défense 7 Cedex

📞 0800 96 5497 Tél.: 33-1-43 56 60 74 - Fax: 33-1-49 00 01 74

Complacency

Marketing punters, some worthwhile reading, and the encroaching millennium are **Francis Glassborow's** current concerns. If only someone else was paying attention to such matters...

How will software handle the turn of the century? As we approach the year 2000 the question becomes ever more pressing. Most programmers understand. Commercial and financial software with date data structures that store only the final two digits of a year will be cause for concern when the clock clicks over from 1999 to 2000.

Although serious the problem is well defined. To prevent catastrophe every line of Cobol code in programs using dates must be inspected. It is a cumbersome task. We must check for code written by programmers who have made implicit use of their knowledge of date structures.

This has nothing to do with C and even less to do with C++, but before people running software written in these languages assume they have no problems I suggest they look at the description of `time.h` in the Standard C Library. If programmers have rolled their own time and date functions worse problems are buried in the code.

Programmers have a natural tendency to use integer types for time and date information. Two typenames in ISO Standard C, `time_t` and `clock_t`, relate to this subject. They can be any arithmetic type, but virtually every library implements them as integer types rather than floating point types. It is quite common to use `long` for `time_t` and `int` for `clock_t`. The later is usually measured in units provided by the processor. The macro `CLOCKS_PER_SEC` indicates the number of units in each second. Calendar time is regularly measured in seconds from a base date such as 1/1/1980, but counting in seconds is not required.

For implementations of the Standard C Library time functions should be within a valid range. An `int` variable will wrap around quite quickly when a 16-bit `int` is counting at a rate of ten per second. Even a `long` is fairly limited for counting time in seconds, and almost useless for counting in smaller units. A 32-bit `long` will record only in the range +/- 68 years from the base date. How many programmers know the valid range of dates with the compiler they are using? When did you last check the effect of porting code from one compiler to another on date data?

Cobol programmers know the problem they must fix, and they have an exact deadline. C/C++ programmers may have no idea what problems lurk in their code. They should examine all source code to learn if it relies on `time.h`. If so, that reliance needs to be documented carefully. If it does not, but it still incorporates dates and times, programmers, while trying to determine what has to be fixed, will soon come to appreciate the benefits of object-orientation.

Make me an offer!

I sometimes despair of the software industry. Mediocre products become household names and best sellers, while excellent products languish almost unknown. I learned last summer that IBM was developing a Windows NT version of the VisualAge C++ development package. Since the OS/2 version was my favourite C/C++ development tool I rang IBM several times to enquire about the progress of the new package, but eventually I stopped asking. A few weeks ago I called again, and was told the software had already been released. I had not combed *every* magazine for release notices, but had been keeping an eye out for an announcement.

A version for NT and Windows 95 would be welcome news to me and other fans of the OS/2 version. IBM ought to have made a major effort to tell everyone about it. Why were they so quiet? Good products do not sell themselves, and winning awards does little to increase sales. IBM needs to beat the gong for VisualAge C++, not least because it includes a first-class set of visual programming tools that leaves 'wizards' and 'experts' a mile behind.
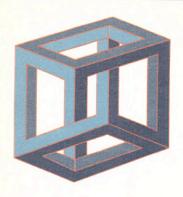
Other products have their merits, but unless you have spent some time trying VisualAge C++ habit is determining your purchasing policy. If you are a serious 32-bit developer using C or C++ on Windows NT, Windows 95, OS/2, Solaris, or possibly AIX or MVS, you should try the product. Ignore IBM's multimedia preview. It is an awful promotional piece reminiscent of TV adverts from the '60s.

Symantec is another company that has difficulty adequately promoting development products. The UK subsidiary seems completely uninterested in promoting its parent company's tools. Silicon River is far more informative and helpful with these products. Symantec's Café, a good package of Java development tools, has been released. You don't have to be an Internet fanatic to use Java – it is a good general purpose object-oriented language with strong relationships to C and C++. It is certainly easier than C++, although it requires yet another set of I/O functions.

Coffee-coloured deluge

The market is flooded with books about Java. The following three titles, fished from the cascade, are among those worth considering. *Java in a Nutshell* by David Flanagan (O'Reilly & Associates, ISBN 1-56592-183-6) is a top notch reference and deserves to be on every Java programmer's bookshelf. If you are quick you may still be able to get a copy at its original price of £10.95. The new price is £14.95.

Having a reference book is fine, but you may want additional tutorial text. Despite its misleading title, *Java Programmer's Library* by Sulei-man Lalani & Kris Jamsa (Jamsa Press, ISBN 1-884133-26-6) is an excellent book. I found only two flaws: at £32.49 (including CD) it is expensive, and file names on the CD are in 8.3 FAT format. Since Java relies on correct file names, they must be adjusted to long names.

Another useful book is *Core Java* by Gary Cornell & Cay Hortstman (Prentice Hall, ISBN 0-13-565755-5) at £32.95 including CD. It demands more effort from the reader, but if time is abundant working through it is rewarding.

The crop of recent C/C++ books is less bountiful, but two new volumes are worthy of shelf space. *More Effective C++* by Scott Meyers (Addison-Wesley, ISBN 0-201-633-71X, £22.95) is even better than his earlier *Effective C++*. Those who have read the latter will grab the new book. Anyone else who has progressed beyond novice C++ programming would benefit by reading both.

The second book is *Enough Rope to Shoot Yourself in the Foot* by Allen Holub (McGraw-Hill, ISBN 0-07-029689-8, £21.95). *ERSYF* is an annotation of the author's rules of thumb for C and C++ programming. You won't agree with all of them, but your programming will improve after reading and considering his views.

Finally, OS/2 programmers should check out the books Wiley has published on the subject. They have half a dozen excellent titles in their lists and lead the field.

Revisiting May's code problem

Here is the code again. Why is it defective?

```
/* various included headers */
enum BOOL (FALSE, TRUE);  // line A
int main () {
      cout << 0>1 ? FALSE : TRUE;  // line B
      return 0;
}
```

In June's commentary on this code I forgot one thing and I missed another.

I forgot to mention that my inversion of FALSE and TRUE in line B was pure whimsy. This was not really fair because such things should be commented, but only one correspondent raised the issue. At the very least such a counterintuitive line of code should be queried.

The other point is more instructive, as it illustrates the importance of checking code even if it passes a compiler. Old C++ compilers give an error message along the lines of 'cannot convert void * ...'. Originally iostream classes included a conversion operator to convert I/O objects to 'void *' to identify the state of an I/O object. You could write 'if(cout)' and it would evaluate to false if cout had failed in some way. When the Standards Committees made bool a built in type, the library committee replaced the operator void *() with an operator bool(). The change did not break existing code, but it did convert some previously ill-formed code to a well-formed variety. Line B above is an example of such code. Those with pre-bool version of iostream will get a diagnostic for line B. Those who use the most recent versions of iostream should also get a diag-

nostic since, due to such idiocies as line B, the Standards Committees have decided to revert iostream behaviour to its earlier form.

Gotcha!

Last month's code is error-free. Truthfully I had intended substitute the identifier 'remove' for 'release' to highlight the following problem. It should have read:

```
class Problem {
      int * ip;
public:        Problem (int size): i (new int [size]) {}
      ~Problem () {delete [] ip;}
      void remove ( int i); // replace the memory ip point
                            // to other member functions
};
```

Many programmers assume macro definitions will be written in uppercase letters. This rule should be followed so those using your code will have some hint that an identifier is a pre-processor one. Unfortunately this is not true of items in the Standard C Library.

Many Standard C Library functions can be implemented as macros, though the non-macro version must be available. For example abs() (in stdlib.h) can be provided as a macro as well as by a function. If you need to insist on the function form you must place the identifier in brackets; abs(x); can be provided by a macro, but (abs)(x); must be the function form. Placing the identifier in brackets inhibits the macro replacement.

A few pure macros break the all uppercase rule. L_tmpnam (in stdio.h) and offsetof() (in stddef.h) are both pure macros. Unfortunately many C programmers know only a small fraction of the Standard C Library. Very few C++ programmers know that library in its entirety, so apparently safe names in non-global scopes can be wrecked by the pre-processor.
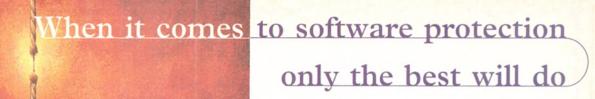
The remove() function/macro that I intended to use last time is a good example. Its unpleasantness, and that of others including rename() and floor(), is that it is a complete English word, attractive to use as a function name and break only if a version of the relevant header file that implements macro versions is included.

How many answers?

Look at the following short piece of code and decide how many different answers might result by using different conforming C compilers.

```
#include <studio.h>
int main () {
    char c = 100;
    int ans;
    c <<= 1;
    ans = ~c;
    ans >>= 1;
    printf ("%i", ans);
    return 0;
}
```

The C++ and JAVA Worlds go Visual

JFactory is the **Visual Builder for JAVA**. It enables developers to quickly create Java applications by dragging-and-dropping typical controls such as buttons, list boxes and menus. JFactory provides a single design environment to manage all aspects of user-interface development and includes a Project Manager, design windows, property sheets, a palette of drag-and-drop controls as well as providing the ability to test an application's interface, generate code, compile, and run the application.

ReportClasses 4.0

A New **C++ Class Library** from ESKA Software, ReportClasses speeds up the creation and maintenance of reports for C++ developers. It divides the two tasks of sourcing the report data and designing the layout of the report so that developers and users can each focus on their own part of the report. The **ReportData-Editor** allows developers to produce multiple views of the data source(s), from which the user can then create their own layouts with the **easy-to-use Report Designer**. Both the report structure and layout are implemented as C++ objects providing portability among data sources, GUIs, compilers and operating systems.

zApp

The **Visual Builder for Cross-Platform** C++ development. zApp Developer's suite is a toolkit combining the zApp Application Framework, Interface Pack, Factory, and Help system to provide single-source cross-platform portability. Visual programming with zApp makes it possible to quickly build and test different designes and then generate fully commented C++ code.

Stingray

"We add class to MFC!" ™

New versions of Stingray Software's MFC extensions C++ class libraries bring more enhancements to MFC. **SEC++ 1.1** increases library to over 40 classes and **Objective Grid 1.1** adds DAO and UNICODE/MBCS support.

Hypersoft Europe

Northern Office: **01159 - 376550**
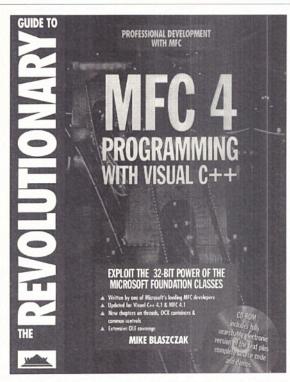Grove House, 13 Main Street, Keyworth, NG12 5AA

Southern Office: **01273 - 834555**
PO Box 901, Hassocks, West Sussex, BN6 9ZS

email: info@hypersoft.co.uk

CALL US NOW FOR INFO & DEMOS

All product names referenced herein are trademarks of their respective companies

✎ CIRCLE NO. 732

GUIDE TO

PROFESSIONAL DEVELOPMENT WITH MFC

THE REVOLUTIONARY

MFC 4 PROGRAMMING WITH VISUAL C++

EXPLOIT THE 32-BIT POWER OF THE MICROSOFT FOUNDATION CLASSES

- Written by one of Microsoft's leading MFC developers
- Updated for Visual C++ 4.1 & MFC 4.1
- New chapters on threads, OCX containers & common controls
- Extensive OLE coverage

CD-ROM includes fully searchable electronic version of the text plus complete source code and demos

MIKE BLASZCZAK

| ISBN 1-874416-923 | Price £46.99 |

WROX **Wrox Press**
http://www.wrox.com/

The Revolutionary Guide to MFC 4 Programming with Visual C++

For professional developers with a desire to get under the covers of the Microsoft Foundation Classes.

▲ Written by one of Microsoft's leading MFC developers

▲ Based on the best selling first edition, this book is updated and expanded to cover the new features of Visual C++ and MFC 4

▲ New chapters on threads, writing OLE control containers and common controls

▲ Covers Windows NT 3.51 and Windows 95

▲ CD-ROM includes fully searchable electronic version of the text, all source code from the book and more...

What's special about this book?

As a member of the MFC development team, Mike Blaszczak has an insight into the workings of MFC that is unavailable from any other source. Littered with descriptions of undocumented features of MFC, the book provides a unique perspective of the underlying structure of the class library.

Available from all good bookstores and computer superstores or by mail order from Computer Manuals: 0121 706 6000

✎ CIRCLE NO. 723

The convoluted processes of authoring help files with RTF in Word or a specialised help creation package may soon become archaic. **Neil Hewitt** looks at a tool which lets you write the help as you go along.

HELP!

Death and taxes are universal certainties, but for the Windows developer another inevitable reality looms: writing help files. Even those privileged software authors lucky enough to work in a large development team, one where writing the documentation is someone else's job, will probably have to tackle the task sooner or later.

Until recently the most common tool for this bit of business was Microsoft Word 2.0. It earned the distinction mainly because it was the only common software that could handle Rich Text Format (RTF) documents, which the Microsoft help compiler requires to do its stuff. With it, creating help files was a long and convoluted process of typing the help text, defining topics, and linking them together. Much messing about with footnotes and hidden text was necessary. Adding to the immense amount of effort involved in writing a help file 'the hard way' the RTF method was staggeringly inefficient. It was so difficult to see what the final help file would look like that frustrated programmers ended up changing what they'd done at least ten times. Consequently no prizes will be awarded to those who predicted specialist help authoring tools would emerge. A whole slew of visual tools is now available (see *Help authoring tools, EXE* December '95), all of which should, in theory, make writing help a doddle.

The process remains essentially separate from development process. It is most common to write the application first, then the help, and then, if necessary, go back into the application to add context-sensitive help

code. Keylink Computers wants to change that. The company's new help authoring tool, imaginatively named Helping Hands, works with Visual Basic 3 & 4 or Delphi 1 & 2 (separate versions of the tool are available for each platform). With it the development and help authoring processes can be integrated into a single stage. Keylink's decision to stop at the two most popular RAD environments is sensible. It might be possible to develop Helping Hands for C++, but more often than not the RAD developer writes help files, because the C++ developer has passed off the task to the technical writer.

A real paper manual!

Helping Hands works from a proprietary format of script which neatly encapsulates the topics and hyperlinking functions of the Windows help system. The script is a plain text file containing the full body of the help tagged with markers for jumps and topic boundaries. What makes Helping Hands different from several other script-based help authoring tools is that the help script is stored, as comments, within the source code of the VB or Delphi project itself. The Helping Hands software scans through the VB or Delphi source files, reads the help script, and writes it out to an RTF file which can be understood by the Microsoft Help Compiler included in the package.

The program itself is relatively compact, and comes on two high-density floppies packaged into a substantial plastic binder which also holds the documentation. Keylink, like Borland, is one of the few remaining companies that believes supplying full printed documentation with devel-

oper products is essential. The 115 page manual is comprehensive, and includes a full index and some useful appendices. It kicks off with a 'quick start' section for the appropriate RAD environment, then walks the reader stage by stage through the creation of a simple help file for one of the sample programs provided in each development package – VB's 'calc' applet and Delphi's 'imageview'. To be productive developers must be familiar with Helping Hands' proprietary scripting language. The quick start sections are comprehensive enough to achieve this efficiently.

Helping Hands is a 16-bit program, so it doesn't understand long file names, which could be a problem if you develop exclusively in Delphi 2.0 or Visual Basic 4.0. The package does not write to any of your project files. It leaves the naming scheme intact, but the final name of the .HLP file may come out mangled! I hope the next version of the program will be a 32-bit implementation. The interface is relatively simple, consisting of toolbar buttons and associated dialogs (see Figure 1). Absent are menus and keyboard accelerators. I suppose the idea

REVIEWS

Figure 1 – The rather tiny Helping Hands control panel.

is that the control panel can reside unobtrusively on-screen with the rest of the VB or Delphi windows. It is a shame that Keylink did not go one step further and integrate its toolbar into VB or Delphi's own.

Getting off the blocks

The help development process begins by specifying the text of the contents page. Getting this part right is quite important, since contents is the first page users will see when they open the help file (unless context-sensitive help is supported). Within VB projects the contents page text must be located in the `Form_GotFocus` handler of the main form, while in Delphi the contents must be placed within the `FormActivate` handler. This is the only restriction imposed on the placement of help text. All of it, including sub-topics, can be located within these handlers, or sub-topics can lie in random locations within the project. The whole point of this method of help file generation is that help text should be placed within the body of the function to which it refers. Since professional developers tend to comment their source code as they write it, the theory goes (and it is indeed a good habit, even for the solo programmer), then it is a small step to actually add end-user documentation at the same time.

Scripts can be written entirely by hand, but to do so would be missing the point of the program. A number of tools are provided which, taken together, can generate the majority of the control tags. To make use of the Helping Hands software at design-time the program should be running alongside Delphi or VB and told to open the same project which is open in the IDE. It would be nice if Helping Hands was sufficiently integrated with VB and Delphi to automatically work on the same project without needing this instruction, but unless you are changing projects every few minutes it doesn't add up to much more work. Once it is synchronised with the project under development Helping Hands can be used to add new tags into the help text or to compile and view the help file under construction.

As mentioned above, the help text is stored within the VB or Delphi source as comments. In VB, comments are denoted by

starting a line with a single quote, ', while in Delphi they are located within pairs of curly brackets: { } This is very confusing for the C++ programmer! The Delphi version has obviously been developed directly from the VB edition: to add scripting commands to Delphi projects still requires prefixing all the lines with a single quote.

Jumping around

The program has a tool which adds topics or jumps into the help. In each case the procedure is similar. Supply a name for the topic or jump, and the program generates a context ID and references automatically. The context ID can be overridden with a custom ID if preferred. The appropriate script text is then generated and copied to the clipboard, where it can be pasted into the function source code. Jumps can be to other pages within the help file, or to pop-up boxes which appear on top of the current page. The basic script text is the same for both. Again, I'd prefer Helping Hands to handle the pasting procedure automatically, rather than leaving me to manually select the function I'm interested in, place the cursor, then

paste the text by hand. Bitmaps can also be inserted into the help text with this tool. To turn bitmaps into jump points, however, the supplied hot-spot editor must be used. More on this below.

While writing a small demo help file I found a number of peculiarities in Helping Hands. Because neither VB nor Delphi wrap the source code text to fit the on-screen window it was possible to carry on typing on the same line continually. It is senseless to have the help text unreadable within the standard editing window, but even when a line break is added at the right edge of the Window, Helping Hands ignores the break and stitches the text together. To add a line break to the finished text a blank line must be inserted (and if you want to add a blank line you have to add two blank lines, and now my head is hurting...). Because IDE and Helping Hands work on the same project concurrently amendments must be re-saved before carrying out any function in

Helping Hands – or an error message appears. Finally, while Helping Hands implements ToolTips (otherwise known as bubble help) for the buttons on its toolbar, it does not employ the standard implementation. A bubble pops up the instant the cursor impinges on a button, instead of leaving a decent pause. It doesn't affect functionality
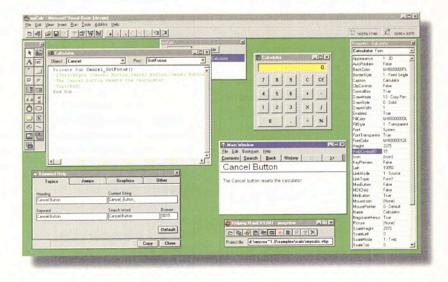

Figure 2 – Implementing context-sensitive help in Visual Basic 4.0.

RHINO TRAINING & DEVELOPMENT
Expert Training by
Expert Programmers

Visual Basic 4.0

Introduction To Visual Basic 4.0

Aimed at programmers new to the language who now want to build Windows based applications in Visual Basic 4.0

Course Content
Building Applications
Language Elements
The Visual Basic Tool Box
Creating a Visual Interface
Managing Disk Files
Understanding the Windows API
Introduction to OLE Server and Classes
Introduction to Microsoft Office Automation
Introduction to Database Development

2 Days	£500

Power Programming with Visual Basic 4.0

This course, for existing Visual Basic Developers, demonstrates the design of powerful applications that take Visual Basic to it's limits.

Course Content
Windows 32-bit Architecture
Processes and Multi-threading
Visual Basic 4.0 and the 32-bit API
Creating DDLs in Visual Basic and C
Developing OLE Servers and Classes
Real Time Programming
Introduction to OCX controls.

3 Days	£650

Advanced Database Development with Visual Basic 4.0 Enterprise

For existing Visual Basic developers, this course will show delegates how to create Visual Basic 4.0 Enterprise Client/Server database applications under Windows NT Server with SQL Server 6.0.

Course Content
Introduction to Visual Basic Enterprise
Client/Server Architecture
The Data Control and Data Access Objects
Accessing Remote Databases using ODBC API
and Remote Data Objects
Visual Basic for SQL Server
SQL Server 6.0

3 Days	£650

Visual Basic 4.0 Intensive

Get up to speed quickly with Visual Basic 4.0 on an Intensive five day course consisting of the Introduction to Visual Basic 4.0 followed by Advanced Database Development.

5 Days	£995

Tailored Training

We can design a training course to help you with a specific application, just let us know what you want to achieve and we will help you acquire the skills and knowledge to do it.

All our course leaders are experienced programmers. Classes are restricted to six delegates and are designed to give practical development experience through the intensive use of Hands on Exercises. All delegates receive a disk containing full source code for the course exercises and comprehensive course material
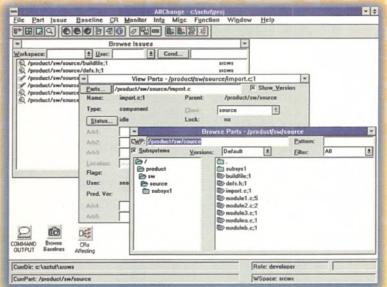
Microsoft®
SOLUTION PROVIDER
Authorised by Microsoft

Visual Basic, Access, Delphi, CA-Clipper, CA-Visual Objects, C, C++, Visual C++, Java, Internet & Intranet, Windows NT

For full course information call Rhino on 01454 417057 or 01302 810000

Figure 3 – The Help Compiler in action.

at all, but after a while I found it incredibly annoying. I later discovered, much to my relief, that bubble help can be turned off.

Implementing context-sensitive help is very easy. Each topic is allocated a context ID automatically by the software. The Help Context property of the related control in VB or Delphi is simply set to match (see Figure 2). It is possible to generate a text file or RTF document containing the text of the help file, and in the case of RTF, the appropriate header and footer information carrying details of the topics and jumps in use.

Graphics support is limited. A graphic can be embedded into a help file, which, if required, can contain copies of the image at different resolutions and colour depths. With the package's Graphics Editor it is possible to use one graphic image to jump to several different points in the help, but unfortunately the tool is only a hot-spot editor and can't be used to create or edit graphics. However, since graphic files for inclusion in help projects have to be in Windows bitmap (.BMP) format any old editor will do, even Windows Paint.

Putting it all together
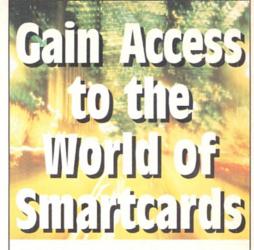
Once help topics are written, the contents page added, and the jumps and pop-ups described, Helping Hands will compile the help file. This is a two-stage process. First the VB or Delphi source is scanned by the pre-compiler, which checks for omissions or errors and alerts the user to them (a pre-compiler can be invoked at any time to check work to date, if required). Once slip-ups are corrected the program writes to disk an RTF file containing the newly-minted help. At last Microsoft Help Compiler is invoked in a DOS box (see Figure 3). The end result is a Windows .HLP file which can be read in the normal Windows Help application. Helping Hands includes a test button to try it out.

After all that I asked myself if the process is any easier than using any of the other third-party tools for help authoring, and concluded that it is and it isn't. It's certainly easier to write the help text itself in bits and pieces than to sit down at the end of the development process and write the whole lot in one long marathon (which tends to make for less readable and accurate text anyway). I would prefer a much more visual editing style, in which I could clearly see the layout and appearance of the help text as it would appear in the finished product, but realistically this is impossible without significant patching of the VB or Delphi code itself.

I can imagine that a solo developer or small team working on a medium-to-large sized VB or Delphi project would find Helping Hands a useful tool indeed – especially as an alternative to writing help in Word. Add its good paper documentation and on-line help (written, naturally, with Helping Hands), and you have a rather unusual but certainly interesting help file authoring tool. The price is a little high for a small piece of software, but like all truly useful tools you will pay for the work it saves you rather than for the code itself. Unless you really enjoy writing RTF by hand I'd suggest that Helping Hands is worth a look. ∎

Helping Hands for Visual Basic 3 & 4 and Delphi 1 & 2 are both priced at £149.95, and are available direct from Keylink Software on 01926 850909.

Building Internet Applications with C++ reviewed by **Colin Smith**

If you've only just come to appreciate the development opportunities of the Internet (à la Microsoft) you probably need a rapid way to catch up. This book is a good starting point. It shows how to implement basic FINGER, WHOIS, FTP, IRC, GOPHER, SMTP, POP3, and NNTP clients, and an HTTP server application with Visual C++, which in essence means MFC.

The authors explain the steps for creating applications using AppWizard and AppStudio, filling in the message map, fleshing out the member functions, and more. However, the broad scope of the volume means only very basic applications are covered. For example, one of the battles application designers have with FTP is directory parsing. A wide range of host types, from NT to VMS, must be supported. But this book's FTP application only parses UNIX flavoured directory listings. Its email and news applications provide minimal checking – a lexical analyser would have to be added so email addresses would be handled properly to conform to the rules in the address format RFCs. BIAC++ conspicuously avoids implementing TELNET, perhaps because of its copious options

and need for a terminal emulation such as ANSI, which would detract from the main focus.

The code derives from MFC's `CSocket` class, which appeared in Visual C++ 2.1 and VC 1.52 onwards. A scratch-built `Socket` class with an identical interface to the derived `CSocket` is included, in case readers are running an earlier version of the VC++ compiler.

QSend – unique to the book – is a kind of poor man's Network OLE. It allows OLE objects to be transferred between sites. Both the QSend client and server provide OLE container support. OLE objects are pasted into the client then sent to the server.

One of the best aspects of BIAC++ is the companion CD, which is presented in HTML. It lists the relevant RFCs and FYIs, and includes some sample Windows sockets and Internet applications including HTML editors and diallers. It also features an interesting collection of OCXs which implement FTP, MAIL, NEWS, and other functions, almost negating the code in the book.

To get the most out of BIAC++ you really need to know MFC beforehand, and in order to do something you should already have Windows Sockets successfully configured. Unless your MFC is really shaky you'll probably skip the chapter that outlines MFC's AppWizard, the

Document/View Paradigm, ClassWizard, and other MFC features.

From a practical viewpoint you have to decide how your rendition of FINGER, WHOIS or another Internet application is going to be better than the numerous existing versions. Realistically, to produce commercial quality versions of these applications you will need to incorporate a lot more features, employ a more sophisticated interface, develop better error checking, and delve into the rather dry RFCs – which this book manages to avoid.

At minimum you'll get some experience of using the Windows sockets API, which is hard to escape if you are in the Windows Internet applications arena, and a working knowledge of the popular Internet protocols. These are good items to put on your CV.

✓ **Verdict: Recommended for MFC developers**

Title:	*Building Internet Applications with C++*
Authors:	*Kate Gregory, Paul Robichaux, Brady Merkel, Markus Pope*
Publisher:	*Que*
ISBN:	*0-7897-0213-4*
Price:	*£46.99*
Pages:	*459 plus CD-ROM*

Creating CGI Applications with Perl reviewed by **Neil Hewitt**

This is a 'mix and match' manual. *Creating CGI Applications with Perl* doesn't clearly explain any specific subject, but instead has a go at several. At 299 pages it is not a particularly weighty volume, but an awful lot of information is packed onto those pages. On the other hand, this book is not a Perl programming manual. Instead of providing instruction in Larry Wall's unique scripting language from first principles, the authors point the reader to a number of on-line Perl resources and some good books on the subject.

Concepts such as the World Wide Web, HTTP protocol, and Common Gateway Interface are broadly defined. Code snippets are included frequently in each chapter, and serve well to illustrate the point being made. However, the authors have often left their explanations of the principles behind the HTML and Perl code to the point where it appears, which means a first read through can be quite con-

fusing. Perl topics are covered in an illogical order, which compounds the problem, but when these chapters are re-read both languages become clear.

The remainder of the book is given over to sample applications, each one demonstrating a particular principle. FormMail posts data from an HTML form via e-mail. Demonstrations of Server Push and Client Pull build dynamic Web pages. A chapter covers the Oracle Gateway (ORG), a CGI-based interface to Oracle databases, and Millicent, a CGI protocol for inexpensive transactions over the Internet. A number of other potentially useful scripts and pages are included.

There's much to be said for this 'learn by example' kind of manual, and the dedicated reader, or one who enjoys learning this way (including me), can pick up a working knowledge of Perl without having to refer to other sources. By the end the reader's understanding of the mechanisms behind CGI should be more than adequate – ultimately I found this a better book for learning about CGI than about Perl.

Given the huge amount of sample code included and the necessity for readers to have a Perl development kit before starting work on the examples, I would have welcomed the inclusion of a CD or floppy disk with *Creating CGI Applications with Perl*. It isn't fun to type in all the source, and might even be cumbersome enough to put less dedicated developers off the book. I'd suggest taking a look at it anyway, no matter how industrious you are, because it's one of those manuals which has enough useful information to make it a worthwhile purchase, if not an indispensable programmers' reference.

✓ **Verdict: recommended, but peruse before you buy**

Title:	*Creating CGI Applications with Perl*
Authors:	*John Deep and Peter Holfelder*
Publisher:	*John Wiley & Sons*
ISBN:	*0-417-14158-5*
Price:	*£22.50*
Pages:	*299*
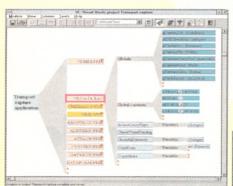
Maintaining, porting or updating applications?
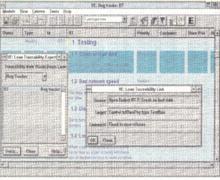## VB support where you need it most

VISUAL
EXPERT
Developer Edition

"The features you need to manage large VB projects."
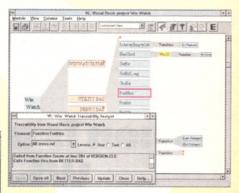Neil Hewitt, EXE magazine, April 1996

Visual Basic Analyst

- Analyze existing VB projects
- View VB code structure graphically
- View changes
- Add user-defined attributes
- Create reports
- Record schedule, status and test information, requirements & specs

Traceability Expert

- Create links between programs
- Show both internal and external dependencies
- Perform impact analysis Understand co-operative applications

Bug Tracker

- Link directly into the VB code
- Record defects, incidents and suggestions
- Powerful but simple to use Maintain structured logs with full status information

We can make your life easier!
Distributed by:

QBS Software Ltd
11 Barley Mow Passage
London, W4 4PH

Grey Matter Ltd
Prigg Meadow
Ashburton
Devon, TQ13 7DF

MRC Associates
Alpha House
High Street
Crowthorne
Berks, RG11 7AD
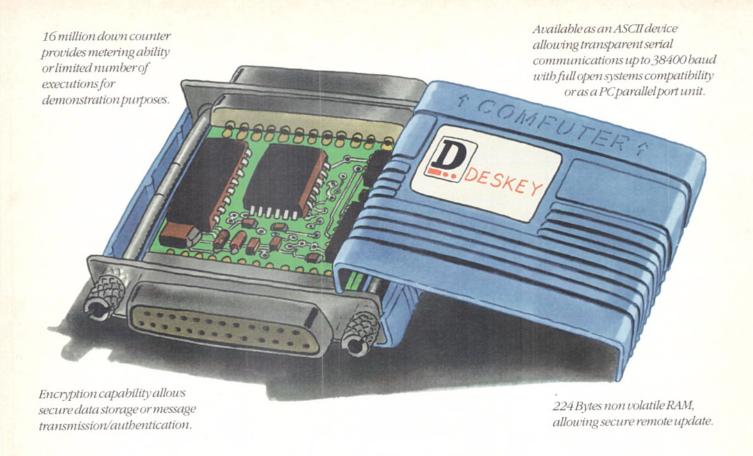
System Science
1-6 Bradley's Close
White Lion Street
London, N1 9PN

Tel:+44 (0)181 956 8000
Fax:+44 (0)181 956 8010

Tel:+44 (0)1364 654100
Fax:+44 (0)1364 654200

Tel: +44 (0)1344 772988
Fax: +44 (0) 1344 778916

Tel: +44 (0)171 833 1022
Fax: +44 (0) 171 837 6411

CIRCLE NO. 727

16 million down counter provides metering ability or limited number of executions for demonstration purposes.

Available as an ASCII device allowing transparent serial communications up to 38400 baud with full open systems compatibility or as a PC parallel port unit.

Encryption capability allows secure data storage or message transmission/authentication.

224 Bytes non volatile RAM, allowing secure remote update.

When is a dongle not a dongle? When it's a DESkey.

Parallel/Serial

PCMCIA

Mac ADB

Data Encryption Systems Limited
Silver Street House, Silver Street, Taunton, Somerset. TA1 3DL

Telephone 01823 352357
Fax 01823 352358
BBS 01823 352259
deskey@silver.cityscape.co.uk

There's never been a dongle like DESkey.

In the past the reputation of the hardware key may have been tarnished by inexperienced manufacturers bringing unreliable and troublesome products to the market. Data Encryption Systems DESkey range is the product of more than 20 years experience in the design and manufacture of security modules.

DESkeys are a totally new generation of product. Infallible in use and, when used in conjunction with DESlock, the most easy to implement security package available.

And unlike any dongle you might have used before you won't even know it's there.

Transparent in use, DESkey plugs into any parallel/serial port, or internal slot. This allows the developer to configure, meter or totally secure software against piracy or mis-use using the most flexible and secure hardware key available.

There are a wide range available with an extensive array of features to meet the requirements of different users:

● DK12 – our most popular high-speed low-cost dongle which, used alongside DESlock, is childsplay to implement.

● DK2 – immensely secure and versatile, among other things DK2 features secure remote field programmability.

● DK8 – our Mac DESkey

● DK37 – PCMCIA device.

● DK96 – everything you need for total software protection, created specifically for the demands of Open Systems.

All DESkeys are based around the very latest Application Specific Integrated Circuit and microprocessor-based technology and can be tailored to meet your exact requirements. We can even supply you with customised packaging and badge engineering.

DESkey and DESlock are just part of a range of unrivalled security products from Data Encryption Systems, the UK's leading software protection specialists.

Member of
FAST
Promoting the legal use of software

CIRCLE NO. 728

Visual Expert

Visual Basic delivers functional code at the pace required to keep up with changing business needs, but at a price. Visual Basic Programs can't always be modified as rapidly as they were created – stories of code written by missing contractors abound – causing untold misery when maintaining and updating code.

There is, however, a solution: Visual Expert Developer Edition. Visual Expert understands legacy code and simultaneously helps tackle bugs. In short it adds structure to Rapid Applications Development, improving quality, productivity, performance and efficiency.

Visual Expert consists of three elements: VB Analyst shows all the projects components in their true hierarchy, not necessarily apparent with VB code. A unique graphical display of the Visual Basic code structure creates a visual colour map of all the project's components – forms, controls, sub-routines, functions, variables, constants, declarations and conditional compilations – it couldn't be simpler. With one click of the mouse the graphical tree becomes a project engineering worksheet in which to record schedule, status, test information as well as requirements, specs, comments, questions and even cost of each element of the code.

Traceabiliity Expert enables Visual Expert to perform report generation based on queries. Need to know what should be tested this week, just ask with a couple of mouse clicks.

Traceability Expert also allows you to relate code in multiple projects, change one and you know what must be changed, or at least tested in the other.

And finally, Visual Expert's Bug Tracker which can track bugs, reports and feature requests. Powerful yet simple to use, the Bug Tracker is the only defect tracking system which links directly into the code – recording defects, incidents and suggestions in structured logs with full status information.

To purchase your copy of Visual Expert at the summer special price of £150 + VAT contact Philip Yorke on 0990 561246.

Manicured mice

Computer shippers who have attempted to jazz up their products by vividly colouring and decorating cases have had their problems. To date buyers have put their efforts into understanding terminology and equipment and have so far indicated that colour customisation is secondary to getting the spec right.

All that's changing. The work of COSMETIC PCs begins after purchase – when the operators are driven mad by bland design and cram/grey appearance of the equipment.

Businesses who affect a corporate image or have a long-established company livery like to make a point when their operations come into the public domain – travel agencies, banks, building societies and so on – that's great for their staff who find themselves with state-of-the-art kit that looks terrific.

The rest of us, forced by circumstance to work with a PC which has zero public visibility, have now got an opportunity to visually upgrade our kit and make it more readily reflect our lifestyles and moods. Tiger skin, motorcycle dayglo, cuddly teddy-bears, football club colours, haute couture fashion houses, nicknames, cannabis leaves – you name it and it can probably appear on your mouse's surface. Keyboards and monitors can look cool or red hot – anything but boring.

Special deals for EXE readers

Send your mouse and a logo or design and Cosmetic PCs will return it within a week looking clean and shiny for only £15.00 + £1.87 recorded delivery. Any mouse will do as long as the upper surface is clear to work on.

Cosmetic PCs can supply Microsoft V2 Intellipoint (not OEM) with adapter finished as you desire for £44.00. Call Cosmetic PCs Ltd on 0171 613 4339 for details.   Information supplied by the publishers

ADVERTISERS INDEX

Discounted books – exclusively to you

Prentice Hall are the suppliers of this month's selection of bargain books

PRENTICE HALL
A VIACOM COMPANY

Java Unleashed

by Michael Morrison et al
Normal Price: £46.95
Price to You: £37.57

SAVE £9.38!

Java Unleashed puts the Java programming language at your fingertips by giving you expert advice on programming fundamentals, embedding Java applets into your Web pages, programming games, multimedia, and animation. You'll learn about security VRML, interactive Web page creation, Javascript, content and protocol handlers, and documentation procedures.

Visual Basic Unleashed

by Scott, Shannon, Font, Hatfield et al
Normal Price: £41.50
Price to You: £33.20

SAVE £8.30!

Provides you with a comprehensive reference to virtually all the topics that are used in today's leading-edge Visual Basic

Applications. Covers latest version for Windows. Loaded with invaluable tips, hints, and shorcuts. Written by authors with real-world Visual Basic programming expertise, this book provides the most comprehensive coverage of this powerful language.

Database Developer's Guide with Delphi 2

by Henderson
Normal Price: £49.95
Price to You: £39.96

SAVE £9.99!

Learn the intricacies involved in developing robust database applications with Delphi. It shows you how to exploit the power behind the new features of Delphi 2 by programming your own database and using the sample client/server applications on the CD-ROM enclosed. Design database applications that can easily be scaled to an SQL server. Master Delphi 2's database component hierarchy, and discover the secrets of sound database applications and the relational theory behind that design.

Information supplied by the publisher

Selection	RRP	Your Price		Selection	RRP	Your Price	Month
Java Unleashed	£46.95	£37.57		Survival in the Software Jungle	£39.00	£29.00	Mar 96
Visual Basic 4 Unleashed	£41.50	£33.20		Distributed and Multi-Database Systems	£49.00	£37.00	Mar 96
Database Developer's Guide with Delphi 2	£49.95	£39.96		The Fax Modem Sourcebook	£24.95	£18.75	Feb 96
				Window Multi-DBMS Programming	£45.00	£33.75	Feb 96
				Object Orientation - Second Edition	£24.95	£18.75	Feb 96
				Internet World 60 Minute Guide to VRML	£18.99	£14.25	Jan 96

Descriptions of all books below can be found in *EXE Magazine*, June 95 to June 96

Selection	RRP	Your Price	Month	Selection	RRP	Your Price	Month
				World Wide Web SECRETS	£38.99	£29.25	Jan 96
TeachYourself Java in 21 Days	£37.94	£29.99	June 96	Internet World 60 Minute Guide to Java	£18.99	£14.25	Jan 96
Delphi 2 Unleashed	£54.95	£43.96	June 96	Windows 95 Secrets	£38.99	£29.25	Oct 95
Programming Windows 95 Unleashed	£46.95	£37.56	June 96	Power PC Programming for Intel Programmer	£48.99	£36.75	Oct 95
Schildt's Advanced Windows 95 Programming	£22.95	£18.00	May 96	201 Principles of Software Development	£20.95	£16.80	Sep 95
Essential Java	£26.95	£21.00	May 96	Oracle: The Complete Reference, Third Edition	£25.95	£20.80	Sep 95
Cross-Platform Programming for Windows	£26.95	£21.00	May 96	Teach Yourself Delphi in 21 Days	£23.00	£17.25	Aug 95
Foundations of Visual C++ Progr for Windows 95	£38.99	£29.95	April 96	Delphi Unleashed	£35.50	£26.65	Aug 95
Visual FoxPro Developer's Guide	£41.67	£31.25	April 96	Heavy Metal Visual C++ Programming	£37.99	£28.50	July 95
Delphi Programming for Dummies	£18.99	£14.25	April 96	5 OLE Wizardry	£28.95	£23.30	June 95
Unix Internetworking, Second Edition	£52.00	£39.00	Mar 96	C: The Complete Reference	£25.95	£20.80	June 95

BOOK ORDERS

Title	QTY	PRICE
Shipping at £3.50 per order		£3.50
TOTAL		

Name: _____

Address: _____

_____

_____ Post Code: _____

My Subscribers Club Number: _____

Payment Options

Cheques or purchase orders only.

❏ I enclose a cheque for _____
(payable to *EXE Magazine* and drawn on a UK bank)

❏ I enclose a company purchase order. Please send an invoice.

Simply fax to 0171 437 1350 (with purchase order) or post (with cheque) this form with your order to:

EXE Book Page, Centaur Communications Limited, Freepost 39 (WD 1414/29), St Giles House, 50 Poland Street, London W1E 6JZ

Please allow 28 days for delivery.

Soft*where?*

Software Designers – Wiltshire.
Superb packages including relocation.

Where can a Software Designer find unprecedented R&D investment, the most up-to-date technologies and practices, a committed team of professionals working together - and a superb lifestyle, living in one of the country's most outstanding areas of natural beauty?

At Motorola ECID, in Wiltshire, that's where.

As one of the leaders in the design and manufacture of digital cellular technology, we can offer the perfect climate in which to develop your career - designing the state-of-the-art base station technology of tomorrow and applying your skills throughout the project life-cycle.

With graduate-level qualifications and at least 2 years' experience of software development and design using C, as well as exposure to the full development life cycle, expertise in any of the following could bring you closer to the leading edge:

- **GSM** • **CASE tools** • **UNIX** • **SSADM**

We offer packages ranging between £20K - £30K, on top of excellent training, a global culture and outstanding career prospects in a growing, international company that is totally committed to investing in the best people and the best technology.

Please write with full CV to Pippa Holdom, Human Resources Department, Motorola ECID, 16 Euroway, Blagrove, Swindon, Wiltshire SN5 8YQ or fax on 01793 539024.

Please quote Ref: PH/NF/1182.

MOTOROLA
European Cellular
Infrastructure Division

Localisation Programmer/ Project Manager

To £20,000   **West Sussex**

Our Client, the UK office of a US media group, specialises in the development of software for the International market. Their strength in attracting strong strategic alliances coupled with their desire to remain at the leading edge has resulted in this requirement for a Localisation Programmer and Project Manager.

Working as a key member of the localisation team you will be required to manage the procedure, design, and development of localised titles from source materials through to finished product.

The successful candidate will be an established 'C' programmer with DOS and Windows programming experience. The ability to develop dynamically with advances in technology is essential, as is a knowledge of European languages and culture.

If you are finding your present role is limiting the use of your initiative, and are keen to join an organisation rewarding proactive contribution to their continued success, please telephone **Beverley Blanchard on 01932 830441, SMA Recruitment, Priory Chambers, 27 Church Street, Weybridge, Surrey, KT13 8DQ.**

SMA

SOFTWARE ENGINEERING APPOINTMENTS

REASON *international*
RECRUITMENT SPECIALISTS

Salaries from £20,000 to £28,000

If you are a recent graduate with a good honours degree or above looking for your first or second career move we are very keen to talk with you about some very exciting and rewarding appointments with several companies involved in key product and systems development projects. You must have solid programming skills using 'C', C++ or Visual Basic under Windows or UNIX combined with an ability and willingness to broaden your technical skills. Opportunities we can offer range from client server design and development to secure electronic payment applications across the Internet.

To discuss your career requirments, please send your full CV together with a covering letter to:

Rob Reason,
Reason International Limited,
North House 3, Bond Avenue,
Bletchley, Milton Keynes MK1 1JJ

Telephone: 01908 373694
Fax: 01908 373836
Email: reason@reasoint.demon.co.uk

ECM SELECTION

REAL-TIME EMBEDDED SYSTEMS

Graduate	Guru	Manager
£20,000 ~	£40,000 ~	£50,000+

First Class opportunities from fresh Graduate £20,000 to Manager £50,000+ to join our client, a high growth, world-beating Company – based rural SE England – to work at the forefront of Computer and Communication Technology.

If you have (or expect to achieve) an excellent qualification in Computer Science, Electronics or related topic and have experience in any of the following fields:

**Real-time Embedded Software
C & Assembler
Speech Recognition
Asynchronous Device Drivers
Video Compression
Ethernet - Token Ring - ATM
Network Protocols
IP Routing
Telephony Switching - PBX core
ISDN - DPNSS
Video Conferencing
ITU Q29xx Q93xx H32xx
Windows NT/95 or UNIX Kernel**

Call ECM now for further information on 01638 742244 quoting ref: 1272. Alternatively, send your CV by Fax: 01638 743066. Email: topjob@ecmsel.co.uk or Post to:

ECM SELECTION LTD

The Maltings, Burwell, Cambridge CB5 0HB
http://www.ecmsel.co.uk

EXE OnLine

It's not WHO you know

Seek out the training you need from a database of over 600 courses, supplied by an ever-increasing number of companies.

If you are a software developer, and you need training, then you need EXE OnLine.

The Software Training Guide

http://www.exe.co.uk

Ctrl Break

Chase my fears away

Chase Technology, in an attempt to cash in on the growing Year 2000 scare, has linked the widespread bug in date codes to the 16th century seer Nostradamus. The company claims the following passage from Nostradamus' writing clearly points to universal mainframe collapse:

The year 1999...
From the sky will come a great King of terror...
There'll be war, there'll be peace,
But everything that day will cease

However the PR punters at Chase edited the first naff-rhyming stanza of Nostradamus's passage for their Year 2000 scare mongering news, but Ctrl-Brk is pleased to provide the complete text:

The year 1999, seventh month,
From the sky will come a great King of Terror:
To bring back to life the great King of the Mongols,
Before and after Mars to reign by good luck.

Chase should call their IT department and have their system clocks put ahead five months, so their Year 2000 problem will coincide with everyone else's. Otherwise it looks like they face a frightening visit from Ghenghis Kahn, armed with an ice cream bar.

Anorak set, unite

In this age of encroaching political correctness, stereotypes about any identifiable group are becoming unacceptable. With this in mind Ctrl-Brk wishes to bring a blatant attack on our collective image to the attention of all readers of this venerable page.

Consumer Surveys of London recently sent us two copies of a news release which begins 'Internet users are a long way removed from the unwashed nerds in anoraks that they are often portrayed as [sic]'. The illustration reproduced here was included with this 'news'. As far as we can tell from a quick survey of *EXE* back issues, the launch edition of *Cyberia*, and the computer supplements of *The Telegraph* and *The Guardian*, Consumer Surveys is the only group that ever portrayed Internet users as spotty long-haired losers with dirty underwear.

We took exception to this cartoon-insult, finding it overly personal. If you think our breed has been slighted, you can reach Consumer Surveys at 0171 403 6885 or fax 403 4536 to tell them so. The spin doctors responsible for the picture are Charlton Communications, 0171 485 4552. Neither organisation provided email addresses. No spotty long-haired losers there, obviously.

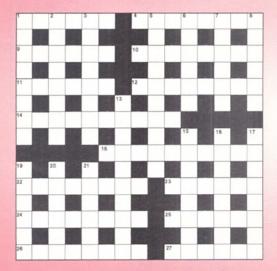## Brion and Betty
by Neil Kerber

Fancy a Java?

Those clever coffee-related Java product names keep coming thick and fast. First was the rather obvious Symantec Café (with the equally punny codename Espresso), then the more obscure Latté from Borland. In typical fashion, Microsoft's marketing department came up with Jakarta, a clear attempt to impose a location- rather than libation-based naming standard. The latest entry is IBM's Arabica, which supports the Beans initiative. What next? Cappucino for Windows? The GoldBlend Development Kit? Or how about Caffeine 1.0 – software so good, it'll keep you up all night!

Just half a cup for us, please.

PRIZE CROSSWORD

ACROSS
1. Comment to flag again (6)
4. Stopped batting and flagged labels (8)
9. Slippery and grey as scrambled eggs (6)
10. Side product of a node? (8)
11. So rest badly in main or backing (6)
12. Looking quickly like a fax machine (8)
14. Eg, nice sex I confuse with unusual events (10)
18. Standard testing tasks (10)
22. Getting on with giants in pattern recognition system (8)
23. Santa with one-way structure in sentence (6)
24. Tars mint something new to send (8)
25. Pressure of work, for instance (6)
26. Called in the Alps ... (8)
27. ... and changed the file on screen (6)

DOWN
1. One of 11 inside teacher's book (8)
2. Never forgotten label! (8)
3. Worked out the order of work (8)
5. Measure of success of program or programmer (10)
6. Failing to find ratio in another table (6)
7. Beginner who shouldn't crow (6)
8. dBase prompt with years - many years (6)
13. It takes so long to get the data! (6,4)
15. Acted like another system (8)
16. Dispute over a function's data (8)
17. Worked out 5 for instance (8)
19. A road you start with no interest (6)
20. Italian river country to the East (6)
21. C7H80 got from tar (6)

SOLUTION TO JUNE'S CROSSWORD

ACROSS:
1. WINDOW 4. CONVERTS 9. INSERT
10. LEARNING 12. ALLYING 13. TRIVIAL
14. LUNCH 15. NOSING 19. BINARY 21. SNAFU
24. PROCESS 26. EXTRACT 27. INFRARED
28. MEDIUM 29. ELEMENTS 30. USURER

DOWN:
1. WRITABLE 2. NESTLING 3. OARFISH 5. OVERT
6. VERSION 7. RAISIN 8. SAGELY 11. AGENDA
16. STYLES 17. CAVALIER 18. CUSTOMER
20. ITERATE 21. SETTERS 22. UPTIME
23. COFFEE 25. SHEET

Method in her madness

Will we need to buy CASE tools? Will we be sent on courses? Do I get to blow the dust off my flowchart stencil? Ms Stob explains the latest methodology fashions.

FRAD. Characterised by bodged code that does its job, Fearfully Rapid Application Development – no I don't care what you thought the 'F' stood for – works as follows: a competent programmer in a good mood grasps the problem and bashes in a bit of code as a first attempt. Somewhat to his surprise it compiles, and it even works with the test data set that he was thinking of as he wrote the code. When tested on real data, the program collapses – but no worries – a cup of black coffee, a few dodgy casts, and we have the thing on its feet again. A couple more real life tests and we have something we can call a beta. After that, we might buckle down and document the design.

FRAD is frowned upon by all major consultancies except one (which doesn't understand it). But it has one feature which distinguishes from all other methodologies – it's used.

BTB. The principle behind By The Book is that all programming problems have been solved before, there is no need to reinvent the wheel (this is BTB's key phrase, repeated by its supporters in plonking tones two or three hundred times a day), one has merely to look up the appropriate solution in the By The Book book. If the solution doesn't seem to be there, it's your fault for not understanding the problem in a BTB way.

BTB was first mooted by Fred Mainchance, a freelance wallpaper pattern traveller from Connecticut who had fallen on hard times. He initially proposed seven universal BTB solutions in a cheaply bound paperback costing $8.25 without California sales tax. The new two-volume hardback revised edition features 146 universal laws, an introduction by Boochy Grade and a trial version of Mainchance Software's BTB Project Kontroller System, and retails for $87.50, or £85 in the UK. Experienced methodologists will recognise that BTB obeys Ted Codd's Law of Multiplying Laws (aka the Reverse Occam Principle).
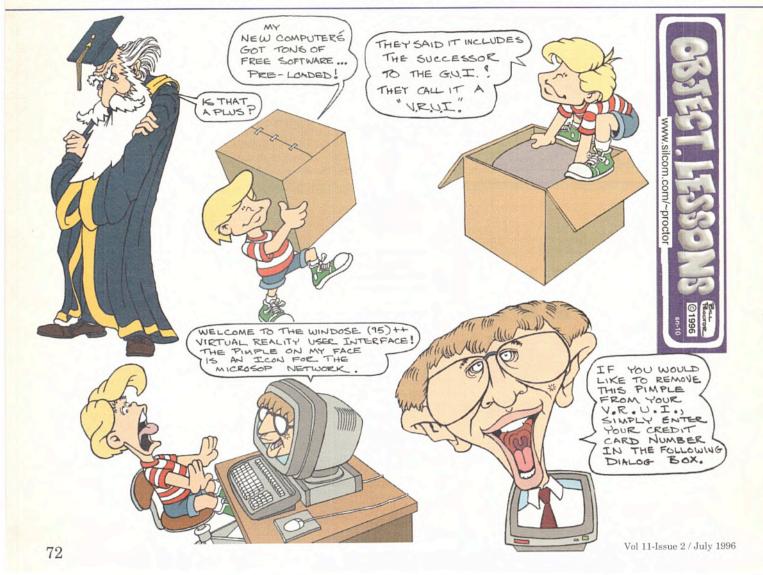
PMD. Pen is Mightier than the DataFlow. Refers to a style of programming where the whole justification for the application, in the mind of the application designer, is to exploit a whizzy new hardware or software feature which may or may not have any practical application in real life. The classic example of this, from which the methodology derives its name, is Pen computing, which is expensive and enjoyable to program, but is very nearly useless in the mouse-ridden real world.

Other examples of PMD are the rewriting of a simple calendar program so that it is 32-bit and multi-tasking, and everybody who in the 1980s worked out how to drive those pretty LED displays on the front of Dell PCs (hello Robert Schifreen wherever you are). It is a primary tenet of PMD that the client must be unaware that PMD is the design system being used. Many *EXE* articles describe good candidates for PMD projects.

IYF. The hottest, excitingest approach around. First the customer prioritises all the features in the project: those he must have, those that are important, and luxuries. The programmer then builds the application to strict deadlines, starting with the most important. Features which cannot be completed are discarded. The customer must review the software at fixed points in the project, and can alter the direction to meet his emerging needs. Of course, after week two he gets bored and doesn't do the review properly, with the consequence that the programmer is more or less left to his own devices until the anti-penultimate week, at which point the customer takes a look at what he is about to get and panics. Thereafter all the usual and well understood principles of Software Management cut in.

'IYF' stands for 'It's Your Fault'. ∎