



Taking control

Get a better grip on VB; read Tim Anderson's taster of two new products: Basic Constituents, custom controls with a difference, and dbComplete, to improve on VB's data-aware controls.

Visual Studio's Service Pack 1 has just arrived on my desk. You can download it from Microsoft's web site, although that is not recommended if you have a dial-up connection, due to its huge size. I was disappointed to find little in it for Visual Basic users. There is an updated data grid control and problems with remote data objects have been corrected. But the irritating bug, whereby changing the font on multiple selected objects crashes Visual Basic, remains.

The sad fact is that Visual Basic was more stable in version 3.0 days, even when running 5.0 on Windows NT. More extensive fixes have been provided for Visual C++ and Visual InterDev.

One thing amused me: the readme file warns you not to run Regclean, Microsoft's registry clean-up utility, or several things will stop working... Some of us have already discovered this.

There is better news for Visual Basic developers in the owners' area on Microsoft's web site. A range of new components has been provided, including a resource editor add-in (Fig 1), an Office 97-style button control, and a .DLL that lets you debug subclassed windows: without this add-on, subclassed windows crash Visual Basic in break mode.

Basic Constituents

Becubed Software is a recently formed company that has taken on several of the former Microhelp products like OLE Tools. Basic Constituents is a new set of custom controls with a difference: the controls are not intended to be used as-is but as parts in creating your own ActiveX controls. The key to creating ActiveX controls in Visual Basic is in deciding which properties, events and

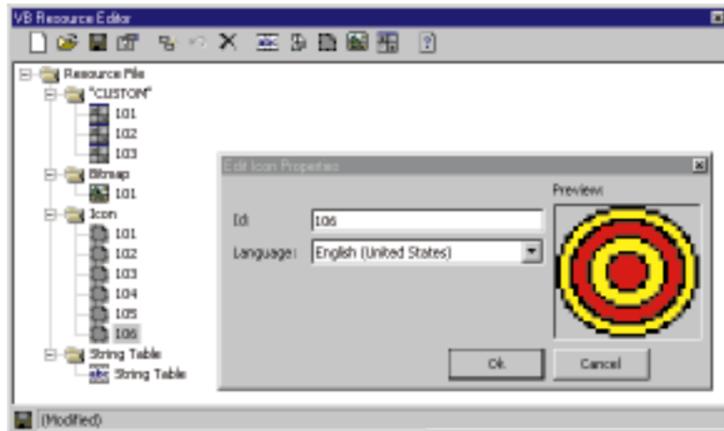
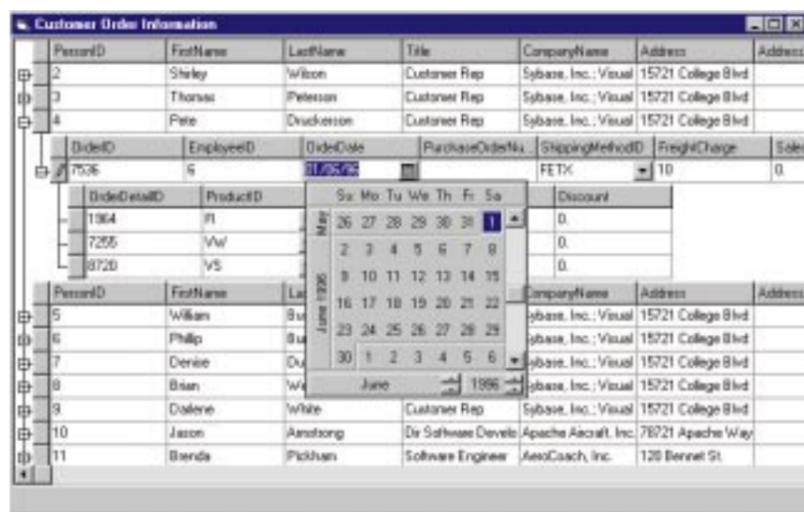
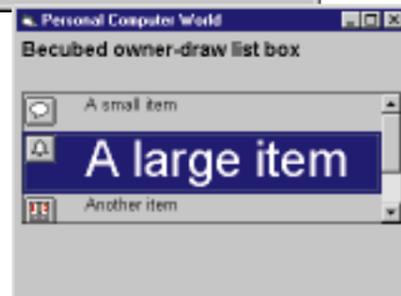


Fig 1 (above) This VB add-in resource editor is simple but useful. You can get it free from the Microsoft web site

Fig 2 (right) An example of the horrible things you can do with a Basic Constituents owner-draw list box

Fig 3 (below) The dbComplete outline data grid is striking but inelegant



Anderson answers...

Q I'm writing a card game, and want to drag the images around in the same way as Solitaire without resorting to the lower-quality card game of Free Cell where the cards are not dragged. I have two challenges: to animate the dragging of a bitmap (not just an icon or outline), and to combine several separate graphics to be dragged as one pile of cards.

Glyn Tomkin

A This sounds like the ideal project for learning Visual Basic. The first point of interest is that a handy DLL containing playing card bitmaps is installed with Windows 95 and NT. It is called CARDS.DLL and if you open it up in a resource editor like Developer Studio you can see that the card faces are helpfully identified by IDs from 1 to 52, while the remaining bitmaps provide a selection of card backs and other useful images.

Even if you have your own card images, I'd suggest that a resource file is the right place for them. There is a problem, though, with CARDS.DLL in that VB can't load resources from DLLs. If you have a resource editor and want an easy life, you can easily convert it to a .RES file, but that's not so good if you want to distribute your application to others who already have CARDS.DLL. A better option is to use API functions to access the DLL directly. See "Loading Resources from a DLL" for code to do this.

A picture box is essential for API work, as it's a proper window with its own handle and device context. But filling a form with picture boxes is a poor idea, as they're heavy on resources. The best plan is to keep one invisible picture box in which to load images, and then copy the bitmap to an image control by using

```
Image1.Picture = Picture1.Image
```

For a card game, you'll want to use a control array for the image controls to

simplify the code and let you create and destroy controls at runtime.

Getting full drag

With the image loaded, you can tackle the problem of full drag (Fig 4) as opposed to outline dragging. The secret is not to use the built-in drag routines as these appear to disable attempts to draw in the outline: write your own

drag routine using the MouseDown, -Up and -Move events of the image control. The idea is to begin a drag with the MouseDown event, move the image in the MouseMove event and end the drag with the MouseUp event. To do this, you need to have some way of telling when a drag is in progress. Also, you need to allow for the mouse-click landing anywhere on the image. You need to move the image relative to where the mouse was clicked.

Here is a suggestion which does both together. Controls in Visual Basic have a Tag property, which does nothing at all. It is there for your use in situations like this (a poor man's custom property). In the MouseDown event, put the line

```
Image1.Tag = Str(X) + Chr(9) + Str(Y)
```

This stores the X and Y co-ordinates, using the Tab character as a separator. In the MouseMove event, use the code in Listing 1 (p309). The code for MouseUp is exactly the same, except that it ends:

```
Image1.Tag = ""
```

If the Tag property is empty, you know the image is not currently being dragged.

Dragging a stack of cards

I suppose you could write extra code to drag a stack of cards. This is not the way to do it, though. If one card lands on

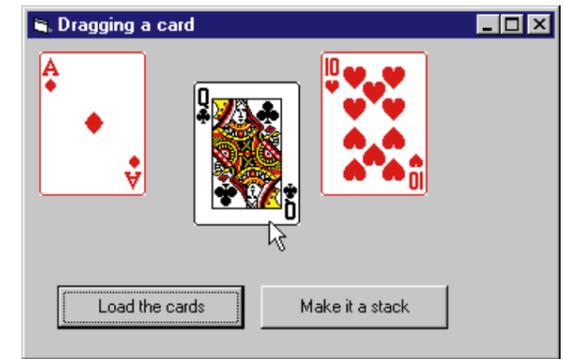


Fig 4 Dragging a card with the full image showing works fine, as long as you do not use the built-in Drag methods

another and you want to treat the two cards as a stack, it would be better to delete the lower image and draw a new image illustrating the top card with some artistic shading added, to show it is the top of a stack. You can do the drawing in an invisible picture control and then copy it to the image control. Listing 2 is a simple example and this code does the visual stuff, but of course you have a lot more to do in creating arrays to hold details of which cards are where, shuffling the pack, applying your game rules, and so on.

Loading resources from a DLL:

Visual Basic 5.0 can easily load resources from a .RES file but not from a DLL, so you need to use the API. The first step is to use the API viewer to add several declares to your project.

You need:

```

Declares:
LoadLibrary
FreeLibrary
LoadBitmap
SelectObject
DeleteObject
GetObject
CreateCompatibleDC
BitBlt
DeleteDC
    
```

(continued over)

methods to expose, and Becubed has a wizard to assist with this but unfortunately it is not yet available for review.

The meat of this package is in the controls themselves which are relatively lightweight, with some interesting features for the serious VB developer. An example is the list box constituent (Fig 2) that can be set to the owner-draw style, allowing you to

create virtually any kind of fancy list box.

The 22 other constituents include tree view, splitter, zip compression, banded toolbar, progress bar, tab strip and tray icon components. There is some overlap with standard VB controls; the difference is that these have more advanced customisation features. Be warned, however, that as with all advanced

Windows tools, you will need to learn about the Windows API to successfully use Basic Constituents.

dbComplete

Visual Components, part of Sybase, is responsible for some of the best ActiveX components around, including the Formula One spreadsheet control. Now added to

Listing 1 MouseMoves

```
Dim XOffset As Long, YOffset As Long
Dim TabPos As Integer

TabPos = InStr(1, Image1.Tag, Chr(9))

If TabPos <> 0 Then
XOffset = Val(Left(Image1.Tag, TabPos - 1))
YOffset = Val(Right(Image1.Tag, Len(Image1.Tag) - TabPos))
Image1.Left = Image1.Left + X - XOffset
Image1.Top = Image1.Top + Y - YOffset
End If
```

Listing 2 Invisible drawing

```
Dim mypic
Dim XOffset As Long, YOffset As Long
Picture2.Cls
Set mypic = Picture1.Image ' contains card bitmap
Picture2.Line (1, 1)-(Picture1.Width + 1, 1)
Picture2.Line (1, 1)-(1, Picture1.Height + 1)
Picture2.Line (30, 30)-(Picture1.Width + 30, 30)
Picture2.Line (30, 30)-(30, Picture1.Height + 30)
Picture2.PaintPicture mypic, 60, 60
```

Listing 3 Adapted declare

```
Declare Function LoadBitmap Lib "user32" Alias "LoadBitmapA"
(ByVal hInstance As Long, ByVal lResourceID As Long) As Long
```

Types:

BITMAP

Constants:

SRCCOPY

There are a couple of points to note. One is that the LoadBitmap declare needs to be adapted to work with a resource ID rather than a bitmap name. Listing 3 shows the adapted declare. Second, the GetObject declare clashes with the VB function of the same name. Replace it with GetObjectAPI shown in Listing 4 (p310).

All this is fairly horrible for someone

used to the normal, common-sense Visual Basic language but this is how Windows works under the surface, so it is just a matter of pretending you are working in C. It is code you only have to write once, and when the bitmap is loaded into the picture box, you can access it via the Image property.

If you want to know more about working with graphics at this

level, get hold of Dan Appleman's book, *Visual Basic 5.0 Programmers Guide to the Windows API* (see "PCW Contacts", p310).

QI have run into problems with Visual Basic 2.0's PictureBox controls. I need to print text into the control below a picture but I cannot get Visual Basic to do this. Is this possible to achieve, or can it be done with a later version of the software or a custom control?

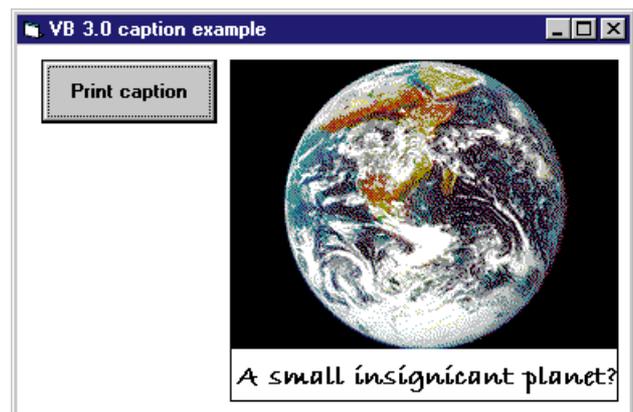
Edward Lea

AWith Visual Basic 4.0 and higher, this is easy. PictureBoxes have a PaintPicture method which enables you to position a graphic anywhere within the control. You can then use CurrentX and CurrentY properties to position text where you want, and the Print method to place it on the control.

In Visual Basic versions 2.0 and 3.0 there is no PaintPicture method and a picture always appears at top left. You can still print the caption to the right or below the image (Fig 5). Otherwise, it is a matter of exploring the API.

PictureBoxes are well suited to API graphics programming since, unlike image controls, they are fully-fledged window controls. (For an example, see the preceding question and answer.) ■

Fig 5 You can add a caption to a picture box, even in VB 3.0



the range is dbComplete, which aims to improve on both the native Visual Basic data-aware controls and rivals like Sheridan's Data Widgets. But stop reading now if you do not like ODBC, because dbComplete uses it exclusively, making it suitable mainly for client-server installations.

We tested a beta version of dbComplete. The product consists of nine

controls, most of which are data-aware gadgets like a drop-down picture combo and a calendar control. The two controls at the heart of dbComplete are an enhanced data control and a high-powered data grid. Key features of the data control are the ability to link one data control to another in a master-detail relationship, and the built-in support for transactions. There is a

CommitMode property, which can be set to commit changes automatically when leaving a record, or to commit changes only when the linked master record changes: you could edit order detail lines and not have the changes committed until you moved to a different order. Another feature is a built-in search dialog that lets you set one or more search conditions at runtime.

Listing 4 GetObjectAPI replacement

```

Declare Function GetObjectAPI Lib "gdi32" Alias "GetObjectA" (ByVal
hObject As Long, ByVal nCount As Long, lpObject As Any) As Long
Sub LoadResource (lCardID as long)
Dim LibHandle As Long
Dim BMHandle As Long
Dim lRetVal As Long
Dim Tempdc As Long
Dim Oldobject As Long
Dim bm As BITMAP

LibHandle = LoadLibrary("CARDS.DLL")

If LibHandle <> 0 Then
BMHandle = LoadBitmap(LibHandle, lCardID) ' 1 gets Ace of Clubs
  If BMHandle <> 0 Then
    Picture1.Cls

    lRetVal = GetObjectAPI(BMHandle, Len(bm), bm)

    ' set properties of picture box
    Picture1.BorderStyle = vbBSNone
    Picture1.AutoRedraw = True
    Picture1.Width = Picture1.ScaleX(bm.bmWidth, vbPixels, vbTwips)
    Picture1.Height = Picture1.ScaleY(bm.bmHeight, vbPixels, vbTwips)

    Tempdc = CreateCompatibleDC(Picture1.hdc)

    Oldobject = SelectObject(Tempdc, BMHandle)
    lRetVal = BitBlt(Picture1.hdc, 0, 0, bm.bmWidth, bm.bmHeight,
Tempdc, 0, 0, SRCCOPY)
    lRetVal = SelectObject(Tempdc, Oldobject)
    lRetVal = DeleteObject(BMHandle)
    lRetVal = DeleteDC(BMHandle)

    Picture1.Refresh
  End If

lRetVal = FreeLibrary(LibHandle)
End If
End Sub

```

The most distinctive part of dbComplete is the data grid (Fig 3), which combines the functionality of a grid with an outline control. The idea is that you link each level of the outline to a master-detail relationship in the database. You can also define groups which enable you to specify a set of records, such as all the contacts at a particular company. At runtime you navigate the data by expanding and collapsing sections of the grid.

The grid is an ActiveX container, so you can set it to display a field through a data-aware control. For example, you can use

the calendar control to display a date field. Like the other dbComplete controls, the grid exposes a full range of properties, methods and events for programmatic control. You can also select Expert mode, which means you write your own SQL rather than have the grid generate it for you.

The beta version of dbComplete which we tested was not in the least reliable. Assuming that Visual Components gets it working properly, it makes a powerful set of tools for creating instant database clients. The outline grid makes a capable but over-busy interface. As with all grid-based

Books in brief

■ Inside Visual C++

by David J Kruglinski

The title is misleading: this book is not packed with inside information, but it is a step-by-step guide to building applications using the Microsoft Foundation Classes and Visual C++. An introduction to COM programming covers ActiveX client applications and automation but not the creation of ActiveX controls, nor the Active Template Library. A new section covers internet programming, with skimpy coverage of MFC's Winsock and WinInet classes. There is a section on MFC's database classes but not the new OLE DB standard. As an introduction to MFC and its document/view architecture *Inside Visual C++* is excellent, but it is not a guide to the Visual C++ interface and tools.

■ Using Visual C++

by Kate Gregory

Microsoft Foundation Classes is the main focus of this book which is broad in scope and touches on topics like the Active Template Library and Microsoft Transaction Server, among others, and there is a section devoted to Developer Studio. Readers coming from other tools such as VB or Delphi will find this book more accessible than *Inside Visual C++* (above). The layout is good and there is a generous selection of Que's additional published titles on the accompanying CD (including *Using MFC and C++ By Example*). The book offers only brief coverage of database topics, though.

solutions, you need to take care not to dump much data into it in one query, or else performance suffers.

The instant master-detail and transaction features are nice, but of course these things can be achieved using VB's standard data control, or remote data control combined with appropriate code.

PCW Contacts

Tim Anderson welcomes your Visual Programming tips and queries. He can be contacted at the usual PCW address (p12) or at visual@pcw.vnu.co.uk.

VB Assist In case you missed it, you will find an evaluation version of VB Assist on our September issue CD (it was reviewed in the same issue).

Basic Constituents costs £125 (ex VAT) from Contemporary Software 01344 873434

dbComplete costs around £320 (ex VAT) from Visual Components 01892 834343

Books from Computer Manuals :

Inside Visual C++ 5.0, Microsoft Press £46.99

Using Visual C++ 5.0, Que £37.49

Visual Basic Programmer's Guide to the Win32 API, £54.95

Tel: 0121 706 6000



The **outlook's** changeable

Tim Anderson shifts from pillar to post this month. He looks at fixes in Outlook, Microsoft's decision to go against the Java grain, VB bugs, and the ever-changing Access MDB.

Java developers need to watch their step. At the TechEd conference in Nice, Greg DeMichillie, Microsoft's group program manager for RAD Tools, explained that the company won't support Remote Method Invocation (RMI). To the uninitiated, this may sound like a technical detail, but in fact this opens a chasm of incompatibility between Microsoft's plans for Java and the direction set down by Sun and other licensees.

RMI is a way of calling the methods of Java objects situated elsewhere on the network. Therefore it is a key element in distributed computing, networked applications in which code can be executed partly on a client machine, and partly on one or more other machines. Microsoft will not support RMI because it competes directly with Distributed COM (DCOM) which does the same thing but using the same technology as ActiveX.

DCOM is coming on nicely, especially with the recent release of Microsoft Transaction Server, which provides intelligent management of DCOM objects. RMI is an attractive technology, especially if you want to build applications that do not require Windows. But RMI needs changes at the Java Virtual Machine level to work, and Microsoft is not going to make those changes in its version of the JVM, despite this being the Reference Implementation for Win32, whatever that means.

This does not mean that RMI will not work on Windows. There are, and will be, other non-Microsoft implementations of the JVM that support RMI. You can be sure, though, that the Microsoft JVM, which ships with Internet Explorer, will be very widely installed on Windows machines. The problem for Java developers is how to



Bill Gates addresses Tech-Ed 97 by satellite. The most sensitive subject is Java

judge the sensible way to proceed. Nevertheless, Sun isn't likely to implement DCOM in its JVM implementations.

Standard issue

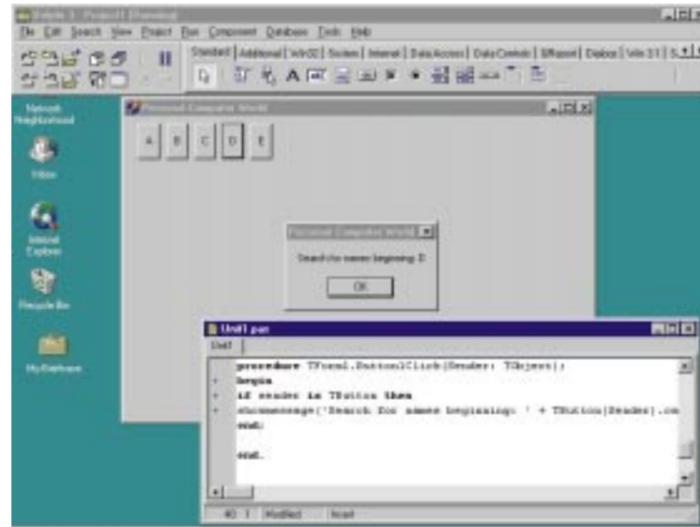
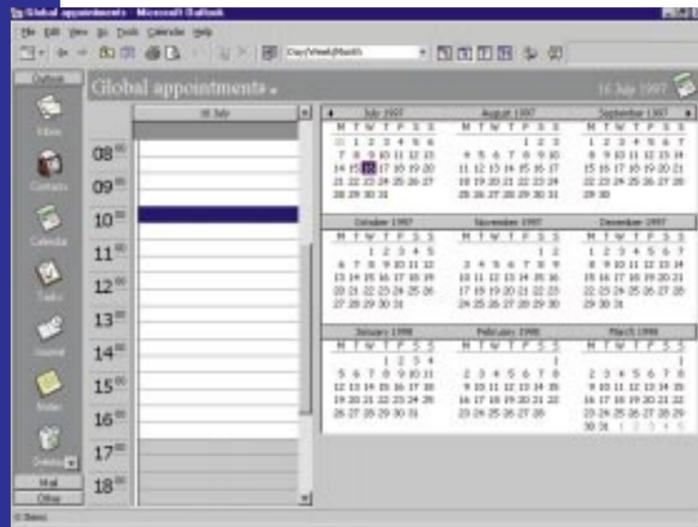
The problem is not confined to RMI. If you look at the Javasoft web pages (www.javasoft.com) you will see a host of APIs, standards and proposed standards for Java, including the Java Native Interface, JAR compression, multimedia APIs and many more. But the only way to find out what is and is not actually implemented on rival JVMs is to research each one individually. There is also the CORBA factor. CORBA is a comprehensive architecture for

distributed objects which, unlike RMI, is language-independent.

The original intention was that Java would use CORBA's IIOP (Internet Inter-ORB Protocol) which in theory makes RMI unnecessary. RMI is simpler and quicker though, and therefore more attractive to Java developers. The latest news is that Sun will work with CORBA's Object Management Group to add RMI support to IIOP. In practice, RMI seems likely to be popular, while CORBA is still more discussed than used within the industry. The truth is that CORBA needs Java more than Java needs CORBA.

As for Microsoft, I suspect the RMI will

p294 >



Above Heading for the web: Microsoft Outlook may go HTML

Above right The beginnings of a Delphi address book. This project uses runtime type identification and a typecast to allow many buttons to share OneClick procedure

be the issue that breaks the uneasy alliance between Bill Gates' company and the rest of the Java community. For obvious reasons, Microsoft is not promoting Java as a cross-platform solution and, speaking at Tech Ed, Greg DeMichillie was keen to point out its weaknesses. There are incompatibilities between versions of the Java Development Kit and between different JVM implementations, there are performance issues, there is the question of how Java can truly be both an open standard and a Sun technology, and there are compromises involved in any cross-platform solution.

All true, but these are problems of implementation rather than weaknesses in the Java concept itself. If there is sufficient will to succeed across the industry, and the signs are that there is, Java will be a viable proposition for cross-platform distributed applications.

Microsoft supports Java primarily as a language, and it will be giving Java code direct access to the Windows API to make it a better tool for creating Windows applications. According to Microsoft, the best vehicle for delivering cross-platform functionality is via HTML, which is becoming an increasingly powerful environment in its own right, rather than with Java. It is a fair point, and should continue to be a solution

whatever the outcome of the Java and distributed object wars.

Look out for the new Outlook

If you are developing solutions in Outlook, you might want to stop and wait for the next version. Microsoft will fix some of the most gaping holes in its developer features. Examples are that it is currently impossible to declare an automation object that is global to an Outlook application, which means new instances have to be created each time a form is opened.

Another vital new feature will be folder events, so you will be able to trigger code when a folder is opened. Sadly, VBA for Outlook is not on the agenda. The reason seems to be that Microsoft is ambitious for Outlook applications to run in many environments. VB Script is designed for this and so it will remain Outlook's scripting language.

Looking further into the future, there is talk of an HTML version of Outlook. In a sense, this would be playing catch-up with

design time and at runtime, component arrays. Delphi has not, but a book called 'Delphi Nuts and Bolts' which my son gave me has a useful tip as to how to create them during runtime. Unfortunately, it does not go far enough, since I have not yet found out how to deal with the array's event handlers, such as 'OnClick'. I went as far as to create a class TArrayButton from TButton and give it both an 'Index' property and an event handler returning the 'Index'. But I have failed to make this work."

Control arrays are invaluable in Visual Basic, partly because they provide a way of creating controls at runtime, which is otherwise impossible. They also make it easier to code in situations where many buttons have a similar function. For example, you might have an address application with a button for each letter of the alphabet. Clicking a letter finds matching names. All those buttons can use exactly the same code for the click event.

In Delphi, control arrays are less important. You can easily create controls at

Fig 1 Declaring a procedure in a Delphi form's class definition

```
TForm1 = class(TForm)
    procedure MyButtonClick(sender: TObject);
end;
```

Netscape's Communicator, which is already a web-based solution. It is an attractive idea though, and would presumably let you use Outlook from any browser with web access. Do not expect it soon, though.

Control Arrays in Delphi

Philip Freeman knows Visual Basic but is trying out Delphi. He writes: "VB has the very useful facility for creating, both at

runtime with code. Here's a minimal example: take a blank Delphi form and in the form's class definition, declare a procedure (Fig 1). Now put the following code in the FormCreate event:

```
var
mybutton: TButton;
begin
mybutton := TButton.Create(self);
mybutton.parent := self;
```

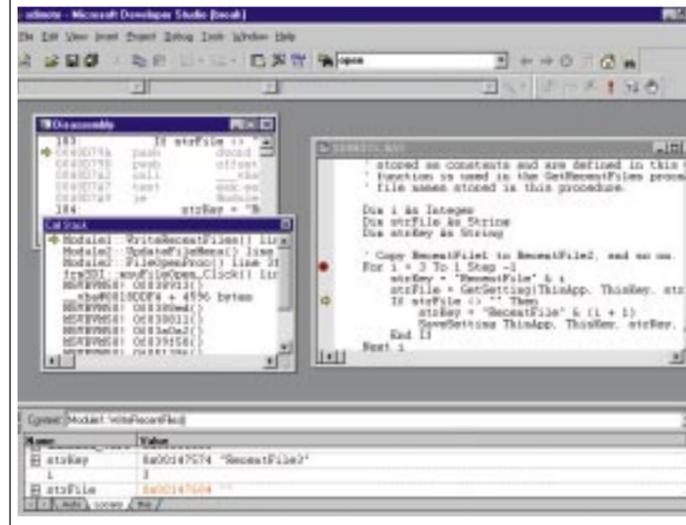
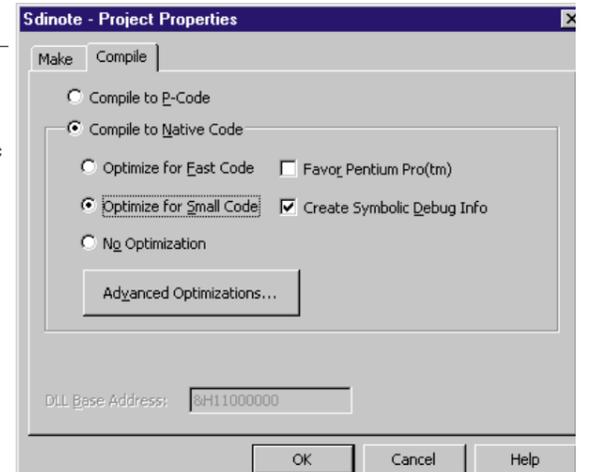
Debugging Visual Basic with Visual C++

There are several reasons why you might want to debug a Visual Basic application in Visual C++. One is that it lets you step through a native code executable, which Visual Basic cannot do. Another is that if you understand assembly language, you can view execution at a lower level than Visual Basic provides.

Here's how to do it. The first step is to compile your application to native code from Visual Basic. In the project properties, make sure that the options for Compile to native code and Create Symbolic Debug Info are both checked.

Next run Developer Studio, choose File Open, and open the executable file. It will appear in the Workspace window. Now open the source code you want to debug. This will be a form, code or class module. For example, using the Calculate sample application, I loaded CALC.FRM. You need to set the open dialog to All Files, since VB source files are not among the types recognised by Developer Studio.

The next step is to find the code you want to test, right click, and choose Insert Breakpoint. I used the Form_Load() procedure. Then choose Start



Above If you want to debug with Visual C++, compile to native code checking Create Symbolic Debug Info

Left In Visual C++ you can open the Visual Basic source and step through the code. Note all the references to MSVBM50 in the call stack window

Debug — Go from the Build menu, or press F5. The application will execute and stop at the breakpoint you have set. Now you can step through the code, inspect variable values, and view the disassembly window.

This is a good technique for solving native code problems, but not for all your Visual Basic debugging. VB's own debugger is better for most purposes. It does give an interesting insight into what happens under the surface in VB, as you can see all the calls to the runtime DLLs.

Finally, a tip straight from Microsoft's Tech Ed conference. When compiling to native code, optimising for small code is often the best choice and is what Microsoft generally uses.

mybutton.OnClick := MyButtonClick; end;

You also need to implement the MyButtonClick procedure (Fig 2).

When you run the example, a button appears on the form. Click the button and a message appears. You can improve the project by setting the properties of the

Fig 2 The MyButtonClick procedure

```
procedure TForm1.MyButtonClick(sender: TObject);
begin
showmessage('You clicked');
end;
```

button to place it on the form, change its caption, and so on.

The second reason for using control arrays is to simplify code. Taking the example of the address application, here is how you might do it. First, place 26 buttons on a form as you would with Visual Basic, using the clipboard and Delphi's alignment

tools to speed the work. Give each button a caption that is a letter from the alphabet. Then write the code for the first button's OnClick event (Fig 3).

Now select the second button. Instead of double-clicking the OnClick event in the object inspector, just select it and use the drop-down button to show the available

methods. Choose Button1Click, which is the same method the first button uses. Do the same for the other buttons. Now run the project. When you click a button, Delphi is able to read its caption and search for the right names.

The line of code "If sender is TButton" is

Fig 3 Code for the OnClick event

```
if sender is TButton then
showmessage('Search for names beginning: ' + TButton(Sender).caption);
```

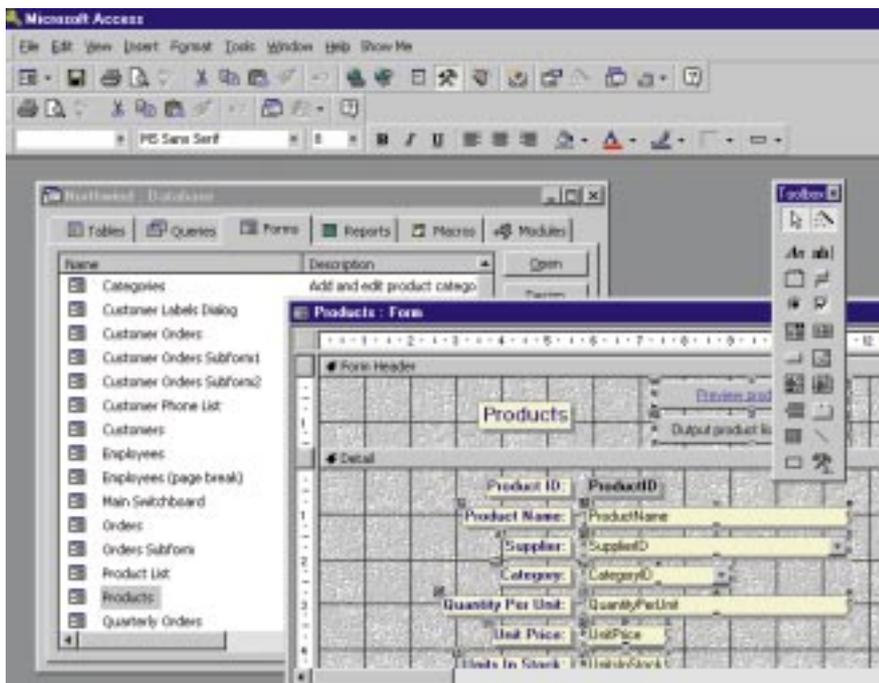
not strictly essential. It uses a feature called Runtime Type Identification to check that the clicked object actually is a TButton. The next line is a typecast, which tells Delphi that it can treat the Sender object exactly as if it were an object of class TButton, rather than TObject. That means it can inspect its caption property and determine how to proceed with the search.

Visual Basic 5.0 bug watch

Steve Simmonds has written in response to the article on VB 5 bugs, in the August PCW. He writes: "If you set the Project Properties for Sub Main as the code entry point and have both a BAS module and a form with a Sub Main, then it runs fine in VB 4 and in the VB 5 programming

The real 'Using Java 1.1'

What a difference a flash makes. Reviewing Que's *Using Java* in this column, I commented that the cover flash, which reads "Covers new JDK 1.1 features", was optimistic given that JDK 1.1 had not been finalised at the time. Java Beans was covered in all of one paragraph. Now the real *Using Java 1.1* has been published, and is enormously improved, with even coverage of all the JDK 1.1 features. This is a common-sense Java handbook which does not require previous knowledge of Java or C++, but equally is advanced enough to be genuinely useful. Each chapter is written by someone knowledgeable in the field covered, so for example the CORBA section is by a distributed system specialist. *Using Java 1.1* is recommended as an excellent overview, although of course there are now further updates to the JDK to consider. Read this book, but keep an eye on the JavaSoft web site as well.



Each release of Access brings new features, but at the expense of compatibility

environment, but when compiled to a VB 5 EXE, the FRM Sub Main runs instead of the BAS module Sub Main. This happens even if the Sub Main is declared as Private within the Form! This took me ages to find. I also had trouble with unloading, showing and hiding forms. I never found the cause, but my project ran in VB 4 but hung in VB 5, even when single-stepping. I fixed the problem by being more meticulous about unloading forms before loading a new one, and not trying to unload a form that was not already loaded."

Thanks to Steve for two useful tips. When Visual Basic 5 compiles to a native executable, it generates intermediate code which is then processed by the same compiler used by Visual C++. Although it should produce the same results, it is different code that is running. This is doubly

true if you have optimisations enabled. Therefore, it makes sense to test the compiled version thoroughly. If you have Visual C++, you can use its debugger to catch hard-to-find bugs (see p295). I could not reproduce Steve's problem though, so it may be specific to certain builds of VB 5, or there may be other factors.

Microsoft has released a service pack for Visual Studio. This is a large download from www.microsoft.com, or you can get it on a CD. The Visual Basic fixes mainly relate to custom controls. There are also important updates to Visual C++ and the Microsoft Foundation Classes.

Which MDB?

George Herrick is working on a Windows for Workgroups/NT 4 network. He writes: "I cannot get a VB 4 16-bit program on

Windows 3.11, to read data in an Access 7 32-bit database on Windows NT 4, across a network. I realise that there is a 16- to 32-bit thinking problem, but am surprised that there seems to be no way around this, even though the VB 4 program is using an SQL query.

"Am I missing something? The work-around I have adopted is to use Access 7 for the Forms and Reports, attached to Access 2 tables on the NT 4 PC, which are also available to the VB 4 16-bit programs."

A particularly irritating feature of Microsoft Access is that the format of the MDB data files changes with each new version. There is a particular problem when there is a mix of 16-bit and 32-bit systems involved, as the last 16-bit Access was version 2.0.

Thinking is not the answer. Access is not a client-server system, so when you are reading an MDB across the network from Windows for Workgroups, the code to read the data is all executing on the 16-bit client. The Access 95 version of the JET database engine does not run on 16-bit Windows, even if Win 32s is installed, so there is no way to read an Access 95 MDB from Windows 3.x. Using SQL does not change this. SQL is just a query language, and although it is most often used by client-server systems, Access uses it to query its own local databases as well.

The workaround George has adopted is exactly what Microsoft recommends. You should keep Access data in the oldest format needed to be compatible with all the clients. You can keep other elements such as forms and reports in newer MDB versions, and attach to the data as required. For mixed networks like this, there is another interesting option, and that is to adopt an intranet solution. Last month's column explained how to use Visual Interdev to create web applications based on an Access MDB or any other ODBC data source. This can be designed to work on any client machine that can run a web browser, even on non-Windows systems.

PCW Contacts

Tim Anderson welcomes your Visual Programming tips and queries. He can be contacted at the usual PCW address or at visual@pcw.vnu.co.uk.

Using Java 1.1 by Joseph Weber and others. Que. Book and CD £46.99 (inc VAT). Computer Manuals 0121 706 6000.



Dynamic digits

How about publishing dynamic data via an intranet? Tim Anderson shows you how with the aid of Visual InterDev. Find out about the latest VB Assist. And your VB queries, sorted.

Spare a thought for third-party suppliers of Visual Basic add-ons: the early versions of Visual Basic had no database support and even VB 3.0 had a hopeless grid control; serious users looked instantly for extra controls.

Visual Basic 5.0, by contrast, is loaded with extras and the development environment is smarter, too. Another factor is that you can create your own ActiveX components, further reducing the need for bought-in help. Companies like Sheridan Software are under pressure, looking for new ways of adding value.

VB Assist is a long-standing favourite, aiming to wrestle extra productivity out of the new, integrated development environment. When you install VB Assist, its toolbar appears as part of Visual Basic. Each of the 15 icons represents a separate assistant, five of which are completely new in this version.

There are some excellent ideas. For example, the Project Control Assistant (Fig 1) analyses a VB project into an Explorer-style view so you can quickly see what event-driven code you have written. Accelerator Assistant (Fig 2) deals with the awkward business of assigning unique accelerator keys to menus and buttons. It will list letters used and detect conflicts. Best of all, when a control is selected, it will show any available matching letters and

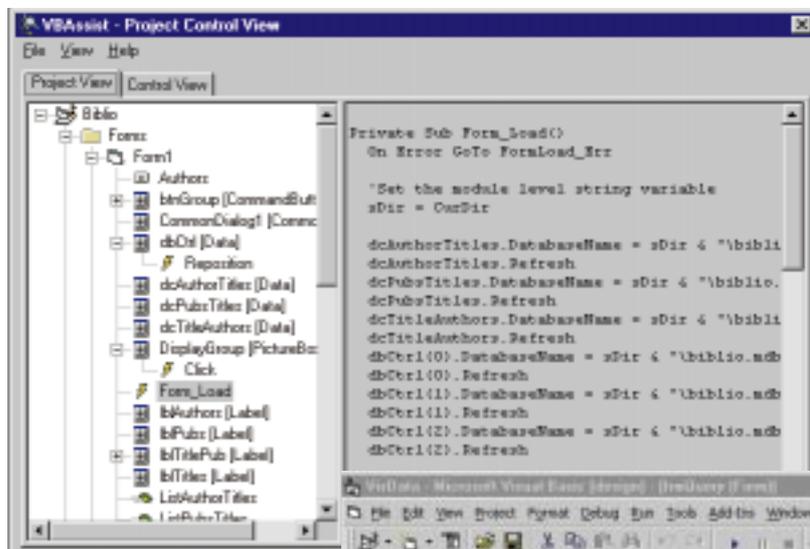


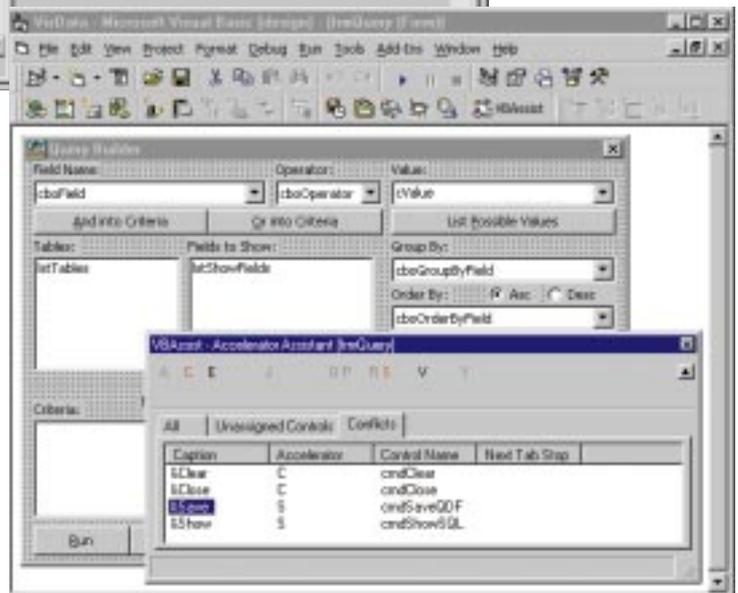
Fig 1 (left) The Project Control Assistant shows a VB project in a tree view, complete with the code behind events

assign one of these with a single click.

Those working with standard 15in monitors will like the Work Space Assistant. This adds an auto-hide feature to four windows of your choice, such as the property window and the toolbox. You can reveal a window by moving the mouse to the corner of the screen. The idea is to free more space for form design; a real problem with Visual Basic 5.0 in SDI mode.

Most of the familiar VB Assistants are still here and, in many cases, improved. These

Fig 2 (below) Assist's Accelerator Assistant detects two sets of duplicate keyboard shortcuts on this Visual Basic form. Note the VB Assist toolbar below the standard one



include a code librarian with a visual clipboard which appends a selection instead of replacing it, a clipboard history utility, a resource editor and various alignment tools.

The useful tab-order Assistant exposes the current tab order visually and lets you

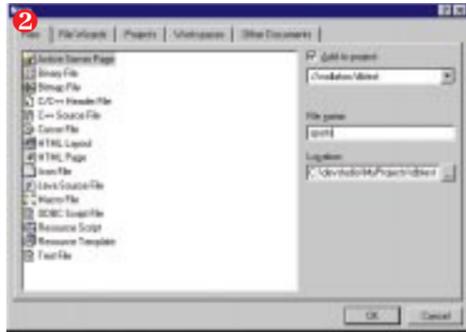
change it by clicking with the mouse. The common dialog wizard automatically writes code for using the common dialog control, while a data binding utility speeds up the process of binding controls to database fields.

Overall, it is safe to say that VB Assist has some tools you will never use and others that will genuinely save you time. VB Assist is good value for professional users.

Data access with Visual InterDev

The fundamental ingredient of an interactive web site is dynamic database access. If your site is hosted by a dial-up ISP, this is something you will need to discuss with your provider, but there are no such problems with an intranet running on a local network.

As an example, what follows is how to publish an Access MDB with Visual InterDev. For this to work, you need to be using a Microsoft web server: either Internet Information Server running on NT, or the Personal Web Server for Windows 95. The example given creates a read-only



ODBC data via OLE DB. In this example, a data source called PCWSports is configured to point at the file SPORTS.MDB. You may be running Visual InterDev on a different computer than the one hosting the web server. If so, you need to ensure that the ODBC data source is valid on the machine hosting the web server. Also in this example, I created the same directory structure on the development machine as on the server.

2. The next step is to run Visual InterDev and create a new web project by choosing

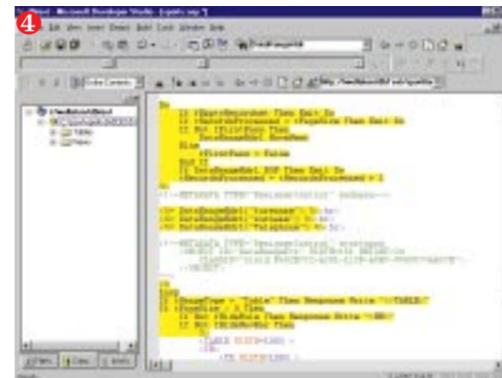


Web Project Wizard from the File – New dialog (screenshot 1).

3. Visual InterDev will create a new web, called dbTest in this example. Now choose the menu option Project – Add to Project – Data Connection. Then select PCWSports from the list of available ODBC data sources. All going well, a new data connection appears in the project window, under the default name DataConn. A data tab also comes to life, which shows tables from the selected MDB in a tree view.

4. Next, choose File – New and create a new Active Server Page (screenshot 2). This is an HTML page capable of running server-side scripts on a Microsoft web server. Visual InterDev opens the new page in the HTML editor. This example is called SPORTS.ASP.

5. Move the edit cursor to just after the



<BODY> tag and right-click the mouse.

From the pop-up menu choose Insert HTML using Wizard, and then choose the Data Range Wizard. This automates the addition of database scripting to the page, making use of two design-time ActiveX controls. The wizard asks how many records to show at a time, and you should click the Show only... option and leave it at the default of 10. Then choose Finish, accepting the default name.

6. Now the DataRangeHeader ActiveX control kicks in with a tabbed dialog (screenshot 3). In the first tab, Control, choose DataConn for the data connection and leave the command type at 0 – SQL. You can either enter an SQL string directly or click SQL Builder to create a query using Visual Data Tools. The query in this example is simple: SELECT * FROM members ORDER BY surname. You should also click



Copy Fields, which lets you choose which fields to display. This is to be a telephone directory, so only Forename, Surname and Telephone fields are needed. When you click OK, Visual InterDev copies the field names to the clipboard. You will see why in the next step. All the other options in this dialog can be left at the default value.

7. Now return to SPORTS.ASP in the HTML editor. The Data Range Wizard has filled it with script, which is tagged with percentage signs and coloured yellow to show that it runs on the server (screenshot 4). There are

also METADATA blocks, which are used by Visual InterDev to identify design-time ActiveX controls. The wizard inserted two controls (a DataRangeHeader and a DataRangeFooter) and you need to find the space between them. Position the cursor there and then choose Edit – Paste. The fields you chose in the previous step are pasted into position.

If you look closely, you will see that the field names come in the middle of a Do ... Loop block. It is confusing, because the DataRangeHeader control starts the loop and the DataRangeFooter finishes it. The idea is that this loops through all the records in the query. By placing the field names in the middle of the loop, you ensure that these fields are displayed for each record.

8. Finally, the moment has come to test the page. Right-click and choose Preview in browser. You will be prompted to save the page. Visual InterDev will display names from the database in a simple list (screenshot 5), ten records at a time. Now you can go back to the Active Server Page and start adding some formatting. For example, the name can be combined into one bold-text line by amending the first line of the field list to:

```
<b><%= DataRangeHdr1("forename") + " + DataRangeHdr1("surname")%></b><br>
```

9. A common misconception is that Microsoft's new web development tool only works with Internet Explorer. In fact, Active Server Pages work well with Netscape or any other browser. There are two things of which you need to be aware. First, Netscape does not support ActiveX controls or VB Script. Second, Netscape and Internet Explorer do not always behave in the same way when confronted with identical HTML. Sometimes you can fix a page that works on one but not on the other, with some trivial changes.

Wizard ways

Roy Vinyard asks: "Can you help me with the problem of deciding which files are required for distribution of a VB4 program

Taking a look at books

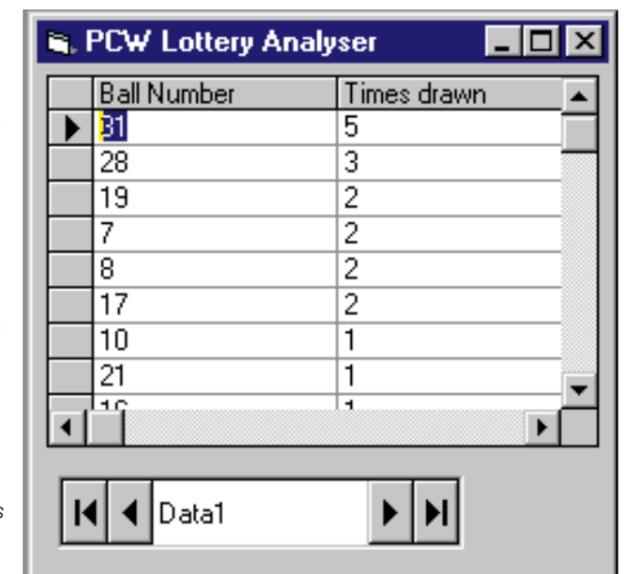
■ **Office 97 Developer's Handbook**
by Christine Solomon

Ease of use in Office is a misconception, claims the author of this guide to Office development. Although anyone can type a document and print it, getting maximum productivity from these large applications takes time and skill. Not just programming skill either: a good part of this book is devoted to general themes like planning, communication and design. Other sections deal with Visual Basic, automation, data access, API programming, intranets, and chapters on the main Office applications (excluding Outlook). The book scores highly on readability, common sense and tips based on experience. Although it dips into some advanced issues, it is not an in-depth technical guide. It is ideal for managers wondering what Office can do for them, or for developers wanting a sound introduction.

■ **Teach Yourself Visual InterDev in 21 Days**
by L. Michael Van Hoozer

Visual InterDev is the Visual Basic of web development and uses much the same technology, served up in a different form. Its aim is to make it easier to create interactive web sites (for example, web sites which query a database rather than presenting static HTML pages). This is a tutorial of 21 lessons, intended for those who already have some knowledge of how web pages work. Topics include an explanation of the basic concepts, the FrontPage Editor, Image composer, Visual Data Tools, and design-time ActiveX controls. Unfortunately, there are so many topics that most get only shallow coverage. I was disappointed to see little attention paid to a key issue: how to make Visual InterDev applications work with Netscape as well as Internet Explorer. The book does provide a good overview, though, and will be helpful to those struggling to get started with this powerful but initially bewildering product.

Fig 3 This lottery analyser works by assigning an SQL COUNT query to a data control. There is actually no need for the data control to be visible at runtime



(16- or 32-bit)? I know about VB40032.DLL and, for example, that SPIN32.OCX is needed if I use a spin control, but what about some of the files that the Setup Wizard includes like MSVCRT20.DLL, or MFC40.DLL, or DAO2532.TLB? Is there an easy way to find out what is needed?"

Removing files that the setup wizard thinks it needs is not recommended. These decisions are based on an INI file that lists dependencies, which you can

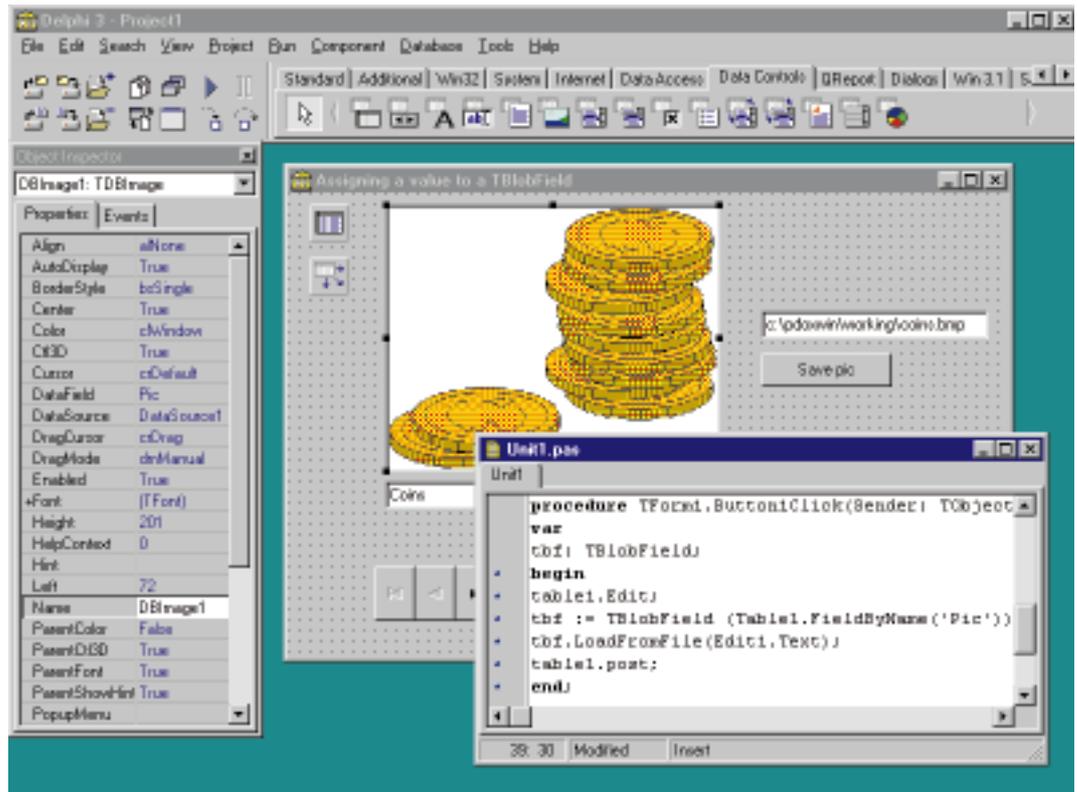
find in the setupkit directory. If MFC40.DLL is included, for instance, it means that one of the components was built using the Microsoft Foundation Classes and uses this runtime library. Although it is very likely already installed on most Windows systems, it is nevertheless essential that you include it because you will almost inevitably discover someone who needs the file or who has the wrong version installed.

As a further check, the WISE installer has a clever feature whereby it runs in the background at runtime, capturing details of all the files used. And finally, a good tip is

Listing 1: Some clever SQL for Lottery balls

```
SELECT balls.BallNo, Count(balls.BallNo) AS CountOfBallNo
FROM balls
GROUP BY balls.BallNo
ORDER BY Count(balls.BallNo) DESC;
```

Fig 4 TBlobField's LoadFromFile method gives you a way to store pictures in a Paradox database, accessed from a Delphi application



to remove references to any unused VB components before making a final build of your project.

Top draw

Ben Wright is analysing the lottery. He asks: "The lottery numbers are stored in an Access database with each draw being a different record. I would like it to show the week's 'hot' numbers; those that have come out the most times. How would you do this in Visual Basic? And, I would like to do some graphs showing how many times each ball has come out."

The starting point for any database problem is to work out how to store the data. There are three possibilities.

First, a bad idea: you could have a record for each draw, with the numbers stored in a single results field, perhaps divided by commas. The problem is that when you analyse the data, you will need to search the database, number by number, counting the number of times each one occurs in the results field. But this is slow and unwieldy.

Next, an easy idea: have a record for each ball, and a TimesDrawn field showing the number of times drawn. This gives you a small database and a problem that solves itself. Simply order by the TimesDrawn field to get the result. The snag is that you cannot reconstruct any individual draw, or

check the accuracy of the database.

The most sophisticated approach is to have a record for each ball drawn, with two fields: one for the date of the draw and another for the ball drawn. Then you can use some clever SQL to get the information you need (Listing 1, p265). You can either return this query into a snapshot-type recordset or use it as the RecordSource of a data control. It will give you a record for each ball, with a calculated field showing how many times it has come up (Fig 3). But how do you know how to construct SQL queries like this without having to write to us here at PCW each time? The answer is, get hold of an SQL tutorial.

Another good tip is to build queries using the Access visual query builder and then choose the SQL view to see what has been generated. You can copy and paste from there directly into a Visual Basic project.

Delphi and TBlobField

J Watson writes: "I have a Paradox table that has a field of type ftgraphic. It has data that was entered in Paradox so I know that it works. The image displays in a DBImage component. I cannot assign a value to new records that I create. If I try:

```
Table1['Picture'].LoadFromFile(Edit1.Text);
```

it compiles but when run I receive an error; 'Variant method calls not supported'."

LoadFromFile is a method of TBlobField, but "Table1['Picture']" returns a field value. You can achieve what you want with a typecast (see also Fig 4):

```
var
tbf: TBlobField;
begin
table1.Edit;
tbf := TBlobField (Table1.
FieldByName ('Picture' ));
tbf.LoadFromFile(Edit1.Text);
table1.post;
end;
```

This works by converting the TField component returned by the FieldByName method into a component of type TBlobField. Then you can use the special methods of TBlobField, including LoadFromFile.

PCW Contacts

Tim Anderson welcomes your Visual Programming tips and queries. He can be contacted at the usual PCW address or at visual@pcw.vnu.co.uk.

VB Assist 5.0 costs £130 (ex VAT) from Contemporary Software 01344 873434; further information at www.shersoft.com.

Books from Computer Manuals 0121 706 6000
 • **Teach Yourself Visual InterDev in 21 Days**, SAMS, £37.50 (inc VAT) book and CD.
 • **Office 97 Developer's Handbook**, Microsoft Press, £37.49 (inc VAT) book and CD.



Stabilising influence

Is Visual Basic 5.0 flaky? Tim Anderson offers some tips on preventing problems. He also tries out BoundsChecker for Delphi and answers your visual programming queries.

Several readers have reported problems with Visual Basic 5.0. Tim Gathercole writes: "I upgraded to VB5 Professional about three weeks ago and the program is proving to be highly unstable when working with larger projects. Four of my projects, which worked fine under versions 2.0, 3.0 and 4.0, now either crash version 5.0 at runtime or work in runtime but crash when compiled. I got the same result on both a Pentium and a Pentium Pro machine, each with 32Mb of RAM." He goes on to enquire about a VB5 to VB4 converter, in order to get his applications running again.

If anyone doubts that VB 5.0 is less stable than it should be, try this: select two or more controls on a form (anything with a font property will do). Have the properties window floating and the environment set to MDI (the default). Then try to change the font. VB responds: "This program has performed an illegal operation and will be shut down".

While this font problem is a reproducible bug, other faults seem to be annoyingly unpredictable. Some users experience constant crashes, while others find VB 5.0 stable and reliable. Native code compilation can be problematic. Microsoft claims a bug-fix release is on the way, but why all the

problems? There seems to be a variety of reasons: the faster forms engine is fussier about the video card and drivers than previous versions; native compilation introduces a new layer of complexity; and VB's dependence on ActiveX makes it susceptible to any problems with OLE or the system registry. Here are some tips for those struggling to get VB 5.0 working:

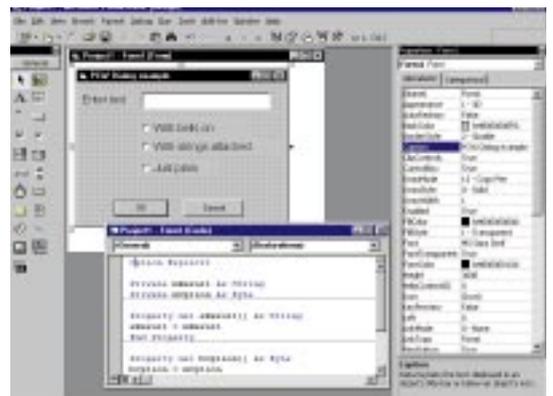
- Check "When a program starts — Save Changes" in the Options-Environment dialog.
- Do a fresh installation of Windows and VB, preferably onto a newly-formatted hard disk. This brute force approach corrects two common problems: mismatched system DLLs and/or a corrupt registry.
- Double-check API calls and any calls to external DLLs.
- Check your current memory and add more if possible. Thirty-two megabytes is good.
- VB 5.0 sometimes corrupts memory when forms with many controls are loaded. Removing a form from a project and adding it back can help.
- Deselect toolbars in the view-toolbars-customise menu.
- Remove add-ins.
- Switch to standard Windows video drivers.

Dealing with dialogs

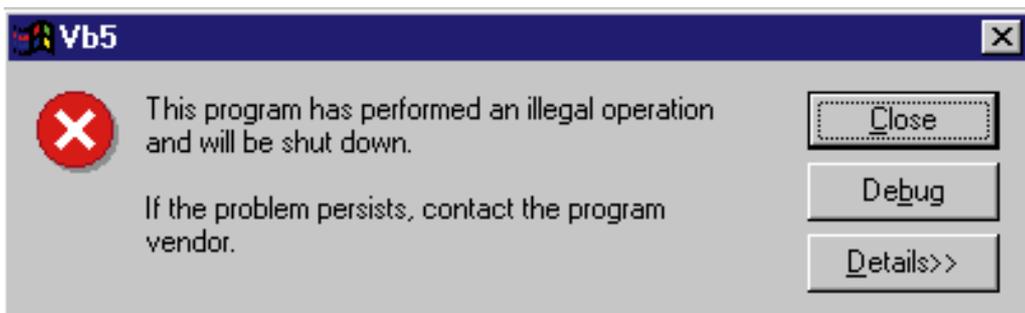
Tim George is using VB4 and tells me: "I would like to be able to create a dialog box which returns a single value, much like the inputbox function but with extra functionality. My problem is that I wish to pass data to and from the dialog box without resorting to global variables, but using some kind of parameter list?"

In the days of Visual Basic 3.0, global variables were hard to avoid. The only other option was to keep the form in memory by using the Hide method rather than unload. Then you could refer to the values of the form's controls after the user has closed the dialog. A better approach in VB 4.0 or higher is to use custom properties.

In the General section of a form module,



Above A good way to do dialogs in Visual Basic is by using property procedures. The secret is to remember that in VB 4.0 and higher, forms are like class modules
Left Visual Basic 5.0 has many great features, but is spoilt by instability during development and at runtime



Taking a look at books

■ Using Delphi 3.0

by Todd Miller and David Powell
Delphi's weakest point is its documentation. The online help is notorious for broken links and skimpy examples, while the printed manuals leave important subjects virtually untouched. That opens the way for third-party alternatives like this.

The book's scope is impressive. In just over 1,000 pages it runs from introductory basics to the mysteries of "thunking", calling 16-bit DLL functions from 32-bit code or vice-versa. The level is suitable for anyone with some programming experience. There is balanced coverage of neglected topics like threads, OLE automation, database programming and creating ISAPI DLLs for web server applications.

It is weak on techniques for object-oriented programming. The bundled CD includes example code, product demos, and four strangely unrelated Que titles in HTML format. Recommended.

■ Presenting JavaBeans

by Michael Morrison
Visual Basic and Delphi programmers should take a keen interest in JavaBeans. Beans are reusable visual or non-visual components that expose properties and handle events, just like ActiveX controls in VB or components from Delphi's Visual Component Library.

Once suitable visual tools arrive, JavaBeans will make rapid application development easier and more effective. This short title explains the JavaBeans API,

and gives four complete examples using Sun's Bean Development Kit. With a warm, good-humoured writing style, the author explains key concepts like properties, introspection, event handling and persistence. Some knowledge of Java is assumed.

It is a basic introduction, useful for anyone wanting to get up to speed on what JavaBeans is all about.

■ Visual Basic 5.0 Programmer's Guide to the Win32 API

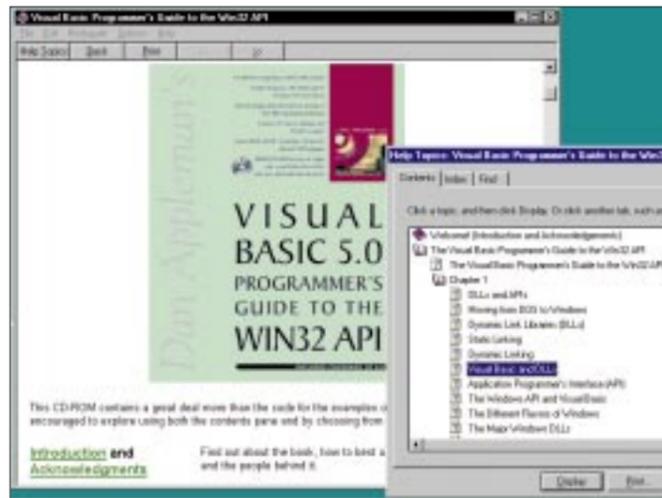
by Dan Appleman
You have to admire an author who writes in his foreword: "If you already own the original, the changes do not justify the price of a new book."

The book in question is Appleman's classic guide to calling the Windows API from Visual Basic. The previous edition was extensively revised to cover the transition to 32-bit Windows, but this time around the API is essentially the same, so fewer changes are necessary.

Actually, Appleman is right. The changes are not enough to justify a new

purchase unless you have to have the latest of everything. The new edition is revised for Visual Basic 5.0 although version 4.0 is still extensively covered for the sake of its 16-bit compatibility. The frequent plugs for products from Desaware, the author's company, are a little tiresome, though.

The book remains a superb reference for those who need to go beyond Visual Basic's built-in functionality. For advanced work with menus or fonts, for example, it is near-essential. The accompanying CD has a slightly expanded version of the book in Windows help format, along with Desaware's API class library.



The CD which comes with Appleman's Guide to the Win32 API has the whole book as a Windows Help file. This is easier to navigate and search than the more common HTML or PDF electronic formats

declare private variables for the dialog's values. Next, create property procedures to access the variables. In your dialog's OK procedure, write the values from the dialog controls back to the private variables. Finally, you can create a ShowDialog method which takes parameters to initialise the dialog values.

When you have finished with the dialog, use SET MyForm = Nothing to clear the form module from memory. Here is some example code:

'Code for the form module

```
Private mResult As String
Property Get sResult() As String
sResult = mResult
```

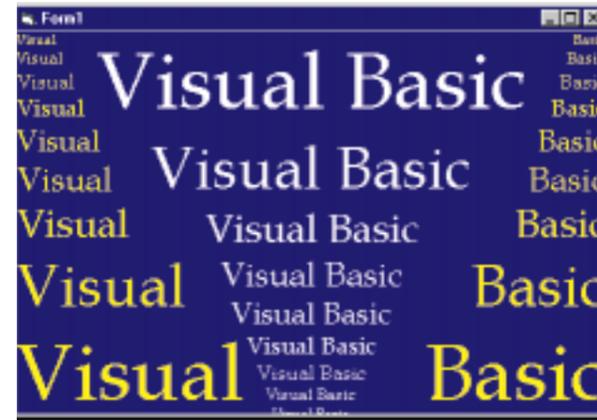
End Property

```
Sub ShowDialog(sParm As String)
mResult = sParm
Me.Show 1 ' show form modally
End Sub
Private Sub Form_Load()
Text1.Text = mResult
End Sub
Private Sub cbOK_Click()
mResult = Text1.Text
Unload Me
End Sub
' code to use the dialog
MyForm.ShowDialog("MyParameter")
MyResult = MyForm.sResult
Set MyForm = Nothing ' clears form
module from memory
```

Poster poser

Liam McAllister is working with fonts in Visual Basic. "I am trying to write a program to print posters of various sizes. When I want to print out the poster, I take a bitmap copy of a form with labels and send it to my printer, stretching/shrinking (using API functions) as necessary. This works fine up to about A4 size but, as you can imagine, when stretched to this size the text looks very jaggy and rough. Also, the text I use on the labels is varying in the font style and the text entered: I would like the text entered to fit into boxes of a predetermined size."

Sending text to the printer as a bitmap is not ideal. It is better to use the Print method of the Printer object, as this will allow



Left By writing code to print directly to a form, and using TextWidth to measure strings, you can obtain good-looking font effects

Windows to scale the font correctly to the resolution of the output device. The disadvantage, though, is that you need to work harder to position text and graphics elements correctly, using the printer object's CurrentX and CurrentY properties.

To determine how much space text will take up when displayed in a label, use the TextWidth method. TextWidth takes a string parameter and returns the width of the text when printed in the current font and size. Unfortunately, labels do not have a TextWidth method, so one possibility would be to use a hidden picture box, adjust its font as needed, and measure text with TextWidth to discover how much will fit into a label which has the same font.

Whatever you do, do not place numerous picture boxes on a form to use as fancy label controls, as this is wasteful of Windows resources. Labels are better or, better still, print directly to the form using its print method. This approach has advantages when you need paper output as well, as you can easily adapt the code to work on the printer object instead of the form.

Book search

People often ask me for book recommendations. Geraint Preston asks: "Could you recommend the best book from which to learn C++? I've worked my way through A Book on C and I'm keen to go on to the next stage. Could you also recommend (though my need is less pressing) the best book from which to learn Visual Basic? I bought Microsoft VB and C++ together at a special price."

The problem with the C++ question is that there are at least four separate skills to learn. First, there are the bare bones of the language, and there are plenty of tutorials available. Next there is the question of object-orientated programming. Third, a

with a class library which needs learning in its own right. An additional factor for Windows programmers is that you need to know how Windows itself fits together, which means familiarity with the Windows API. Since Geraint has Visual C++, a good choice would be *Inside Visual C++* by David Kruglinski (Microsoft Press).

The tutorials included with Visual C++ are also good. Be warned that these resources are specific to Windows, the Microsoft Foundation Classes and Visual C++. If you want general C++ skills, they may be positively unhelpful.

Visual Basic is easier to learn. Another advantage is that the supplied manuals are actually very good. If your version came without printed manuals, you can buy them separately as Microsoft Press titles. Once you have digested the official *Programmer's Guide*, you will be ready to tackle a more specialist title depending on which aspect you want to focus on. Books are regularly reviewed in this Hands On section and past columns in this section are included on our PCW cover-disc.

Name that user

P Blomfield is using VB 4.0. He asks: "On a network, how do you find the machine name? How do you find the username of the current user logged in?" This information is provided by the Windows API. The relevant functions are

GetUserName and GetComputerName. If you copy the declaration from VB's API viewer, you can then call the functions (Fig 1).

BoundsChecker comes to Delphi

BoundsChecker is the flagship product of NuMega Technologies. Its purpose is to track down bugs which are otherwise hard to catch. Until now it has been for C or C++ developers only, but version 5.0 gives Delphi applications the same treatment.

BoundsChecker installs itself into the Tools menu of the Delphi 2.0 IDE. A patch to work with Delphi 3.0 should be available from NuMega's website by the time you read this. Delphi 1.0 is not supported.

The natural question to ask is: what can BoundsChecker do that Delphi's own debugger cannot? One of the issues is detecting memory leaks. For example, your program might include the following code:

```
var
nullstring: pchar;
begin
nullstring := strAlloc(50);
strcpy(nullstring, 'Never freed');
end;
```

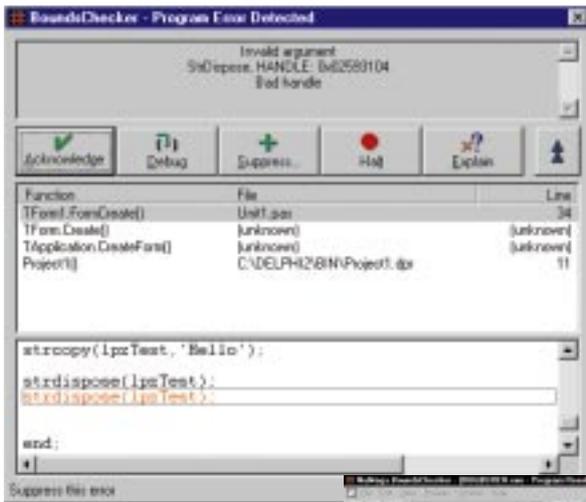
Delphi's debugger will not show any fault and the program will run fine. The problem is that memory is allocated but never freed. When BoundsChecker is active, the leak is reported in an error window after the program runs. Double-clicking the error takes you to the point in the source where the memory was allocated.

Another type of leak occurs when objects are created in code but not freed. For example, you might have a listbox which contains a list of music CDs. Using the AddObject method, you could associate a TStringList object with each item in the list box, perhaps to store a track listing. When the listbox is destroyed, Delphi will automatically free the items in the list but

Fig 1 Finding the user

```
Dim sBuffer As String
Dim lRetVal As Long
Dim lLength As Long

sBuffer = String(32, " ")
lLength = 31
lRetVal = GetComputerName(sBuffer, lLength)
Label1.Caption = "Computer name: " & Left(sBuffer, lLength)
sBuffer = String(32, " ")
lLength = 31
lRetVal = GetUserName(sBuffer, lLength)
Label2.Caption = "User name: " & Left(sBuffer, lLength)
```



Left This pop-up dialog intercepts an error and offers several options for dealing with it

Below BoundsChecker shows the type of error, source code when available, call stack, and online help explaining how to fix it

not the associated StringList objects. Running this through BoundsChecker reveals the leak, for example, "32 bytes allocated by ReallocMem in TStringList.SetCapacity". The solution is to write code that frees the StringList objects before the listbox is destroyed.

BoundsChecker illustrates the point that Delphi is not as safe an environment as Visual Basic or even Java. These languages have a feature called garbage collection which frees objects once no valid reference to them exists. Advanced Delphi programmers need to do their own memory management to some extent. In particular, working with the Windows API inevitably means using pointers, the most common source of memory errors.

NuMega's BoundsChecker reports API and OLE errors as well as memory leaks. It also has an event reporting feature. When enabled, this collects all the Windows API calls, parameters and messages your program sends and receives. The mass of resulting information does make it hard to track down problems, but with patience this can reveal places where code is not working or efficiency can be improved.

This is a specialised tool. While it is great for finding memory problems, it will not help you find logic errors in your code. It is most useful for more advanced programmers with large applications to debug, and in these situations should soon repay its purchase price. Its integration with Delphi is impressive, or will be when the Delphi 3.0 patch is available.

Note that BoundsChecker for Delphi is not quite as capable as the C++ version. The latest Visual C++ edition uses a newer



technique called FinalCheck which can find an additional set of memory and pointer errors. Even so, BoundsChecker is an excellent resource for Delphi developers.

And finally...

An unfortunate mix-up resulted in the June 1997 issue Hands On Visual Programming feature not being printed last month — in its place, an older version appeared. The actual June issue column can be found at www.cix.co.uk/~tim-anderson. Topics include Visual Basic in Visio, VB 5 subclassing, and exploiting units in Delphi.

PCW Contacts

Tim Anderson welcomes your Visual Programming tips and queries. He can be contacted at the usual PCW address or at visual@pcw.co.uk.

BoundsChecker 5.0, Delphi Edition, costs £345 + VAT from Grey Matter 01364 654100. Further information at www.numega.com.

The following books are available from Computer Manuals 0121 706 6000:

- *Visual Basic 5.0 Programmer's Guide to the Win32 API* by Dan Appleman, Ziff-Davis Press. Book and CD £54.95 (inc VAT). Further information at www.desaware.com.
- *Using Delphi 3.0* by Todd Miller, David Powell and others, Que. Book and CD £46.99 (inc VAT).
- *Presenting JavaBeans* by Michael Morrison, Que. Book and CD £32.95 (inc VAT).



The Outlook is variable

Tim Anderson finds Outlook, the latest Visual Basic-enabled Office 97 application, to be a powerful but frustrating solution. He gives his views about programming and creating forms.

Cedric Maddox writes: "I'm trying to get to grips with Delphi. Your review of the Appleman Guide was very interesting and I would like a similar guide for Delphi: can you recommend one? If there isn't a Delphi guide available, would the Appleman Guide for VB help with Delphi API calls?"

Much of Daniel Appleman's guide to the Win32 API (Ziff-Davis) would be useful to Delphi developers since it is, after all, the same API. I hesitate to recommend it though, because large parts are specific to Visual Basic. Calling the Windows API is actually easier in Delphi than in Visual Basic: partly because the language is a better fit, with direct support for pointers and easy handling of Windows messages; and partly because Delphi's developers have helpfully defined the types, function and procedure headers for you.

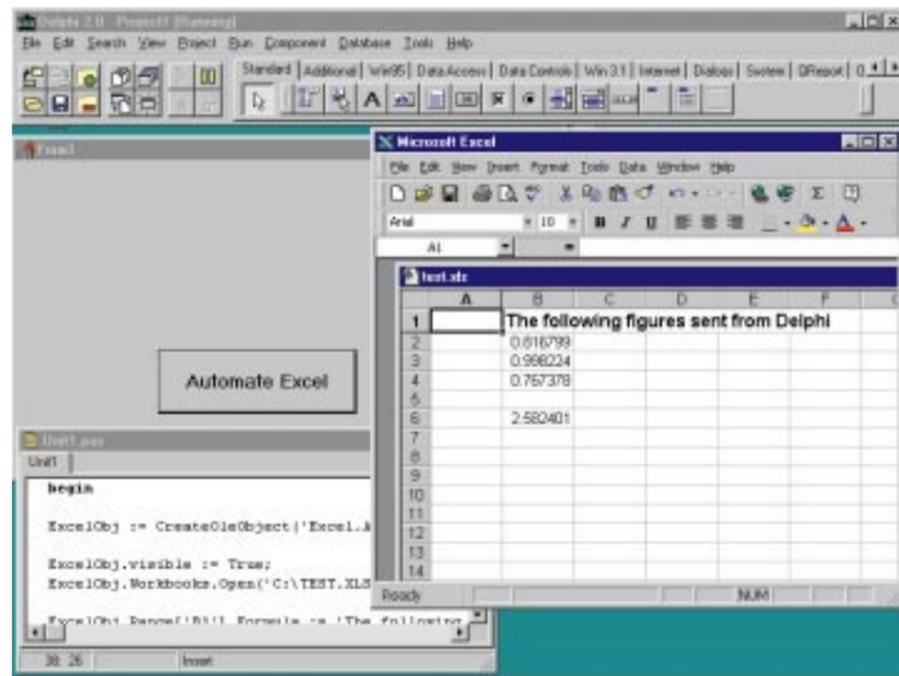
Often, you can use an API function as if it were native Pascal. But although calling the API is easy, understanding how it works is another matter. More advanced Delphi titles like *Delphi 2.0 Unleashed* or *Delphi 2 Developer's Guide*, both from Sams, contain helpful API tips. The official Microsoft reference is essential, and an online version comes with Delphi. Finally, Charles Petzold's *Programming Windows 95* (Microsoft Press) is useful not as a reference title, but as an explanation of how Windows works behind the scenes.

Unfortunately, if you want to get deeply into the Windows API, you have to put up

with reference material aimed at C and C++ programmers, since Windows itself is mainly written in these languages.

Delphi and Excel 97

Another reader's problem is: "According to my testing, there is a problem with Delphi 2 and OLE-automation using the version of Excel delivered with Office 97. Specifically, I can start Excel via OLE-automation and do some things (like `ExcelApp.Visible := True`), but if I try to access a range, for instance, it just crashes with an `EOLESysError`. Exactly the same test program works fine if I uninstall Office 97 and use Office 95 instead."



Get the syntax right and automating Excel is a powerful way of extending a Delphi application (see "Delphi and Excel 97", below)

I suspect this is a bracket problem. The code in Fig 1, when used with Excel 97, fails with a "Member not found" message. The solution is to replace the last line with:

```
RangeObj := ExcelObj.  
Range[ 'B2:B4' ];
```

Rich text problems

Gavin Docherty writes: "For about three months I have been trying to paste bitmaps, metafiles and OLE objects into the standard richtextbox control found in the 32-bit common control DLL but without success. Then, to my amazement, you covered the subject in your May column and gave code examples for VB,

Book Reviews

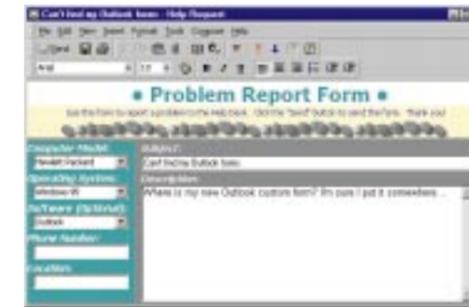
■ Building Applications with Outlook 97

£37.49 (book and CD)
Microsoft Press

Outlook is a strange combination of elegant simplicity and arcane complexity. The Outlook bar is delightful, with easy access to network and internet mail, appointments, contacts and to-do list. With its own form designer and a version of Visual Basic, it seems the ideal platform for building groupware applications. Start developing, though, and Outlook shows its other face. Want to save a form? There are three ways to do it, claims the online help. Did you want the open folder, the file or the forms library? And if the latter, did you want the personal forms library, the folder forms library, or the organisation forms library?

The other problem is that Microsoft seems determined to disguise the close relationship between Outlook and Exchange. This book is a case in point. The cover wording refers several times to groupware but never to Exchange. Without Exchange, though, Outlook is only suitable for groups of one.

If you do have Exchange Server and want to develop with Outlook, this title is all-but essential. It describes Outlook's main elements and explains how to use the form designer and Visual Basic Script. It concludes with a step-by-step guide to creating three sample applications: the first is a form for requesting business cards, the second a help desk application, and the third tracks



The Help Desk application is explained in *Building Outlook Applications*

document production. This last is the most interesting since it links to an Access database using Data Access Objects. It's a useful guide, and really should have been part of the Office 97 Developer Edition. The only other caveat is that you will need a lot more help than this when it comes to managing the back-end of an Outlook application, Exchange Server itself.

■ Using Visual Basic 5.0 by Mike McKelvy, Ronald Martinsen and Jeff Webb — £36.99, Que

Smartly published to coincide with the release of Visual Basic 5.0, this title in Que's classic "Using" series begins right at the beginning, with topics like what is a program? And what is a variable? By the end of its 950 pages it is tackling API programming, callback functions and remote automation servers. The result is a comprehensive book, but lacking in sparkle.

There is too much here for real beginners, while experienced VB developers migrating to version 5.0 will find themselves skipping large chunks of the material. It has more the style of a manual than a real-world developer's book. Because Visual Basic is now such a large product, this comprehensive approach means

little depth on any individual subject.

This is a good buy if you do not have printed documentation, with clear, thorough explanations covering every aspect of Visual Basic. Others will be better served by one of the many more specialist Visual Basic titles now available.

■ Instant Visual Basic 5.0 ActiveX Control Creation — £27.49, Wrox Press

Sporting no less than seven authors, this title covers the hottest new feature of Visual Basic 5.0: the ability to create ActiveX controls. It mainly covers the Control Creation Edition, although it will be equally useful to owners of the full version of Visual Basic. It is aimed at experienced VB developers.

After an introductory section, the main part of the book takes you, blow-by-blow, through developing several example controls, including an aggregate control, a data-bound control, and a user-drawn control. An aggregate control is one that contains several other controls, while the term "user-drawn" describes a control whose visual display is handled entirely by the program.

Despite the book's multiple authorship, the style is clear and consistent. Creating ActiveX controls presents many new issues for VB developers and this is a helpful and detailed guide. It would be better still if more space were given to the design issues behind component programming as opposed to just the mechanics of how to do it. Other vital topics are version control and web security, neither of which are given sufficient coverage. This does not detract from the high quality of the subject matter included: recommended.

● The above books are available from *Computer Manuals* on 0121 706 6000.

but I couldn't get it to work. What have I done wrong?"

Unfortunately, the news is not as good as I thought. The code printed in May's issue was tried and tested with Visual Basic 4.0. But at some point, some application or other had installed a later version of the RICHTEXT32.OCX component which makes it work, complete with an updated help file. This later version is required in order to work with pictures.

Programming Outlook 97

Outlook is an excellent starting point for an Office 97 application. Most people need an email reader and an address book or contact manager, and Outlook does both. The natural next step is to add functionality. For example, Outlook's contact menu already has an option to start a new letter to the current contact.

Fig 1: A problem with brackets

```
var  
ExcelObj: variant;  
RangeObj: variant;  
  
begin  
  
ExcelObj := CreateOleObject('Excel.Application');  
ExcelObj.Visible := True;  
ExcelObj.Workbooks.Open('C:\TEST.XLS');  
RangeObj := ExcelObj.Range('B2:B4');  
...
```

You might want to add new options, perhaps a choice of several standard letters. Getting more ambitious, you could pull in other information such as account information or product preferences. Another scenario could find you placing an

order for a contact you have just called. Many things are possible since Outlook has a forms designer and a scripting language, but this is VB Script and not the powerful Visual Basic for Applications. VB Script is the cut-down version of Visual

Basic first used in Internet Explorer.

Outlook is a handy contact manager but its database is a simple flat-file affair which is not suitable as the main data store for a business. The key then is to integrate Outlook with other data sources. Since VB Script has neither built-in database features nor the GetObject or CreateObject functions needed for programming COM objects, it does not, at first, seem promising.

By a roundabout route, though, it does

have those functions. Outlook's Application object has a CreateObject method that opens the door to Data Access Objects as well as the automation of applications like Excel and Word. This means you can keep a key field in each contact record that matches the key field in an external database (which might be a desktop database like Access) or an SQL server, and link to this external data as needed (see "An Outlook example, page 300). Then you can use Outlook for its

Creating and saving Outlook forms

Outlook form-handling is perverse. For a start, you cannot just get on and design a new form as you would in Visual Basic. The best way to design a form is to pretend you want to create a new item in the current folder. That opens a blank, default form. Then you can choose Design Outlook Form from the Tools menu. Note that some forms, like the Contacts form, cannot be completely customised. Some parts are read-only. There are plenty of spare tabs, though, which you can use as you want.

It is when you have designed your form and wrestled with the wretched script editor that the real fun begins. It is no use just saving the form, since all that does is to create one new record with a customised form. You must choose Publish Form As... from the File menu. You can publish to several locations, depending on how widely the form should be available. If you do not have Exchange server, or are working on a test form, the best choice is either your personal forms library or in the specific folder where it is needed. If you choose the latter option, it gets its own entry on the Compose menu.

Next, you have to tell Outlook to use your new form as the default for new items in this folder. To do this, right-click the folder name in the folder list and choose Properties. On the General tab is an option: "When posting to this folder, use..."; here you can specify the new custom form. Now your custom form is used by default for new entries. But it is not over yet. If you try opening any existing items in the folder, you find the old form still being used. The solution is to run a short VBScript routine. Each item in Outlook has a MessageClass property, a concept alien to VB developers but familiar to experts in Microsoft's Mail API, or MAPI. This property determines the form used to view the item.

This is the code to change it:

```
sub UpdateClass

Set currFolder = Application.ActiveExplorer.CurrentFolder
numItems = currFolder.Items.Count

For countvar = 1 to numItems
Set currItem = currFolder.Items(CInt(countvar))
currItem.MessageClass = "IPM.Contact.PCW Sports Club"
currItem.Save
Next
end sub
```



A key step is to specify the default form for posting in this dialog

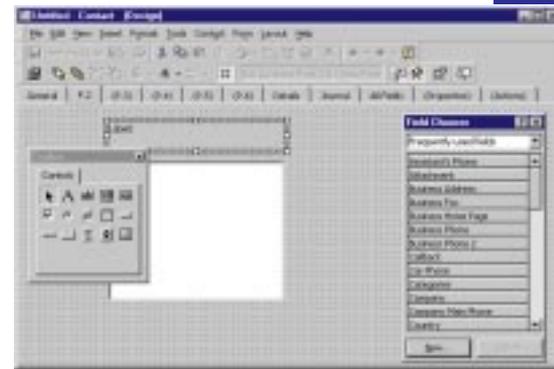


Fig 2 Create a new contact and then choose Design Outlook Form to open the form designer

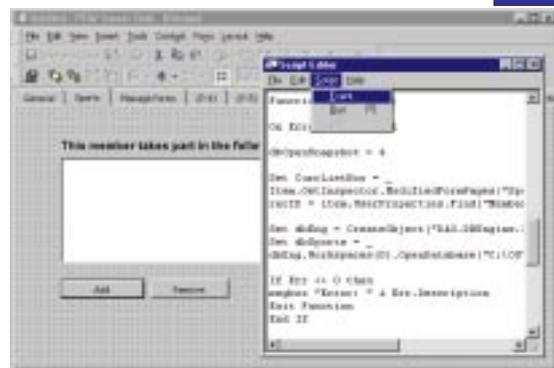


Fig 3 Choose View Code to open the Script Editor. The cunningly-titled Run option does not run the code but merely checks the syntax

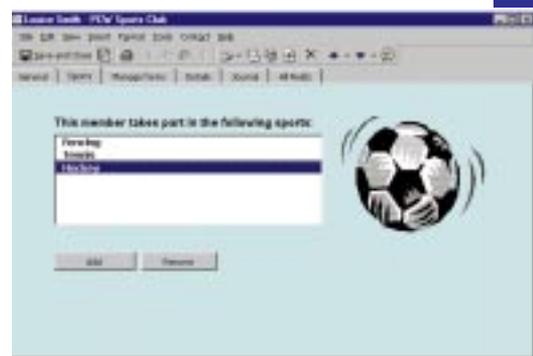


Fig 4 When the form opens, Outlook looks up the list of sports from an external database

address book and calendar features but keep the mission-critical data in a fully-fledged database management system where it belongs.

Embark on Outlook development and you hit the bleeding edge of Office 97. Documentation is scant, some procedures are hopelessly counter-intuitive, the script editor is primitive, and late binding of COM objects means performance is poor compared to real Visual Basic. It is such a

Fig 5: Accessing additional data from within Outlook

```

Function Item_Open()
On Error Resume Next

dbOpenSnapshot = 4

Set CurrListBox = _
Item.GetInspector.ModifiedFormPages("Sports").LstSports
recID = item.UserProperties.Find("MemberID").Value
Set dbEng = CreateObject("DAO.DBEngine.35")
Set dbSports = _
dbEng.Workspaces(0).OpenDatabase("C:\OUTAPPS\SPORTS.MDB")

If Err <> 0 then
msgbox "Error: " & Err.Description
Exit Function
End If

sql = "Select * from sports, sportlink where sports.ID = sportlink.sportID "
sql = sql & "and sportlink.MemberID = " & recID

Set snSports = dbSports.OpenRecordset(sql, dbOpenSnapshot)

If not (snSports.eof and snSports.bof) then
snSports.moveLast
snCount = snSports.Recordcount

snSports.MoveFirst

For countvar = 1 to snCount

currListBox.AddItem(snSports.Fields("SPORT"))
snSports.MoveNext

Next

End if

snSports.Close
dbSports.Close

End Function

```

nice component in other ways, though, that it is worth persevering. Expect it to get easier in the next version.

An Outlook example

The following example uses the same Sports Club database as the recent *Hands On Visual Basic Workshop* (PCW February-May issues).

It is a relational database with three tables. The Members table includes name and address details, and is easily imported into a new Outlook contact folder. When mapping the fields, it is important to include

the key field which uniquely identifies each record. Outlook will not let you map an imported field directly to a custom field, so the solution is use a spare built-in field. For a tidy result, you can then add a custom field to hold the ID number and write some code to copy it across. In the example, this field is called MemberID.

Note that not all the data is imported, but only the Members table. In particular, the information about which sports each member enjoys is not available. The trick now is to access this additional data from within Outlook, without actually importing

a copy of the tables.

Here's how to do it (also, see **Figs 2-4**, page 299):

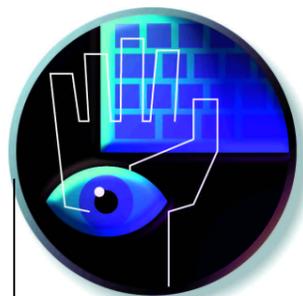
1. Open the contacts folder and choose New Contact from the Contact menu. This opens the built-in Contacts form.
2. From the Tools menu choose Design Outlook Form. This opens the form in design mode with six spare tabs available. Click the second tab, rename it Sports, and add a list box control. Name the list box LstSports. You can add labels and other decoration as desired.
3. On the Form menu, choose View Code. This opens the Script editor. From the Script menu, choose Event and then add the Open event. **Fig 5** is the code. Note that you should replace the database filename with the actual location of the data on your system if you use this example. Note also that you can use the Run option in the Script editor to find syntax errors before you save the code. The Run option does not actually run the code as that would be far too easy. Since the script editor has no syntax highlighting and VB Script has no debugger, it can pay to enter and check chunks of code in Visual Basic or Visual Basic for Applications, beforehand.
4. Publish the form and make it the view form for contacts in this folder. This step is so far from being intuitive that I have devoted a separate panel to it (*page 299*). Now, when you open a contact in this folder, Outlook looks up the list of sports from the original database. Of course, a real-world

system could look up a far greater range of information. The important thing is to realise that this sort of link is possible.

PCW Contacts

Tim Anderson welcomes your Visual Programming tips and queries. He can be contacted at the usual PCW address or at visual@pcw.co.uk.

The microsoft.public.outlook97 newsgroup is a valuable source of help for Outlook development, as is www.Microsoft.com. Another useful site is www.Outlook.Useast.com.



Sax appeal

Sax Webster is a browser builder that is just the last word in web applications. Tim Anderson models it here for you, taking care not to neglect his widgets and tools while he's at it.

Forget laptops and mobile phones. The fashion accessory of the moment must be the personal web site. Web sites are no use unless they are visited, so why not build point-and-click access into the applications you distribute? You can do this by calling an external application like Netscape or Internet Explorer, but Sax Software lets you go one better by building a customised browser right into the application.

The Webster control is a 32-bit browser OCX that drops directly into any compatible development tool, such as Visual Basic 4.0 or Visual C++ 4.0. With the rampant growth of the internet and increasing corporate usage of intranet networks, Sax Webster has turned up at just the right moment. For example, online help might now mean dynamic information on a web site, rather than the static file shipped with an application. Another option is to direct the hapless user to a site offering further



Fig 1 All done with Webster: VB 4.0 visits the PCW home page

products and services. HTML pages can be loaded from disk as well as from the internet, so you could also use Webster as a multimedia browser.

Sax Webster is a complete application wrapped in a control. You can create a browser simply by dropping the Webster control onto a form in VB or Delphi. It claims to support HTML version 3.0, but Sax adds that, "because 3.0 is not yet defined as a standard, it may differ from what Netscape or some other 3.0 browser supports." Here is the problem with Webster and ultimately with the web itself: lack of tightly defined standards, resulting in compatibility problems. It may not matter too much, since it would be foolish to use a Webster application as a replacement for Netscape or Internet Explorer. Webster makes better sense as a tool for accessing specific web sites that are linked to the container application, so you can ensure the

Listing 1: Intercepting the mailto command

```
Private Sub Webster1_DoClickURL(SelectedURL As String, Cancel As Boolean)
If Left$(LCase$(SelectedURL), 7) = "mailto:" Then
' run MS Exchange, using file association
ShellExecute 0, "open", SelectedURL, "", "", 0
SelectedURL = ""
' stop Webster attempting to act on this command
Cancel = True
End If
End Sub
```

Listing 2: Screensaver application

This application, which toggles the screensaver on and off, needs a VB form, a button and a code module. Note that to work in Windows 3.1, the declarations will need to be adapted. Code for the form:

```
Private Sub Form_Load()
bOldActive = isActive()
If bOldActive = True Then
Command1.Caption = "Disable screen saver"
Else
Command1.Caption = "Enable screen saver"
End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
SetActive (bOldActive)
End Sub

Private Sub Command1_Click()
If isActive() = True Then
SetActive (False)
Command1.Caption = "Enable screen saver"
Else
SetActive (True)
Command1.Caption = "Disable screen saver"
End If
End Sub (continues page 285)
```

available as a 32-bit OCX, Sax has now released a 16-bit OCX as well, but nothing yet for VB 3.0 or Delphi 1.0 diehards.

Widgets for your data

Sheridan's Data Widgets has long been one of the most popular Visual Basic additions, particularly since the VB 3.0 controls in VB 4.0 are better, but still leave room for third-party enhancements. Version 2.0 brings the expected

conversion to 16- and 32-bit OCX format, but with enhancements. Sheridan has taken the opportunity to restructure the data widgets using objects and collections, bringing it into line with other programmable OLE objects. This makes for more logical code and increases the programmer's control, the disadvantage being that code which worked with Data Widgets 1.0 will have to be extensively rewritten. For example, to put a button in a DataGrid cell in version 1.0 used a ColBtn property: `SSDBGrid1.ColBtn(2) = True` which in version 2.0 becomes: `SSDBGrid1.Columns(2).Style = 1 ' edit button.`

The actual Data Widgets controls are the same six as before: Data Grid, Data Combo, Data Dropdown, Data OptionSet, Data Command and the Enhanced Data Control. All are useful but the Data Grid is the reason people buy this package. Its neatest trick is to link with a Data DropDown so that users can click on a grid cell and select values from a dropdown list bound to a field in another table (Fig 3).

compatibility of those particular pages. Some problems can also be overcome by writing code to intercept Webster events. For example, Webster does not support the mailto command that HTML uses to initiate an email message. The VB 4.0 code in Listing 1 will intercept mailto and call whatever application is associated with that command in the Windows 95 registry.

Another useful feature is the GetContent method, which lets you read all or part of an HTML page into a variable. Initially only

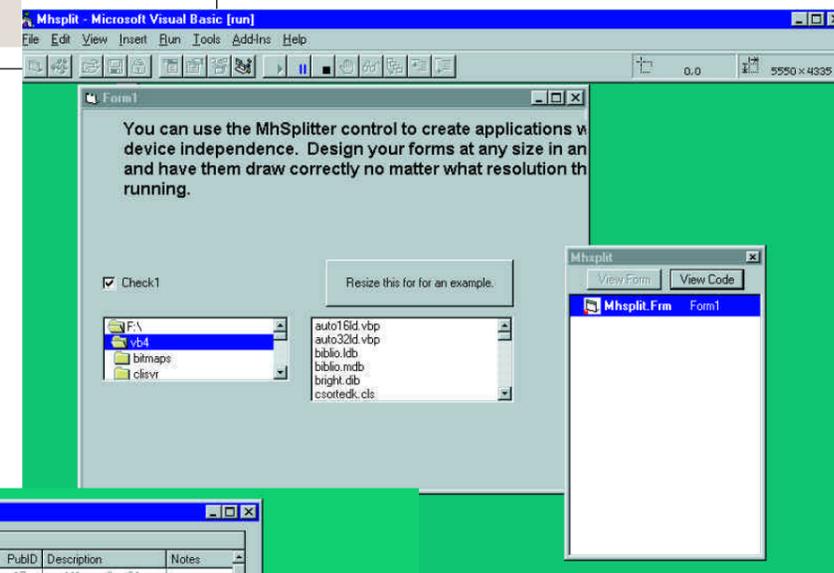


Fig 2 (above) The MhSplit control from OLE Tools attempting resolution independence. Unfortunately, this text box does not always get resized correctly...

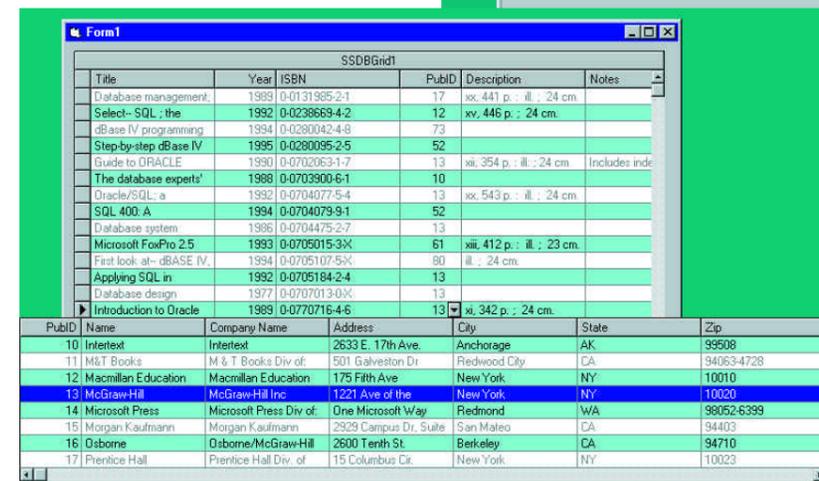


Fig 3 (left) Using a data grid and a data dropdown. Clicking the PubID column drops down the publisher table, so you can see the full details when choosing the ID

Do you need Data Widgets? It depends entirely on how you prefer to program. If you make extensive use of bound controls, this bundle is all-but indispensable, particularly if a data grid is a key part of the user interface. The data control in VB 4.0 is not compromised in the same way as VB 3.0's effort, so this is a perfectly sound approach. The cautionary note is that large OCX controls like these cause substantially slower loading of your VB application, and that grids are often not the best way to present data to the user. Finally, the Data Grid also works well as an unbound virtual list control, a further enticement which may sway doubters.

OLE tools

Microhelp's OLE tools may have up-to-date OCX technology, yet this package conveys a dated impression. The main reason is that apart from their OCX conversion, many of the controls are little changed from earlier versions, right down to their description in the manual and the clunky example applications. OLE tools also slipped up during review when one of the genuinely new items, **MhSubClass**, failed to deliver. This is a message-trapping control that can catch Windows API messages and either kill them, or respond with a custom event and then pass them on. **MhSubClass** is fine for some purposes, for example if you want to inspect **WM_MENUSELECT** messages in order to provide a help text as the mouse runs down a menu. But a common requirement is to trap a message and then write code to determine whether to kill it or pass it on. **MhSubClass** cannot do this, since the fate of the message has to be determined before the VB event is triggered. Rivals such as the MessageBlaster OCX have no such handicap.

Never mind the quality. With 54 separate controls, the bundle still rates as good value. **MhCalendar** is a data-aware calendar control. **MhSplitter** allows you to build resolution-independence into interfaces by automatically resizing controls within the container, albeit rather slowly (Fig 2). **MhReallInput** is a text box that improves on VB's masked edit control for working with real or currency values. And so it goes on, providing something of value for most VB projects.

Microhelp supplies two versions of these tools. OLE tools has 16- and 32-bit OCXs, while VB tools stays with the old VBX

Listing 2 (continued from page 283)

Code for the module:

```
Option Explicit
Global bOldActive As Boolean
Declare Function SystemParametersInfo Lib "user32" Alias
"SystemParametersInfoA" (ByVal uAction As Long, ByVal uParam As
Long, lpvParam As Long, ByVal fuWinIni As Long) As Long
Public Const SPI_GETSCREENSAVEACTIVE = 16
Public Const SPI_SETSCREENSAVEACTIVE = 17

Function isActive() As Boolean

Dim lRetVal As Long
Dim pvParam As Long

lRetVal = SystemParametersInfo(SPI_GETSCREENSAVEACTIVE, 0,
pvParam, 0)

If lRetVal = False Then
MsgBox "Call to SystemParametersInfo failed"
isActive = False
Exit Function
End If

If pvParam = False Then
isActive = False
Else
isActive = True
End If

End Function

Sub SetActive(bActive As Boolean)
Dim lRetVal As Long
Dim pvParam As Long

lRetVal =
SystemParametersInfo(SPI_SETSCREENSAVEACTIVE, bActive,
ByVal pvParam, 0)

If lRetVal = False Then
MsgBox "Call to SystemParametersInfo failed"
End If

End Sub
```

format. There are differences between the two. For example, the inadequate **MhSubClass** is OCX-only, while the clever **MhOutOfBounds** universal data binding control is VBX-only. Finally, VB tools used to come with a version of Farpoint's Grid control, but that has now been dropped.

Hacking the system in Windows 95

Mark Horton writes: "I've just bought a new system with Windows 95 and VB 4.0. My

computer has a Win/TV card, and I wanted to write a program that would turn the screensaver off and on without having to go into the display properties tab. How or where can I find out about the API calls necessary to change the screensaver settings? Is there a book on the market which describes all the Win32 (and/or Win16) API calls?"

Windows 3.1 introduced a handy function called **SystemParametersInfo**.

This reads or sets numerous system parameters including the screensaver settings. Listing 2 (pp283/285) shows a small VB application for Windows 95 which toggles the screensaver on and off. The two key functions, **IsActive** and **SetActive**, work by calling **SystemParametersInfo**. The application checks the current state of the screensaver on loading, so that it can be restored on exit.

Another possibility is for your application to disable the screensaver whenever it has the focus. Windows activates the screensaver by sending a **WM_SYSCOMMAND** message with **wParam** set to **SC_SCREENSAVE**. By intercepting and killing this message, you prevent the screensaver from kicking in.

Delphi programmers can trap messages easily, but VB users will need an add-on like the MessageBlaster OCX.

Many problems like this can only be solved using the Windows API. That in turn means having a good API reference, and

the starting point is the Windows SDK help file (Fig 4) called WIN31WH.HLP for Windows 3.1 and WIN32.HLP for 32-bit Windows. Surprisingly, Visual Basic 4.0 comes with declarations for the 32-bit API but not the 20Mb help file. An alternative is

Daniel Appleman's book, *VB Programmer's Guide to the Windows API*, which provides what is needed for Windows 3.1 and is to be updated for Win32.

Tips for Visual Programming

- Speed VBs load time and slim your applications by stripping down **AUTOLOAD.MAK** (VB3) or **AUTO32LD.VBP** (VB4) to include only controls and references essential to every project.

- Avoid **Dim iA, iB as Integer**. This code declares **iA** as a variant. Instead, use **Dim iA as Integer, iB as Integer**.
- In VB4, disable **Compile on Demand** (in Tools - Options - Advanced) to have the compiler check for syntax errors before a project runs.
- Your Delphi application can easily check for command-line parameters. **ParamCount** returns the number of parameters; **ParamStr(0)** returns the path and filename of the application, and **ParamStr(n)** returns the nth parameter up to **ParamCount**. (Listing 3)
- If you are adding lines to a string control like a listbox or memo, or an outline component, use **BeginUpdate** to increase performance by preventing screen updates. (Listing 4)

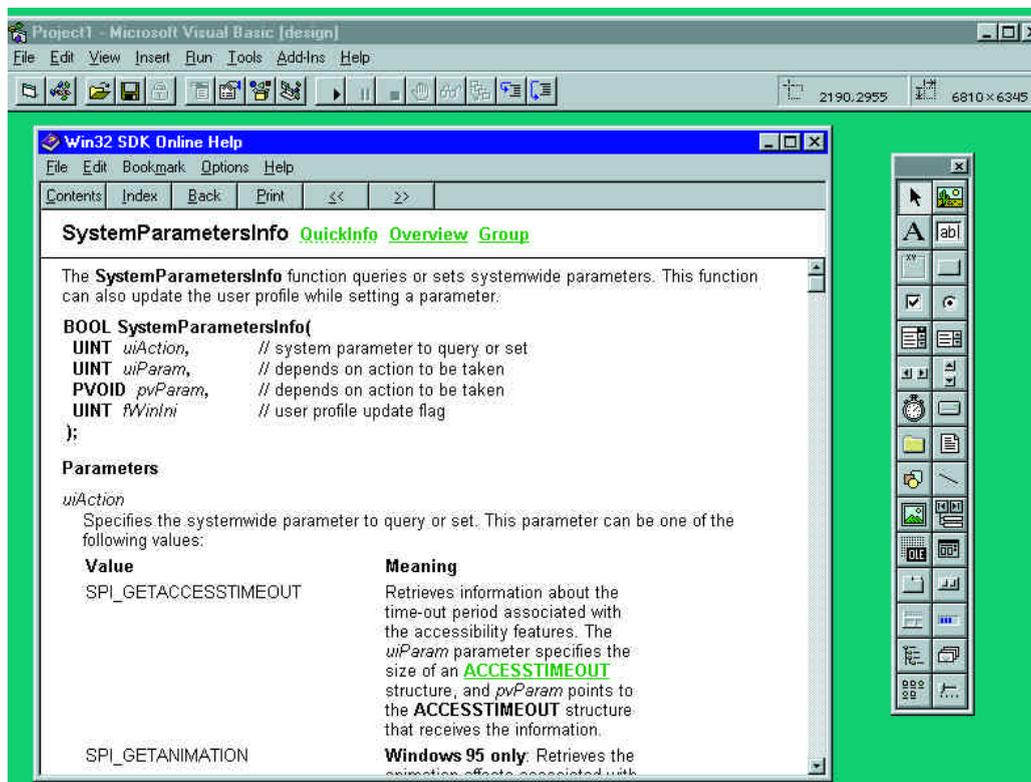


Fig 4 Although aimed at C/C++ developers, the Win32 SDK is an essential reference for Visual Basic developers. So why is this help file not supplied with Visual Basic 4.0?

Listing 3: ParamCount

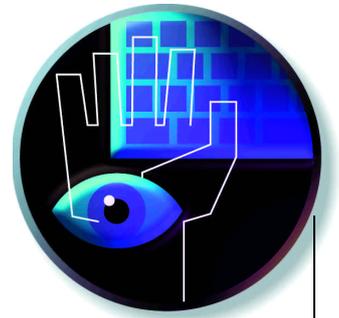
```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;
begin
  for i := 0 to ParamCount do
    MessageDlg(ParamStr(i), mtInformation,
      [mbOk], 0);
end;
```

Listing 4: BeginUpdate

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  listbox1.items.beginupdate;
  listbox1.items.add('One item');
  listbox1.items.add('another item');
  listbox1.items.endupdate;
end;
```

PCW Contacts

Tim Anderson eagerly awaits your comments, queries and tips, either at the usual PCW address or by email at visual@pcw.co.uk. *Visual Basic Programmer's Guide to the Windows API* by Daniel Appleman (Ziff-Davis Press, £33.02) **Computer Manuals** 0121 706 6000 **Sax Webster** £110 (plus VAT) **Data Widgets 2.0** is £99 (plus VAT) **OLE Tools** is 149.00 plus VAT and **VB Tools** £99 (plus VAT) from **Contemporary Software** 01727 811999



Open and shut case

Office 97... just more of the same and not worth the upgrade? To casual users, maybe; but for developers, it offers a more open programming environment. Tim Anderson explains.

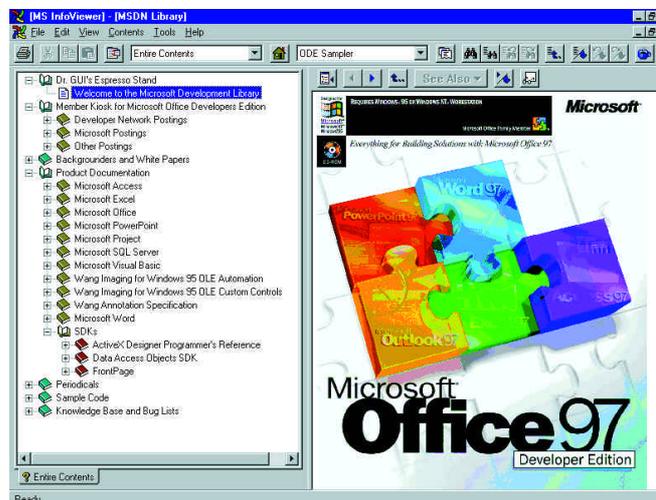
I have heard muttering to the effect that Office 97 is not much different from Office 95, or even Office 4.x. From some perspectives, that is correct. Word looks similar, Excel looks similar and all the casual user will notice at first is the Office Assistant (fantastic or horrific, according to taste) and a new, flatter, look to the toolbars.

But developers should welcome Office 97 with open arms. It is even worth explaining to the users why they really should upgrade, even if the animated Clippit is not their cup of paperclips. The reason is Visual Basic for Applications combined with the updated Office object model, all of which is exposed for programming. In most cases, equipping an Office user with suitable templates and macros soon pays for itself in increased productivity.

To do these wonderful things, you need appropriate resources. This might or might not include the Office 97 Developer Edition,

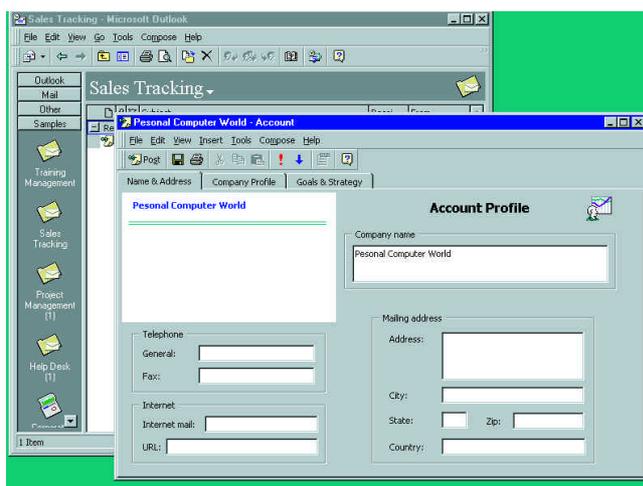
Microsoft's one-stop solution. What you get is the Office Professional CD, plus a further CD of developer tools. There are also two books: the *Office 97 Visual Basic Programmer's Guide* and *Building Applications with Access 97*, along with a booklet of Object Model charts.

Before getting too excited, though, it is worth recalling that the original Office



Left The Office 97 Developer edition is great if you need Access runtime, but otherwise it is not essential

Below, left You can develop applications in Microsoft Outlook by creating custom forms driven by VB Script. Nearly wonderful, but not quite there yet



Development Kit CD used to be more or less given away by Microsoft, presumably on the grounds that it pays to have people develop Office solutions, since in order to deploy them a copy of Office must be purchased for each installation.

Another noteworthy detail is that the books in the Developer Edition are available separately from Microsoft Press. A third

observation is that not all the documentation you might need is actually included, neither on paper nor online. A notable example is *Building Microsoft Outlook 97 Applications*, which would be particularly valuable as Outlook is brand new. Another noteworthy example is the Office 97 Resource Kit: the version on the CD is for Office 95, or at least it is in my US shrinkwrap copy. The resource kit is aimed at network administrators but contains useful information for developers as well.

The most essential developer reference, the VBA reference for each application, is actually on the Office 97 Professional CD so it is questionable whether the Developer Edition is worth having. Most of the material can also be found on the Microsoft web site, often in updated form. There is only one convincing reason for buying this product: to obtain the runtime version of Access 97. This gives you a licence to deploy Access applications royalty-free, and could soon pay for itself. If you don't need it, just subscribe to MSDN (or buy a library

Fig 1 Using the clipboard

```
Dim cr As String
cr = Chr$(13) & Chr$(10)
RichTextBox1.Text = "This is a picture" + cr
RichTextBox1.SelectStart = Len(RichTextBox1.Text)
RichTextBox1.SetFocus
Clipboard.Clear
Clipboard.SetData Image1.Picture, vbCFBitmap
SendKeys "^v"
```

controls come digitally signed with CAB versions included for web distribution. Active Threed offers seven controls which are intended as plug-in replacements for the standard Windows items like command buttons and check boxes, but with extra features. These include marquee captions which

since there is no InsertPicture method. You can insert OLE objects, though, using the OLEObjects collection. For example, this code inserts a line of text and a picture:

```
RichTextBox1.Text = "This is a picture"
RichTextBox1.OLEObjects.Add , , "c:\test.bmp"
```

Unfortunately, this can have unpredictable results depending on how OLE file associations are set up in the registry. There is also an overhead involved with OLE which makes the rich text box update rather slowly when an object is inserted. A safer approach would be to use the clipboard. The example in Fig 1 and the Delphi one (Fig 2) assume you have placed the picture you want to insert into an invisible image control on a form. The final

CD from time to time) and buy the books you really need from a bookshop.

Outlook: nearly great
Is it a Personal Information Manager, or an email client, or maybe groupware? Outlook is ambitious, and nearly the foundation of a complete Office solution. For example, it should be possible, with a bit of customisation, to right-click a contact name and open up a customer's order history or an index of previous correspondence.

There are two snags, though. One is that Outlook has VB Script but not yet Visual Basic for Applications. The second is that you need Exchange Server to do anything serious with Outlook over a network, like sharing an address book or viewing other people's calendars. In fact, Outlook without Exchange Server is less capable than the old Schedule, a fact which has not gone down well with small businesses running peer-to-peer networks. Exchange Server needs Windows NT, is priced for the Enterprise market and needs client licences, too, which makes Outlook far less attractive. Incidentally, if you decide to get going with Outlook development, a trip to Microsoft's web site is essential. Documentation and numerous sample applications are available for free download.

Sheridan's Active Threed
Sheridan products now come on a Toolkit CD containing all the company's developer tools. You can install demonstration versions of any tool, or full versions where you have the right key code. For instance, if you purchase Active Threed you get the code for this product along with the CD. There is no manual, the lame excuse being that Sheridan wanted to check the printed manual against the release code. A voucher lets you obtain it at nominal cost. But the manual aside, the all-in-one CD is a great idea. Another plus is that the ActiveX

blink, scroll, slide and bounce, plus animated pictures created with a sequence of bitmaps. There is also a splitter control which lets you create windows with resizable panes. The controls are 32-bit only; not even Visual Basic 4.0 16-bit is supported. Packages like this cause me to hesitate since many VB applications are slow enough without the additional weight of controls which aren't strictly necessary.

Two things make ActiveThreed worth a second look, though. Firstly, the SSplitter control is well implemented and provides a feature which has become something of a Windows standard. Secondly, the SSRibbon control allows you to create toolbar icons with an active border, as seen in Office 97 and Internet Explorer 3.0. I was also glad to find that Delphi samples had been included.

Rich Text in Delphi and VB
Dr Francis Burton asks: "I want to be able to alternate graphics and text in a scrollable window, with the ability to cut and paste text (and possibly bitmaps/metafiles, too). New text and graphics are appended to the end of the window. Do you know of any controls, either Visual Basic or Borland Delphi, which implement scrollable graphics/text windows?"

If you are working in Windows 95 or NT, the standard rich text control can display formatted text and graphics. It is easy to miss this functionality in Visual Basic,

Fig 2 Using WPTools

```
var
lpzCR: pchar;
lpzText: pchar;

begin
lpzText := stralloc(256);
lpzCR := stralloc(3);

try
strcpy(lpzCR,chr(13));
strcat(lpzCR, chr(10));

RichText.clear;
RichText.Font.Name := 'Arial';
RichText.Font.Size := 24;

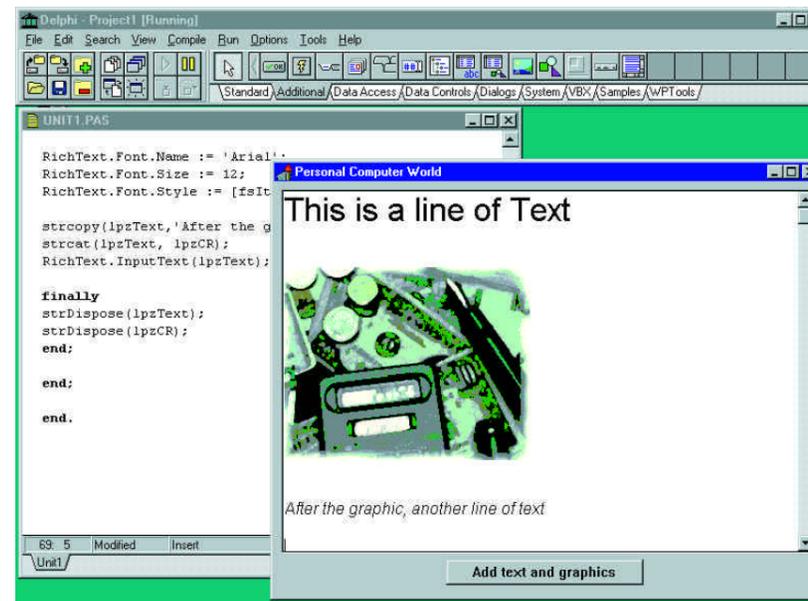
strcpy(lpzText,'This is a line of Text');
strcat(lpzText, lpzCR);

RichText.InputText(lpzText);

RichText.PicInsert( image1.picture, 0,0 );
RichText.InputText(lpzCR);
RichText.Font.Name := 'Arial';
RichText.Font.Size := 12;
RichText.Font.Style := [fsItalic];

strcpy(lpzText,'After the graphic,
another line of text');
strcat(lpzText, lpzCR);
RichText.InputText(lpzText);

finally
strDispose(lpzText);
strDispose(lpzCR);
end;
```



Even Delphi 1.0 is able to display text and graphics within a scrolling document using WPTools (illustrated here), or a component such as Visual Writer

SendKeys statement simulates the CTRL-V keypress which pastes from the clipboard at the insertion point.

For some reason, Delphi's equivalent rich text control does not support graphics. It is odd, since both use the same underlying common control, and it is likely that careful investigation of the Visual Component Library would reveal a way to overcome the problem by creating a new component which exposes more of the features in the rich text control. Or, you could use VisualWriter, an OCX and VBX control which does support graphics. Better still, use a native Delphi component like WPTools (it's shareware but works well). The WPRichText control, part of WPTools, has a PicInsert method which lets you insert a picture. It also support fonts and styles so you could implement scrollable text and graphics as required. The main snags with WPTools are its uneven documentation, and extensive use of pointers which can be error-prone. Fig 2 shows example code using WPTools.

String along with SQL
Michael O'Reilly writes: "I have been following your excellent VB tutorial, and while using some of the code given in the February issue article I encountered a problem I can't solve. In the example given, you take a string from a text box and use it in an SQL query. How do you do the same

a string like this:

```
sSql = "select * from members where
members.surname = " &
str$(myID)
```

The following question comes from Andrew Shaw: "I need a lot more input boxes than the PCW Sports Club uses and want to implement some kind of counter/loop to run through the DisplayPerson code. I want to avoid:

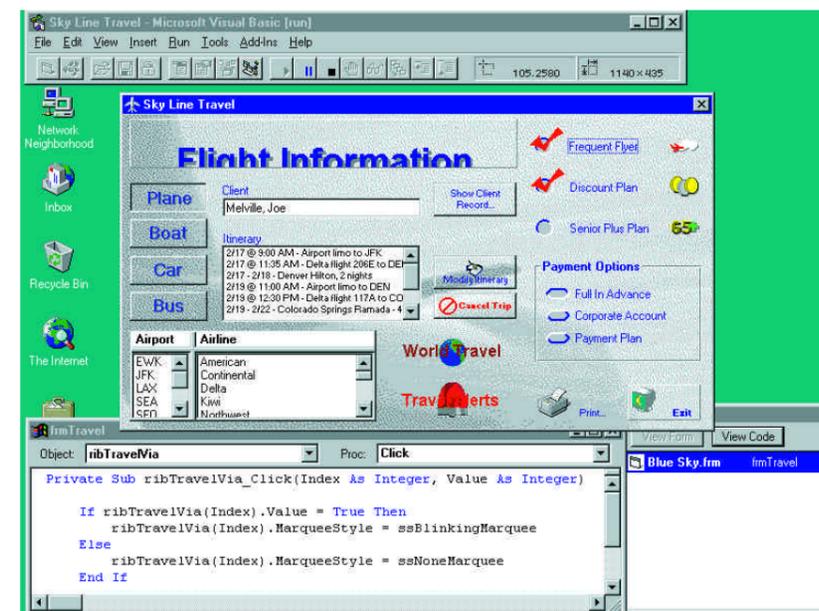
```
txtForename = CurrPerson.Forename
txtSurname = CurrPerson.Surname
```

"I tried an array of text boxes and a laborious trawl through the manual, with no success. I want to try something like:

```
txtInput(Counter) =
CurrPerson.Counter
```

"What should the CurrPerson.Counter part look like?"

Andrew's idea is to write a loop that



A printed picture does no justice to this Active Threed form, which is crawling with animation. Note the split window at bottom left — a genuinely useful feature

with a number from a text box?"

When you create an SQL string for querying a database, you use a different technique according to whether the field is character or numeric. If it is the former, the value must be in single quotation marks, as in:

```
sSql = "select * from
members where
members.surname = " &
mySurname & ""
```

Admittedly this looks ugly, but it works well. If the field is numeric, then the single quotation marks must not be used. All you need to do is convert the numeric value to

iterates through all the fields of a particular record, filling text boxes with the values along the way. This can be done as follows:

```
Dim iCountvar As Integer
For iCountvar = 0 To
(ds.Fields.Count - 1)
Label1(iCountvar).Caption =
ds.Fields(iCountvar).Name
Text1(iCountvar).Text = "" &
ds.Fields(iCountvar).Value
Next
```

The trick is to get at the Fields collection of a Recordset object. You could make the routine even more flexible by creating the necessary labels and text boxes at runtime.

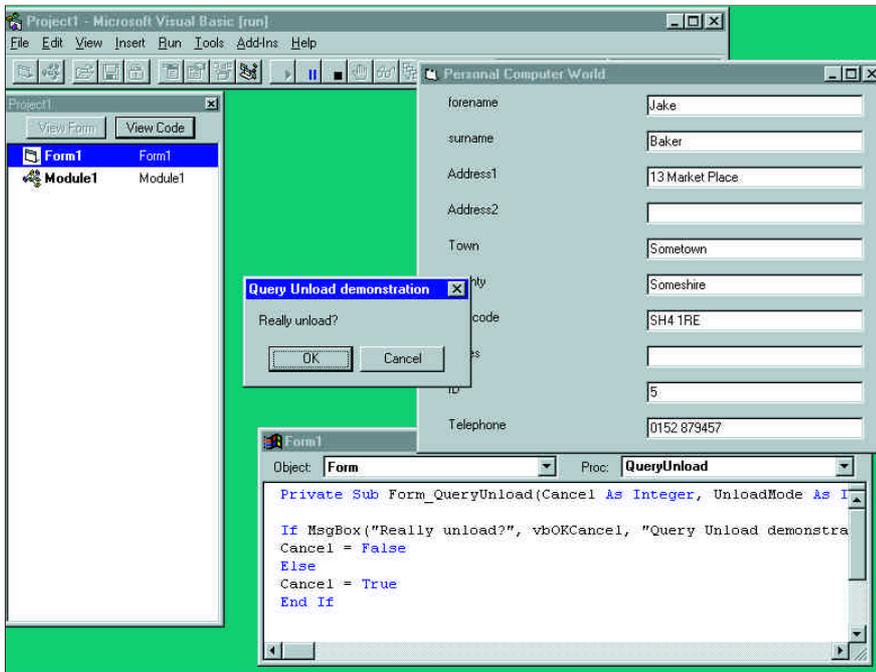
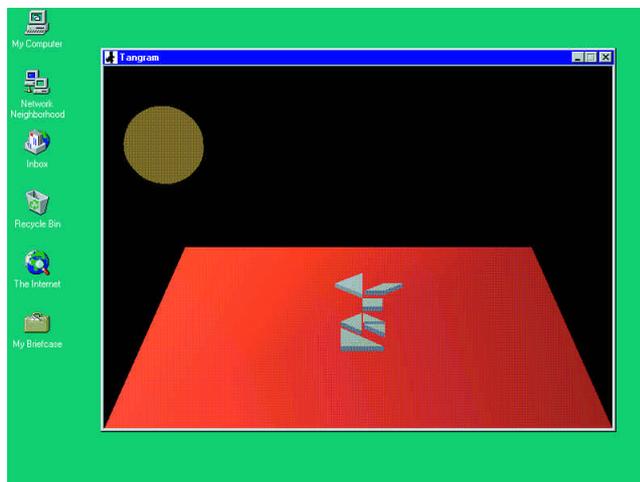


Fig 3 (above) Using the QueryUnload event to confirm a close decision

Left (see "Inside Com")

This application from Inside COM demonstrates aggregation, containment, and interchangeable components. The Tangram pieces look like a rabbit... allegedly



The main value of a routine like this is in applications where the number and type of fields in the recordset may vary at runtime. For example, you could let the user choose which fields they wanted to view, build an SQL string to return just those fields, and display them using the procedure described.

Where's the tab strip?

Phil Richard asks: "In a recent column, you mentioned a tab strip option in VB4 which I would like to use; either TabStrip or SSTab. Neither seem to be included in my installation of Standard Edition VB4, which I purchased as an upgrade. I have, however, found TABCTL32.OCX from the Sheridan web site. Is there a way of registering it, as it appears as not found when trying to load the PCWClub project."

Unfortunately Phil is correct, and the Tab custom controls are only present in the

Professional edition of Visual Basic. While a lot can be done with the Standard version, it is severely restricted both in its use of the JET database and in the number of custom controls supplied. It is also cheap, and my guess is that Microsoft views it as an introductory, learning product rather than a real development tool.

With TABCTL32.OCX, most third-party OCX vendors allow free distribution of its controls. So in order to make some sales, use of an OCX in developing an application is allowed only if you have purchased the control. When you do buy it, you get either a .LIC file or special registry entries that allow you to use it for development.

How do I cancel?

Ammar EL-Hassan has this query: "I am trying to add a facility to enable the user of my VB application to CANCEL the

operation which Closes his form. The user clicks the control box in the top left corner of the form. They then click the Close option, which unloads the form (application dead!). How can I interrupt this to enable the user to cancel after selecting Close?"

VB forms have a QueryUnload event for this purpose (Fig 3). Use code like this:

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
  If MsgBox("Really unload", vbOKCancel) = vbOK Then
    Cancel = False
  Else
    Cancel = True
  End If
End Sub
```

You can even discover why the form is trying to close, by inspecting the UnloadMode parameter. If it is vbAppWindows, then the user is trying to close down Windows.

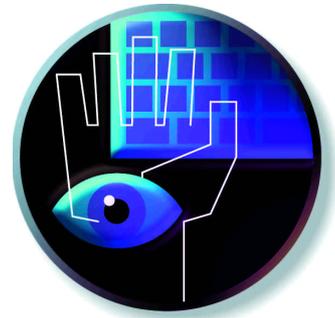
Inside COM by Dale Rogerson

Inside COM is a book that takes you step by step through the mysteries of COM interfaces, reference counting, globally unique identifiers, containment, aggregation and automation. All the examples are in C++ but the author has avoided Windows-specific code where possible. The strength of the book is that it is about COM rather than OLE or ActiveX technologies which are based on COM, so it does a good job of explaining what COM is and how it works. Of course, the impressive thing about tools like Visual Basic and Delphi is that you can use COM without needing to understand much about it. When it comes to advanced development or troubleshooting, though, a book like this provides an invaluable background.

PCW Contacts

Tim Anderson welcomes your Visual Programming comments, queries and tips. Contact him at the usual PCW address or email visual@pcw.vnu.co.uk.

Office Developer Edition is £639 (ex VAT). Upgrades from Office 97 Professional are £215 (ex VAT). Contact **Microsoft** 0345 002000
Office 97 Visual Basic Programmer's Guide available separately at £32.49 from **Computer Manuals** 0121 706 6000
Sheridan Active Thread £99 (ex VAT) from **Contemporary Software** 01344 873434
Inside COM (Microsoft Press) is £32.99 from **Computer Manuals** 0121 706 6000.
Visual Writer is £195 (ex VAT) from **Visual Components** 01892 834343
WP Tools is shareware. Contact Julian Ziersch 100744.2101@compuserve.com



Clean-up campaign

Tim Anderson wrestles with the registry in an attempt to unscramble his settings, tries to get Access from Delphi, and plays Sherlock Holmes to detect which applications he has running.

Imagine you have paid a four-figure sum for a top-of-the-range client-server development system. One day you open up the development environment and the splash screen declares it to be the entry-level hobbyist version. Next, you open the application you are working on to be informed that you are not licensed to use some of its components. Sighing, you reinstall the product from CD but it does not fix the problem.

Sounds fun? This is exactly what can happen with Visual Basic 4.0. The reason, as you will have guessed, is that both VB itself and the many OCX controls which come with it depend on numerous registry settings. If the registry gets scrambled, this is the kind of thing that can happen.

The good news is that Microsoft's web site has a fix. Article Q149619 is entitled "Visual Basic displays incorrect splash screen", although the splash screen is the least of your problems. It is not such good news though. The official fix goes as follows:

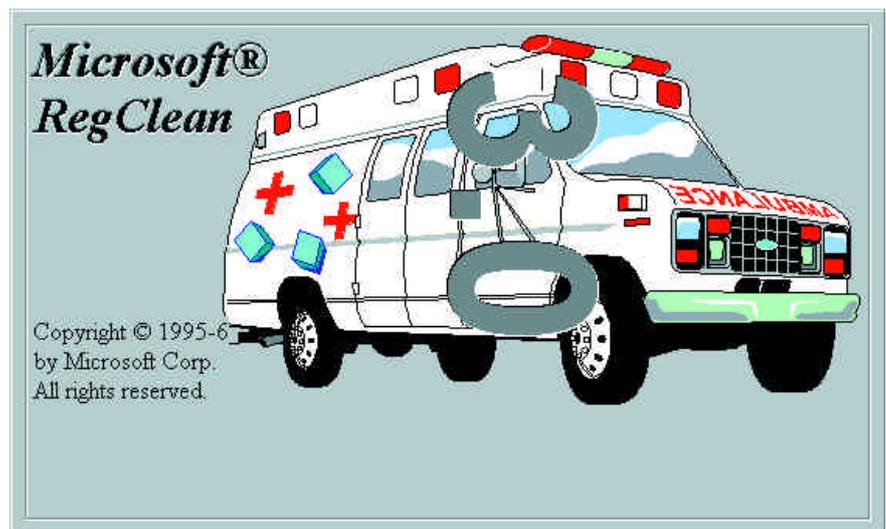
1. Using a registry editor, delete the HKEY_CLASSES_ROOT\LICENSES key.
2. Run Regclean.exe and delete all *.OCX and *.OCA files.
3. Delete OLEPRO32.DLL.
4. Restart Windows and reinstall Visual Basic.

Is this a good fix? Well, it's better than destroying your hard disk with a sledgehammer, but not much. As a developer, you will know that those .OCX and .OCA files represent most of the ActiveX controls on your system. An OCA file, by the way, is an OLE-type library created by VB when you first load an OCX. And ActiveX, says Microsoft, is becoming the foundation of Windows. Then there is

the license key to think about, which any number of applications may be using. A clue to the extent of this devastation is given in the note at the end of the fix. "Reinstall third party custom controls," it says, "and any software that may use the registry to store licensing information."

As for Regclean, a utility that comes with VB, I have come to mistrust it deeply. In a

An added twist is that the software industry now gives huge distribution to beta versions, via demonstration CDs and over the web. We are all encouraged to spend our time installing trial software, often laden with ActiveX elements, and probably fixed to stop working after a certain date. Frankly, the registry stands no chance of staying clean in these circumstances. Naturally, it is



Microsoft's RegClean 3.0: proceed at your own risk

misguided moment I ran the latest version 3.0 which you can download from www.microsoft.com. The idea was to fix the annoying messages VB gives you when something is awry in the registry: "Object server not correctly registered". To my great amusement, the end result was worse. Post-Regclean, VB gave me this inspiring piece of technical information 148 times before it would open the Custom Controls dialog. At times like that, you reach for your registry backup with relief.

This problem is not going to go away.

not just developers who install all this stuff, but clients and users as well. Any application that uses standard Microsoft or third-party ActiveX controls or servers may find the ground sweetly removed from under its feet. In the meantime, here are my tips for avoiding registry hell:

1. Check your registry backup procedures.
2. Press Microsoft to come up with proper registry management tools, rather than these draconian "delete everything and reinstall" solutions.
3. Install beta software on a machine



Borland's Developer Conference CD has some great resources, but why pay when you can visit the web site?

Thanks to the popularity of Microsoft Office Professional and Visual Basic, desktop data is frequently stored in Access MDB files. This creates a problem for other applications which need to get at the data, especially since Microsoft has never documented the structure of an MDB. In any case, the format changes with each new release of Access. Borland's Database Engine can only get at an MDB through ODBC, which is the method Guy has tried. Sadly, the BDE is not at its best with ODBC, and Microsoft's ODBC drivers for Access are nothing special either.

The situation is complicated by the inclusion of ODBC drivers with Microsoft Office, that are designed only to work with Office applications. This might well cause the error Guy is seeing. It is important to get hold of the separate ODBC desktop driver pack, for example from the Microsoft Developer Network CDs, but even then it might not work. It needs the right combination of DLLs, registry entries and even INI files to work as it should, and one or other can easily get corrupted. Sometimes the only solution is to remove

dedicated to that purpose. Do not install it on a system used for real work.
 4. Persuade your users to adopt the same policy.
 5. So you only have one PC? Well, you have been warned.

Delphi

Borland's Conference CD

Borland developers who look with envy at the Microsoft Developer Network CDs, stuffed with documentation and tips, will be interested in the recently issued Developer Conference CD. At first glance it looks great, with technical papers and example code covering many real-world problems. The two most prominent products are Delphi and C++ 5.0. The catch is that what you get depends on whether individual speakers at the 1996 Borland conference bothered to send in their notes.

For example, an entry on "Client server development using Delphi and Oracle" leads to a detailed article with source code and a Powerpoint slide show, while another entitled "Rapid application with Delphi 2.0" brings up only a speaker biography. Everything is in HTML and no search program is provided, so you are left to use your own search tools. You also get a collection of patches, technical notes and demonstration versions. Overall there are plenty of good nuggets of information, but it is all rather a mish-mash and mostly available free from Borland's web site. A useful resource, but not for the price Borland is asking.

Mixing Delphi and Access

Guy Cartwright writes: "I'm led to believe that, using Borland's Database Engine, I can access data stored in a Microsoft Access database. I've followed the procedure in a book and created an alias called TstAccess, but I get the message 'Application is not enabled for use with this driver. Alias: TstAccess'. I've trawled the net for an answer but to no avail."

Fig 1 Routine written from the DAO COM interface

```
var
  sSql: string;
  dbEngine: Variant;
  db: Variant;
  snMembers: variant;

begin

  sSql := 'Select * from members order by surname;';
  dbEngine := CreateOleObject('DAO.DBEngine');
  db := dbEngine.OpenDatabase('C:\DATA\SPORTS.MDB');
  snMembers := db.OpenRecordSet(sSql, 4);
  {4 is dbOpenSnapshot}

  If not snMembers.EOF Then
  begin
    Edit1.text := snMembers.Fields['SURNAME'].Value;
  end;

  snMembers.close;
  db.close;

end;
```

Powers of detection

Once you get started with Windows programming, you soon find you need to communicate with other applications. At its simplest, for example, you might want to run the Windows calculator from a menu option in a VB application. Easily done with the Shell function but what if the Calculator is already running? In that case, you probably want to bring forward the existing instance rather than starting a new one. Here is how you can find out.

The key to detecting an application is to look for its main window. The API offers functions for listing or searching all the current windows. FindWindow takes two parameters, both null terminated strings. The first is a classname, the second the text of a window title. You can search for one or both and if it finds a matching top-level window, FindWindow returns the handle. For example:

```
hwnd = FindWindow(vbNullString, "Calculator")
If it returns 0, then Calculator is not running. Of course FindWindow must be declared, and you can copy the declaration from VB's API viewer.
```

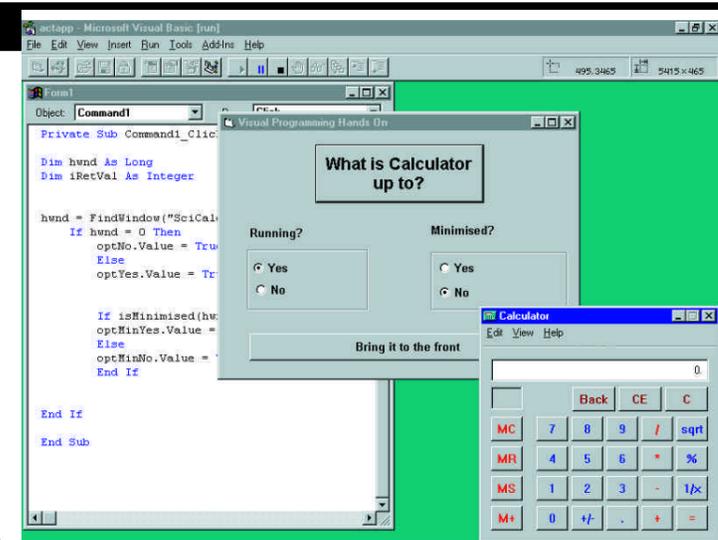
In the example above, FindWindow searched for the window title. This works fine with Calculator, although you could not be sure which calculator you were getting. It falls down with MDI applications, where a maximised document window adds its title to the main window. You might want to use the classname instead. It is not obvious what the right classname is, but there is another API function, GetClassName, which reveals all. Calculator turns out to have a classname of "SciCalc", while Word is "OpusApp". VB is "ThunderMain", and a VB application, "ThunderForm" or in version 4.0, "ThunderRTForm". Delphi applications get their classname from the name of the main application window, for example "TForm1". So the decision to look for a classname, a window title or both depends on which application you are trying to detect.

If the application is running, the next step is how to bring it forward. One possibility is the API function BringWindowToTop. For example, the following code detects Word and brings it forward if found:

```
hwnd = FindWindow("OpusApp", vbNullString)
If hwnd <> 0 Then
  BringWindowToTop (hwnd)
End if
```

The one time this will fail is if Word is running but minimised. A minimised window brought to the top is not much help. Time for another API function or two, in this case GetWindowPlacement and ShowWindow. Using the API viewer, add the declarations for the following:

```
GetWindowPlacement
ShowWindow
Type POINTAPI
Type RECT
Type WINDOWPLACEMENT
Public Const SW_SHOWMINIMIZED
```



Using API functions you can find out which other applications are running

```
Public Const SW_RESTORE
You can now discover whether a non-VB window is minimised like this:
Function isMinimised(hwnd) As Boolean

  Dim lpWnd As WINDOWPLACEMENT
  lpWnd.Length = 44 ' 22 in 16-bit Windows
  Call GetWindowPlacement(hwnd, lpWnd)

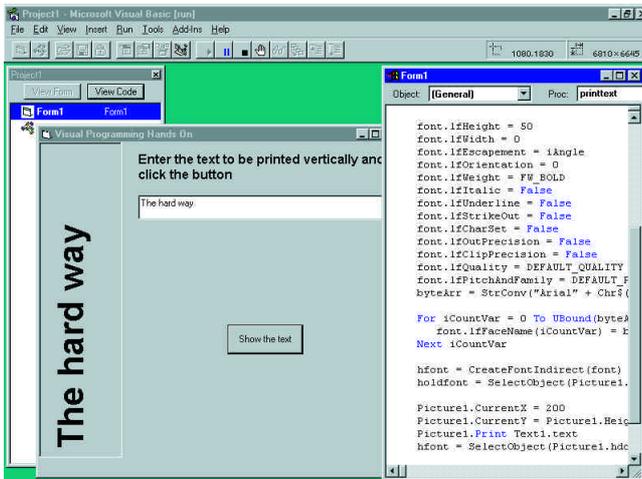
  If lpWnd.showCmd = SW_SHOWMINIMIZED Then
    isMinimised = True
  Else
    isMinimised = False
  End If

End Function
```

Now the function for bringing Word forward can be modified as follows:

```
hwnd = FindWindow("OpusApp", vbNullString)
If hwnd <> 0 Then
  If isMinimised(hwnd) Then
    iRetVal = ShowWindow(hwnd, SW_RESTORE)
  Else
    BringWindowToTop (hwnd)
  End if
End if
```

If you look up GetWindowPlacement and ShowWindow in an API reference, you will find numerous other fields and parameters that give you fine control over the results. One point to notice is that the length field of a WINDOWPLACEMENT type (or structure in C) must be set before it is passed as a parameter in GetWindowPlacement. Unfortunately VB has no SIZEOF function, so you cannot do this neatly. All you need to know for the moment is that in 16-bit Windows the magic number is 22, and in 32-bit Windows it is 44. Occasional inconveniences like this are the price you pay for avoiding the intricacies of C.



Left Vertical text the hard way, setting the font with the Windows API

both the ODBC driver and the BDE, weeding out any registry entries as well, and then to reinstall them both. Microsoft Query, which comes with Office, lets you test ODBC data sources by running queries against them.

There is another option if you are running Windows 95 or NT. Microsoft has created a COM interface to the JET database engine under the name Data Access Objects (DAO). It is documented and can be called from Delphi, and you can write routines like in Fig 1 (page 308). For this to work, DAO must be installed on the system, as it will be if you have Microsoft Office 95, for example.

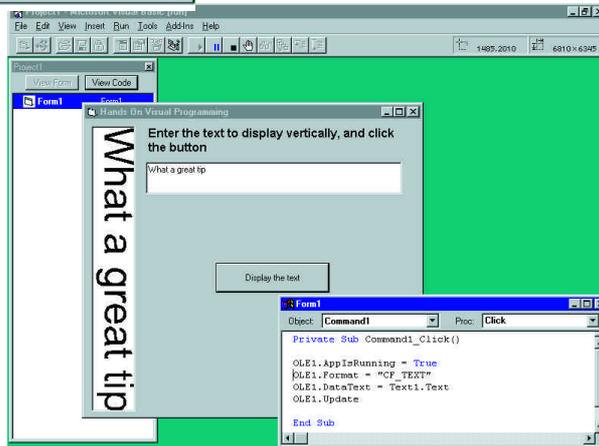
There are several other problems. Microsoft's documentation is aimed at users of Visual Basic or Visual C++, so you have to feel your way to some extent. None of Delphi's data-aware components will work. Finally, you cannot freely distribute the DAO files with a Delphi application. All but the last can be fixed by buying a third-party tool for using DAO with Delphi. Two well-known ones are Titan Access and Opus DirectAccess, while Nortech Software has a third in preparation. One of these is likely to be the smoothest route towards using Delphi with Access MDBs.

Visual Basic

Going vertical

Andy Smith asks: "How can I print vertical text in a Visual Basic application?"

There are a couple of easy solutions, and a better but more difficult one. The easy way is to use a paint program to rotate some text — Windows Paint or the shareware Paintshop Pro will do nicely — and paste it into an image control. You could even have several different messages and load them at runtime. For the best



Below Vertical text the easy way, using the OLE container and a WordArt object

performance, do not load them from disk but use invisible image controls, or the PicClip control, or the Imagelist control.

If you want to be able to specify any text you like at runtime, another possibility is to use the WordArt applet that comes with Microsoft Office or Publisher. Here's how:

1. Pop an OLE container onto a form and set it to contain a new WordArt 2.0 object.
2. Right-click the OLE container and choose Open. In the WordArt dialog, choose the text shape and font required.
3. Use code like this to update the text at runtime:

```
OLE1.AppIsRunning = True
OLE1.Format = "CF_TEXT"
OLE1.DataText = Text1.Text
OLE1.Update
```

The snags with the WordArt approach are firstly that you need the applet installed on the user's system, and secondly a little overhead thanks to OLE. If that rules it out, the heavy coder's method is to call the Windows API. Windows uses a structure called a LOGFONT to define font characteristics, including several properties not exposed by VB's Font properties. One of these is lfEscapement, which specifies the angle of the text. Assuming that the y

co-ordinates count from top to bottom, the lfEscapement field specifies the anti-clockwise angle in tenths of a degree. That means you can print diagonal text or even write a routine using a timer that would rotate text around a central point. To set a font using the API, take the following steps:

1. Declare the necessary API types, constants and functions.
2. Define the fields of a LOGFONT variable.
3. Create a logical font by calling CreateFontIndirect. This returns a handle to a font.
4. Select the font into a device context by calling SelectObject. For example, VB

Picture Boxes, Forms, and the Printer object all have hdc properties which give you a handle to the device context.

5. Print to the device context using VB's print method or API functions such as TextOut or DrawText.

6. Clean up by unselecting the font and calling DeleteObject with the font handle.

Minimal sample code for drawing vertical text in VB 4.0 is included on the

CD. Similar code works in VB 3.0 or 16-bit VB 4.0. It seems complex at first but it is the kind of code you can use again. Then again, alongside the four lines needed to automate WordArt, it does look like an argument for sticking to the easy way.

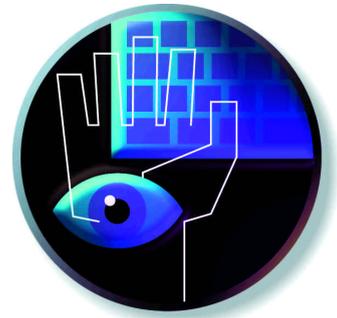
Cover CD

The MSDN starter edition for Visual Basic is on this month's cover-mounted CD-ROM. It includes 125Mb of searchable information on VB 3.0 and VB 4.0.

PCW Contacts

Tim Anderson welcomes your Visual Programming comments and tips. He can be contacted at the usual PCW address or at visual@pcw.vnu.co.uk

- Borland Developers Conference CD** £59 (plus VAT) from Borland 0800 454065
- Delphi 2 Developer's Guide** (Pacheco and Teixeira) from SAMS/Borland Press £54.99
- Opus DirectAccess** £189 (plus VAT) from QBS 0181 956 8000, www.opus.ch
- Nortech Software** is at www.wizzkids.com
- Titan Access 32** is £225 (plus VAT) from QBS 0181 956 8000, www.reggatta.com



A new Deal

Tim Anderson checks out the new Formula One spreadsheet control in the latest upgrade to the Visual Developers Suite Deal, answers VB and Delphi queries and hides a blinking caret.

Visual Components has upgraded its Visual Developers Suite Deal, a collection of ActiveX controls for Visual Basic and other ActiveX clients. These are heavyweight components, each being almost an application itself. They are supplied as both 16-bit and 32-bit OCX controls. The runtime versions can be distributed royalty-free, making the Suite excellent value if you need this kind of functionality.

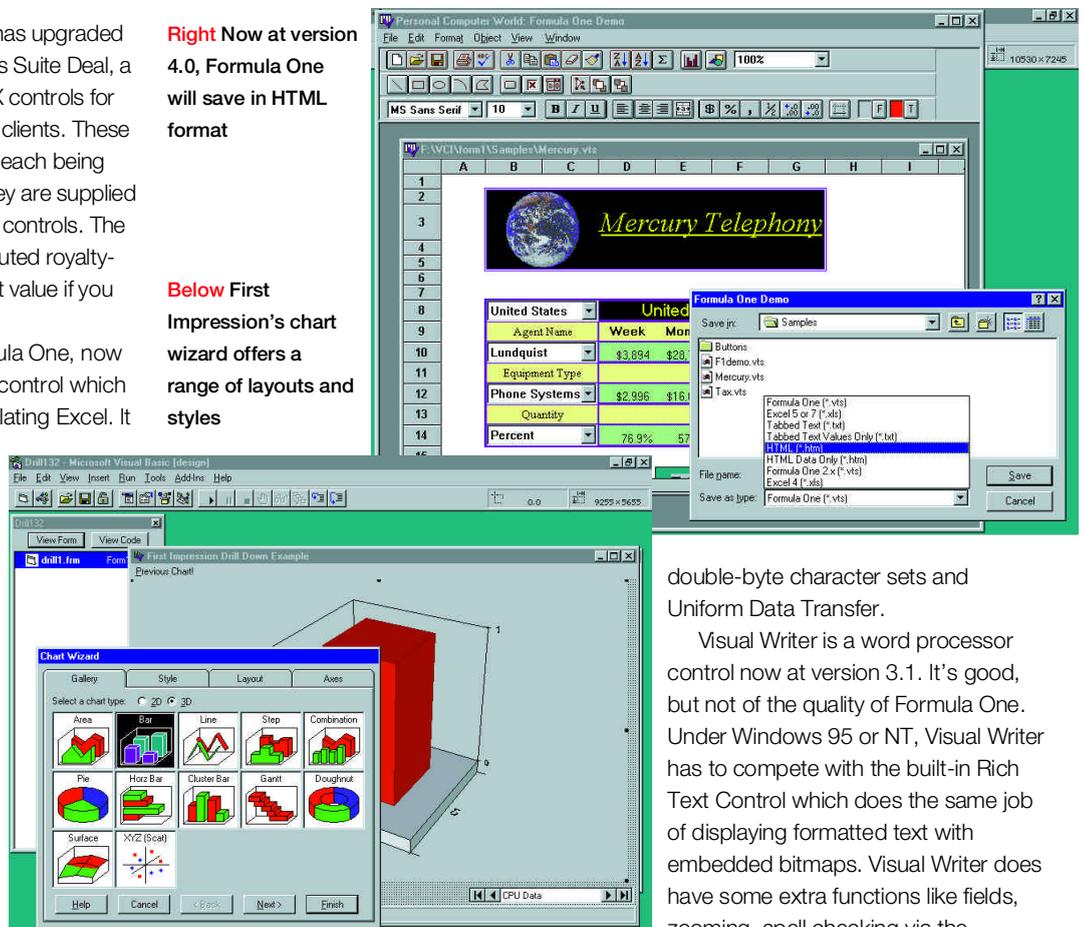
Cream of the crop is Formula One, now at version 4.0, a spreadsheet control which does a remarkable job of emulating Excel. It can read and write files in Excel format up to version 7.0, as in Office 95, but there are limitations: Formula One does not understand Excel charts or macros, for example. A large number of worksheet functions are supported, and the ability to move sheets to and from Excel is a valuable asset. Formula One has its own drawing tools and can link with First Impression, the charting control in the Suite Deal, to create charts. You can place buttons, checkboxes and drop-down listboxes on sheets.

New in version 4.0 is support for double-byte character sets, HTML export, and Uniform Data Transfer, an OLE standard which lets you drag and drop data between applications. Formula One is not a data-bound control, but it has built-in ODBC support so you can query an ODBC database and the results appear in a worksheet.

Version 4.0 includes several new ODBC functions. Another nice touch is the workbook designer, a fully-featured

Right Now at version 4.0, Formula One will save in HTML format

Below First Impression's chart wizard offers a range of layouts and styles



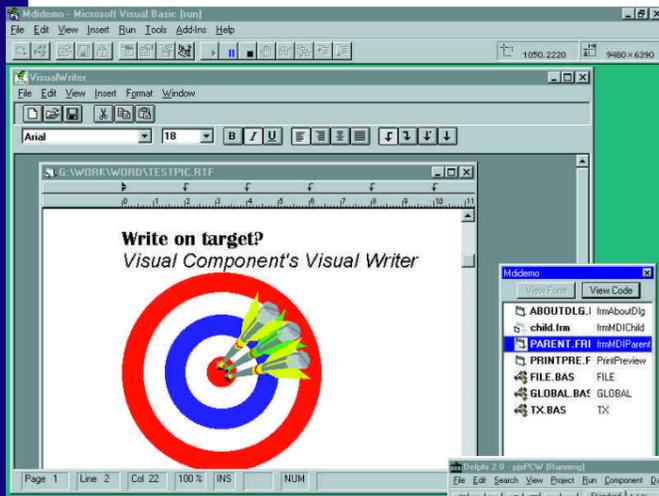
double-byte character sets and Uniform Data Transfer.

Visual Writer is a word processor control now at version 3.1. It's good, but not of the quality of Formula One. Under Windows 95 or NT, Visual Writer has to compete with the built-in Rich Text Control which does the same job of displaying formatted text with embedded bitmaps. Visual Writer does have some extra functions like fields, zooming, spell checking via the supplied Visual Speller control, print preview, and a ready-made button bar, status bar and ruler. It also has some quirks. Rich Text Format (RTF) is supported, but it prefers its own proprietary format. This is a disadvantage, especially since it will not accept .RTF as a valid format when bound to a document database. Also lacking is any kind of HTML support. For Windows 3.1 developers, though, Visual Writer or something like it is all but essential if you need to display formatted text. It's a shame the supplied 16-bit version is an

spreadsheet application which pops up on demand to enable you to create workbooks interactively.

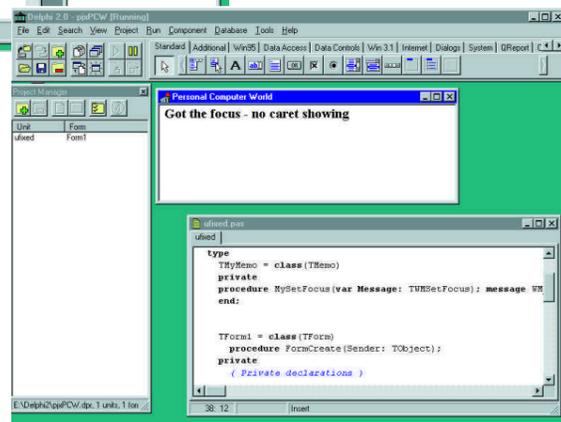
Formula One is superb and its main competition is from Excel itself. Excel is a strong development tool and its worksheets can be embedded in other applications and controlled programmatically. Excel is the more powerful, but in comparison Formula One is small, nimble and royalty-free.

First Impression, the Suite's charting component, is updated to version 2.1. Not much has changed, mainly the support for



Left Visual Writer is invaluable for 16-bit Windows but less useful in Windows 95

Below A solution to the flashing cursor problem. The TMemo object is subclassed, creating a component which is sufficiently general to be used many times over



class the TMemo object, giving it a custom message handler. You can either do this entirely within your application, or create a new custom memo component and install it on the component palette. For a one-off, the first approach is fine, the only snag being that because you create the control at runtime, you cannot place it visually or use the Object Inspector. How it works is shown in Fig 1.

Fig 1 Trap that SetFocus

- In the type section of the form unit, declare the following object:

```
TMyMemo = class(TMemo)
private
  procedure MySetFocus(var Message: TWMSSetFocus); message WM_SETFOCUS;
end;
```
- In the public declarations for TForm1, include:

```
Memo1: TMyMemo;
```
- In the implementation section, include:

```
Procedure TForm1.MySetFocus(var Message: TWMSSetFocus);
begin
  inherited; {call the default handler for this message}
  HideCaret(self.Handle); {hide the caret}
end;
```
- In the FormCreate method, include:

```
Memo1 := TMyMemo.create(self);
Memo1.Parent := self;
HideCaret(Memo1.Handle);
```

Of course, you will also want to set other properties and perhaps write event code for the memo control, all of which you can do in code. Sledgehammer to crack a nut? Maybe. But once you have learnt how to subclass Delphi controls, many other problems can be easily solved.

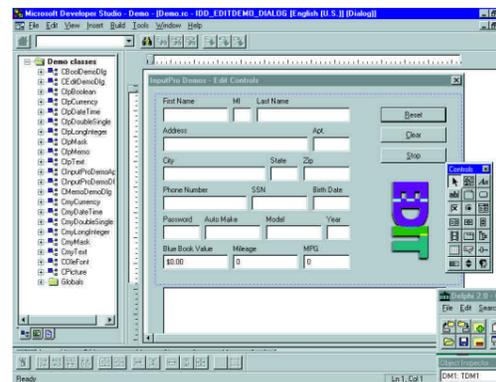
You can also build up a library of customised components which can be used many times over. For example, you could create a memo with a Boolean ShowCaret property that turns caret display on or off. In the long term, the productivity gains are enormous.

Input Pro

Once upon a time, it was Aware/VBX. FarPoint has renamed this set of data-entry controls to the more natural Input Pro. It is an unglamorous collection, but is also one of the most useful for anyone doing data entry forms in Visual Basic or other ActiveX clients. A VBX version is also supplied.

There is not much extra functionality in Input Pro, as opposed to Aware/VBX. The main difference is the move to

ActiveX. There are eight controls including currency, date and time, masked edit, and a memo control which overcomes the normal 64Kb limit. All are data-aware. The main purpose of InputPro is for validating data entry (never an easy task): its controls greatly simplify matters. For example, the DateTime control rejects invalid dates and times, can limit their range, and can auto-complete partial entries.



Above FarPoint's Input Pro gives fine control over validation during data entry

Right Behind those drag-and-drop data controls lurks the Borland Database Engine, a substantial application in its own right

Rolling your own

Reader Richard Hustwayte writes: "My project will require some databases to be made — nothing complex like client/server but simple, flat-file databases. I have looked at two versions of Delphi: the standard version (about £70) and the desktop version (about £230). The latter version is advertised as having the Borland Database Engine. What is this? And if I don't have it, am I unable to create database applications?"

All versions of Delphi come with the Borland Database Engine. This is a library of functions designed to simplify database work by acting as an intermediary between your application and the driver software which actually accesses the data. One benefit is that you can use data-aware components, so you can create simple database applications without writing any code. A number of database drivers are available for the BDE and it can also use ODBC drivers, the Windows standard for database access.

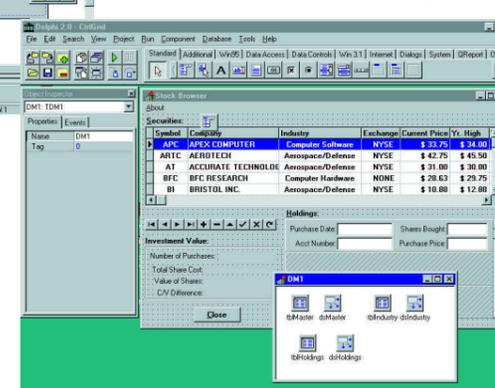
The BDE is good, but there is a cost involved. The BDE is a substantial piece of software and adds a considerable overhead

Fig 2 Pseudo-code for "polling"

```
Begin do loop
DoEvents or Sleep to allow windows to run other processes
Check InBufferCount property
If there is data, read input property and add to string buffer
Check buffer is not too full and correct if necessary
Check for time out, data complete, broken connection or other errors
End do loop
```

to your application. Additionally, not all database drivers work well with it. In particular, it is not a good match for data in Microsoft Access format. For several reasons, then, you might not want to use it.

The most efficient alternative for a simple database is to roll your own using Delphi's file functions. If that does not appeal, there are database libraries such as Sequiter's



CodePascal which provide a lightweight alternative. Then again, the BDE comes in the box and is fairly easy to use, so most Delphi developers do not look elsewhere.

Which Delphi book?

Darren Davies writes: "I'm just about to buy Delphi Developer 2.0. I was wondering if you could recommend a book to learn how to program in this language? I've had quite a bit of experience with Pascal for DOS and object-oriented Pascal for Windows, but not much with visual programming."

For someone with programming experience, a good choice is *Delphi 2.0 Unleashed* by Charles Calvert (SAMS). At 1,400 pages, it goes some way to compensating for Delphi's poor documentation.

A terminal problem

"I'm trying to get my new Terminal Program to automatically log in to a BBS. How do I get MSCOMM to wait for a login prompt before it enters the user details like Login name and Password?" asks Aaron Hodgson.

The MSCOMM custom control, which is

similar in Visual Basic 3.0 and 4.0, offers two ways of intercepting data. The first technique involves a program loop which continually checks the receive buffer, a technique called "polling". Fig 2 shows what it looks like in pseudo-code. While it is a useful technique, it is difficult to write a well-structured application if it spends much time sitting in a loop like this. Another method is preferable, which is to use the OnComm event to respond to incoming data. This event fires whenever a communication error or event occurs. You can respond with a select case statement, like that shown in Fig 3 (page 319).

Your code should respond to all the

Java books

Professional Java Fundamentals by Sly Cohen, Tom Mitchell and others

Most Java programmers are already skilled in another language: often C++. This book is aimed at that readership, providing a concise introduction to Java and focusing on its distinctive features. Beginning with a description of the Java language and object-oriented programming, it goes on to explain packages, threads and streams. Five chapters are devoted to the Abstract Window Toolkit, including a detailed explanation of various layout managers. The most advanced chapters cover networking, building libraries, implementing an application framework, and interfacing with C++.

There seem to be lots of poor Java titles around, and in contrast here is a knowledgeable and well-judged guide which complements rather than repeats what is easily obtainable online. Recommended.

Using Java (Second Edition) by Joseph Weber and others

The flash on the cover states: "Covers new JDK 1.1 features" which is a bold claim since, at the time of writing, the JDK 1.1 was still in beta. You will find some useful material on JDBC database classes and a little on remote method invocation but, of course, much of JDK 1.1 is not actually included. What you get is over 1,000 pages which take you step-by-step through Java's tools, language, classes, applets and applications, graphics and layout, security and more. There is an emphasis on Sun's tools rather than third-party contributions, although the online version includes a chapter on different development environments.

Overall, *Using Java* is a thorough guide, although at times rather ponderous and unexciting. On the CD, you get online versions of four other titles covering JavaScript, Visual J++, CGI scripting and HTML, along with additional chapters and example Java applets. As a one-stop Java reference library, this book is hard to beat.

Fig 4 WaitFor function

```
Function WaitFor(sWaitString As String, lTimeout As Long) As Integer

Dim lStartTime As Long
Dim sBuffer As String
Dim iOldThreshold as integer

lStartTime = Timer
iOldThreshold = Comm1.RThreshold
Comm1.RThreshold = 0
' prevents comEvReceive firing

Do
DoEvents ' or call Sleep API function
If Comm1.InBufferCount > 0 Then
sBuffer = sBuffer & Comm1.Input
' should check for buffer too large
End If

If InStr(sBuffer, sWaitString) > 0 Then
WaitFor = 0
Exit Do
End If

If Timer >= (lStartTime + lTimeout) Then
WaitFor = 1
' you can define constants and report errors
' using the return value
Exit Do
End If

Loop

Comm1.Rthreshold = iOldThreshold

End Function
```

Now you can write code like this:

```
If WaitFor("login: ", 60) = 0 Then
' waits for up to 60 seconds
Comm1.Output = "qix" & Chr(13)
MsgBox "Successfully posted response"
Comm1.Rthreshold = 1
' Enables comEvReceive event
Else
MsgBox "Login error"
Comm1.PortOpen = False ' Closes port
End If
```

possible events in order to trap communication errors. You also need to check that the string buffer is kept to a reasonable size. Using the CommEvent it is possible to write a reasonable communications program in Visual Basic, and there is an example called VbTerm that comes with Visual Basic.

Although the event-driven approach is better for most purposes, Aaron's particular problem can easily be solved by polling. You can write a WaitFor function that doesn't return until a particular piece of data

has been sent, or until an error has occurred. An example of this is shown in Fig 4.

Note that if you have also written code to respond to the OnComm event, you need to ensure that events of the type comEvReceive do not fire when the WaitFor function is running. You can do this by setting the RThreshold property to zero.

Finally, communications code is tricky, mainly because so many things can go wrong. At one extreme, poor lines and

dropped connections cause difficulties, while the opposite problem is data coming in so fast that some part of your software cannot keep up. If this last problem occurs, Microsoft makes two recommendations. One is to extract data immediately the OnComm event fires, without bothering to check the type (see Fig 5). Also, the MSCOMM control may not always be satisfactory and as a last resort you can call the Windows API directly. This is well covered in the first edition of Daniel Appleman's *Visual Basic Programmer's Guide to the Windows*

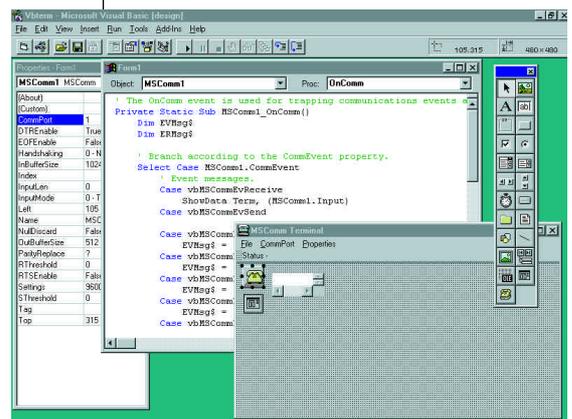
Fig 3 A select case statement

```
Select Case Comm1.CommEvent
Case comEvReceive
sBuffer = sBuffer & Comm1.Input
' or send to data processing function
Case comRxOver
MsgBox "Error: receive buffer overflow"
Case comTxFull
MsgBox "Error: transmit buffer full"
End Select
```

Fig 5 Extract data

```
Sub Comm1_OnComm ()
Static ReceiveBuffer As String
ReceiveBuffer = ReceiveBuffer &
Comm1.Input
Etc...
```

API but not in the second, 32-bit edition, although there is some material on the CD which accompanies the book.



Above The VBTerm sample comes with Visual Basic and demonstrates the use of the MSCOMM control

PCW Contacts

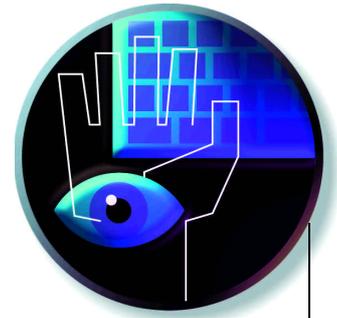
Tim Anderson welcomes your Visual Programming comments and tips. He can be contacted at the usual PCW address or at visual@pcw.vnu.co.uk

Visual Developer's Suite Deal is £235 (plus VAT) from Visual Components 01892 834343
Input Pro (FarPoint) is £105 (plus VAT) from Contemporary Software 01344 873434

Professional Java Fundamentals, by Sly Cohen, Tom Mitchell and others is £32.49; ISBN 1-861000-38-3, published by Wrox Press.

Using Java (Second Edition) by Joseph Weber and others costs £56.49 (incl. VAT); ISBN 0-7897-0936-8, published by Que.

Delphi 2 Unleashed by Charles Calvert costs £54.95 (incl. VAT); ISBN 0-672-30858-4. These books are available from Computer Manuals 0121 706 6000



Active service

Tim Anderson investigates the Active Platform – is it really new? Plus how to use resources in Delphi, new books reviewed, and a preview of the Visual Basic control creation edition.

I'm sitting here looking at a sheaf of press releases and a stack of CDs which comprise the Microsoft Active Platform in its current, beta guise. The papers are an intricate display of verbal gymnastics: there are generous sprinklings of key buzzwords like open, standards-based, scaleable, multiple operating systems, and so on. The name Active Platform itself is a political statement. Sun calls Java a platform; Netscape Communications calls its browser a platform; others see the Network Computer as a platform. At stake is the question of who will be at the centre, and who will be satellites. Like all the best prima donnas, none of the main industry players wants to be anywhere less than centre stage.

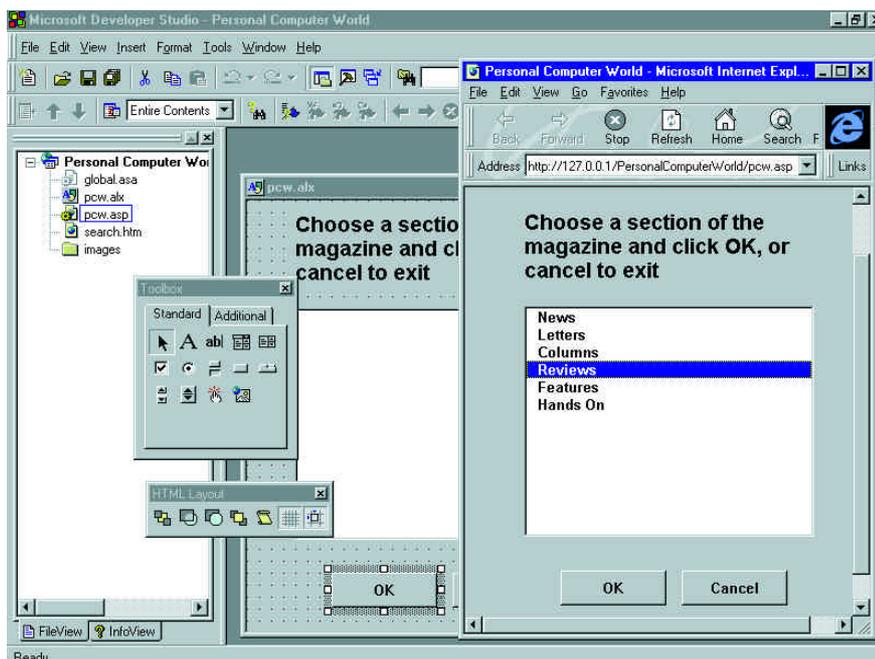
Where there's an O there's an A

You might be forgiven for wondering where this new Active Platform comes from. Microsoft's publicity implies that a range of new technologies, such as the Active Desktop, the Active Server and Dynamic HTML, have emerged brand new and sparkling from a magician's black hat somewhere in Redmond.

The truth is more prosaic. For years Microsoft has been promising to rebuild Windows on an OLE foundation, and that strategy has not changed. In many cases Microsoft has simply replaced the word OLE with Active. So, ActiveX controls are OLE controls, OLE automation servers are now Active Servers, and similarly the OLE object model once known as Data Access

Objects has become the ActiveX Data Object or ADO. With that in mind, here's a plain English summary of what is in the Active Platform.

1. Active Desktop This is essentially a web browser with support for HTML, VB Script, Java applets and ActiveX controls. In other words, it is Internet Explorer. Full implementation is in the forthcoming version 4.0, which is fully-integrated into the Windows shell.
2. Active Server This means that Internet Information Server can be controlled through what used to be called OLE automation.
3. Active Server Pages Here, Microsoft is referring to the ability to embed scripts, typically written in Visual Basic, into HTML web pages. Previously such scripts could only be executed by Internet Explorer on the client's PC. Now, a new tag lets you run the script on the server. Web sites have been doing this for years using CGI scripts, but this new approach is easier and removes the need to compile the script into a binary executable.
4. Dynamic HTML Code-named Trident, this is a set of extensions to HTML which implement much-needed features like layering and exact positioning. It provides an enhanced object model with more control over frames, tables and scripts.
5. Active Data Object Like Data Access Objects, this is a COM object model for database access. It hooks into ODBC for connectivity to a broad range of database servers.
6. Design-time ActiveX These are add-ins for Internet Studio which typically generate HTML and VB Script in response to user input while authoring a web page. You can



The Internet Studio project browser in file view (left), an ActiveX layout being designed (centre), and the resultant form at runtime in Internet Explorer



Unmistakably a platform; but with the ActiveX Platform, things are less clear cut

think of them as web page wizards. They are ActiveX controls but are not used at runtime and do not need to be downloaded to the client's computer.

7. HTML layout One of the most useful ActiveX controls is the HTML layout component, simply a container for other controls. Using the Active Control Pad or Internet Studio's layout editor, you can build forms which include scripts, much as you would with standalone Visual Basic.

Will it work?

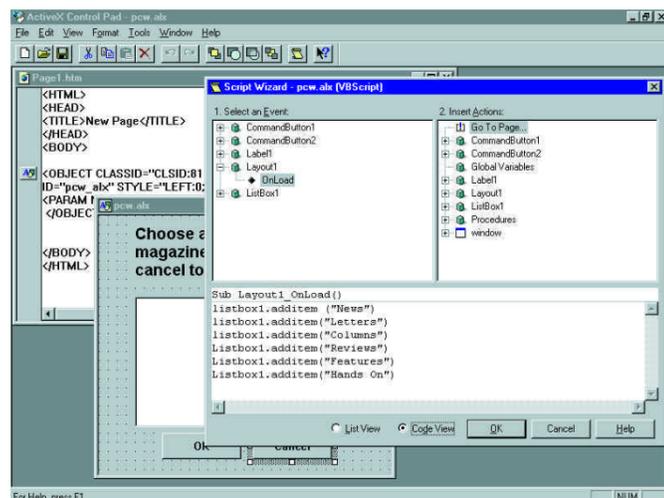
The usefulness or otherwise of Microsoft's new web initiative depends on which hat you wear. For general Windows developers, this is significant. Internet Studio, the tool that brings all these gizmos together, is a viable alternative to Visual Basic and Access. You can design forms, write VB code, and simply have your final application run within Internet Explorer rather than directly from the Windows desktop.

As Windows evolves, that last distinction will become increasingly blurred. The advantages are that your application can be Intranet-ready and database independent. On an Intranet, you have full control over whether code is executed on the client or on the server. Assuming Windows remains popular, I see this kind of approach as gradually replacing existing development techniques.

The ActiveX Control Pad

Released without fanfare onto Microsoft's web site, the ActiveX Control Pad is an essential tool for authoring ActiveX applications. It combines a simple text editor with a VB-like form designer. The idea is that you open an HTML document in the editor, design a form known as an HTML Layout, and then insert it into the document. The HTML layout is itself an ActiveX control, but exists only as a container for other components. You can also insert ActiveX controls directly, without using a layout. The control pad generates a bit of HTML code using the OBJECT tag, including the long alphanumeric CLASSID which uniquely identifies each ActiveX component.

The control pad does not only handle the placement of controls. Using the script wizard, you can also write code to bring the form to life. The scripting language can be either VB Script or JavaScript, although the two cannot be mixed on one page. In list view, the script wizard will



The ActiveX control pad, showing an HTML document, a layout, and the script wizard

write code for you based on your response to dialogs, or you can choose code view and bang out code in the old-fashioned way. There are many advantages to using the control pad. One is that you can position controls precisely within the layout,

something which you cannot do with pure HTML. Controls have a z-order too, so you can position one in front of another. The other plus is that a control's properties and methods are listed in the property editor and script wizard, so you do not need to look them up. Visual Basic programmers will soon feel at home. A similar tool is in Internet Studio, where it is called the HTML layout editor.

The biggest problem is that the control pad does no syntax checking and has no debugger — a sure sign of immaturity. Internet Explorer will report errors in your code, but otherwise you are reduced to tricks like throwing up message boxes to check the status of variables. The other problem is that ActiveX layouts currently work only with Internet Explorer 3.0. No surprise there.

What about Web developers? It is these that Microsoft is courting most visibly, with its "site builder" initiative. But success is far from assured. Microsoft can do what it likes with Windows, but does not own the web. The grave weakness of its Active Platform is that, despite noises to the contrary, it is not a cross-platform initiative. There is no problem with the server-side aspect, since the server can do what it likes as long as it delivers HTML that the browser can understand. The problem is with the Active Desktop. ActiveX controls, remember, are binary executables which run natively on the client's computer. If you want to create an ActiveX control which runs on, say, Windows, the Macintosh and Unix, then you must create three separate executable files. Even if Microsoft delivers what it promises, versions of Internet Explorer for these

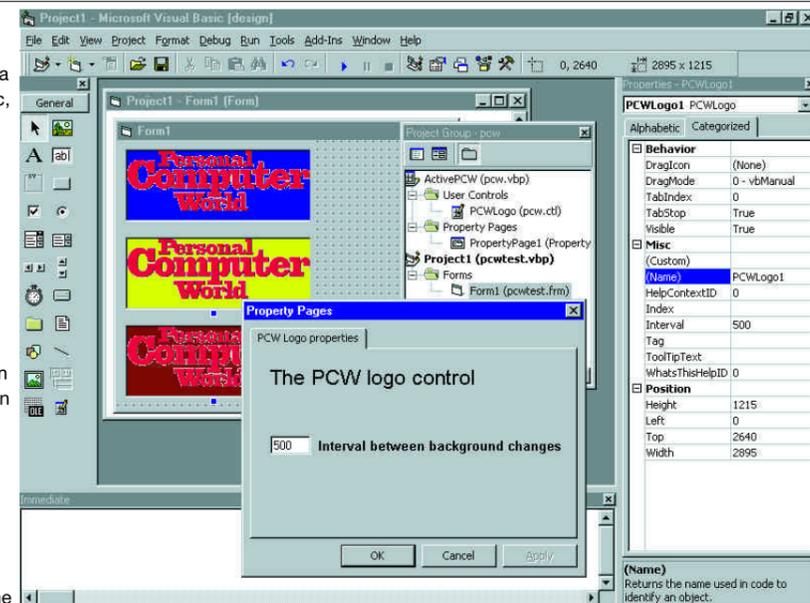
platforms, it is hard to see this strategy working. By contrast, a Java applet runs on any platform for which a Java Virtual Machine exists. That means Sun's Java Beans model holds all the cross-platform aces. Java applets can accomplish many of the same tasks as ActiveX components. Performance can be poor, but just-in-time compilers and eventually Java-based operating systems will crack that problem. Microsoft is making it enticingly easy to create web sites built with ActiveX controls, but such sites will to some extent shut out non-Windows browsers. If that drives more people to use Windows, Microsoft wins. But if these factors lead to Java rather than ActiveX dominating the Internet, the popularity of Windows itself will inevitably decline. The stakes are high.

Visual Basic Control Creation Edition

Unlike Internet Studio or the ActiveX control pad, the VB Control Creation Edition is not just for web development. As its name implies, it is a tool for creating ActiveX controls in Visual Basic, and these controls can then be used in any Windows development tool or document capable of hosting ActiveX, formerly known as OCX controls. In its determination to reinforce the ActiveX standard, Microsoft is making the control edition a free download, both the beta and final versions. Incidentally, it also offers a preview of what the VB 5.0 interface may look like when it emerges.

Since version 4.0 Visual Basic has been able to create OLE automation servers. You can declare an object class in a VB project, and then have other applications create objects of that class. Borland's Delphi 2.0 is similarly capable. The one piece missing in both products is the ability to create OLE objects that have a visual interface, or in other words, ActiveX controls. That gap has now been plugged. With the control creation edition, you can develop ActiveX components that can be installed on the component palette in products like Visual Basic, Access and Delphi. It is a great step forward, the main snag being that in this version, compilation to native code is not possible, so performance will not match ActiveX controls written in C++. VB controls can be very small, but require a substantial runtime library which makes distribution awkward. Microsoft now calls this the VB Virtual Machine. The implication is that a VM for Visual Basic may be implemented on more than one platform, although Microsoft has not stated this explicitly. Such a move would make ActiveX a more plausible cross-platform contender.

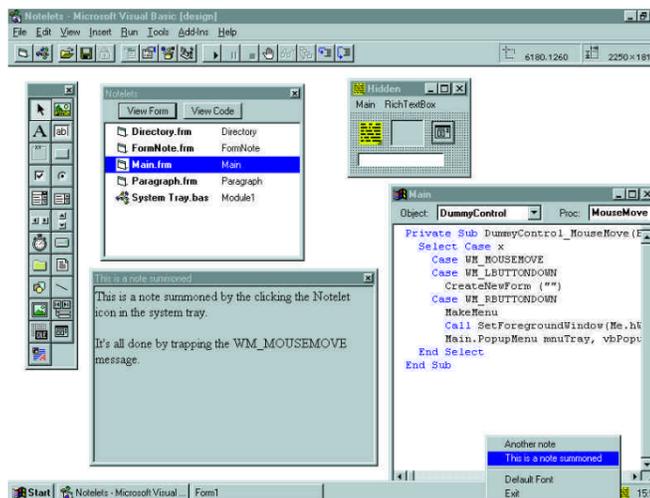
To test the control creation edition, I built a simple control. Using an image control and a timer, I displayed the PCW logo on a form. With one line of code I made the logo's background colour change whenever the timer event fired. Next, I used the Interface wizard to choose which properties and methods to expose, including a custom



A preview of Visual Basic 5.0. The Control Creation Edition at last makes it easy to write ActiveX components

property to set the timer interval. The property page wizard created a standard property page, and finally Make OCX built the control.

Nobody can now say that creating an ActiveX control is difficult. The main flaw in the VB control creation edition is not technology, but human fallibility. Creating a control is easy; but creating a good control still requires skill. The documentation observes how important it is to maintain a consistent interface when controls are developed, and warns that a poorly-implemented control can be a security risk even without malicious intent on the part of its developer. For example, if a method is exposed that enables a named file to be created on the user's hard disk, the control is not safe for scripting. Considering the number of VB developers, both professional and hobbyist, mistakes are inevitable.



Delphi
Delphi and Windows resources

when this might be essential: creating a screensaver for Windows NT, for example. It is yet another of those areas which Borland has scarcely bothered to document. Bizarrely, there is

Creating full system tray apps with native Visual Basic 4. See the full code on the cover CD

Delphi can use standard Windows resource files (with a .RES extension) and indeed there are occasions

documentation in WINAPI.HLP, supplied with Delphi, that covers the Microsoft command-line resource compiler, but not for the Borland resource compiler actually supplied. You didn't know Delphi comes with a resource compiler? It does, and it is the executable called BRCC.EXE or BRCC32.EXE, the 16- and 32-bit versions respectively. The versions called BRC.EXE and BRC32.EXE are shells which are able to call both the resource compiler and the resource linker, RLINK, to bind a resource to an executable — but you do not need to know this since Delphi will do it for you. To find out how these programs work, run them from a command line without parameters and the options are displayed.

What Delphi does not have is a resource editor. Simple resource scripts can be created by hand, otherwise you will want to use an editor such as the one distributed

with Borland's C++ products. Using resources in Delphi takes several steps:

1. Create a resource script and compile it to a .RES file.
2. In your Delphi application, include the compiler directive:

```
{ $R MYRES.RES }
```

where MYRES.RES is the name of your resource file. A good place to put it is in the project source below the similar directive {\$R *.RES} which Delphi includes by default in all projects. The reason is that the application icon is stored in a .RES file of the same name as the project. It is best not to edit this generated resource file, since Delphi may overwrite your work.

3. Your Delphi code can now load these resources using API functions. Here is a simple example. The following resource script contains a string table with one string:

```
STRINGTABLEBEGIN1, "I wandered  
lonely as a cloud"END
```

Save this as TEST.RC, compile it using BRCC to TEST.RES, and then include it in a Delphi project. Now you can retrieve the string in your Delphi application as follows:

```
lpzTest := stralloc(26);  
LoadString(hInstance, 1, lpzTest, 25);
```

where lpzTest is declared as a pChar. LoadString's second parameter is the ID of the string to load, often replaced with a constant for clarity, and the last parameter is the maximum length of the string to retrieve.

Visual Basic

More about the System Tray

James Talbut writes:

"You mention the usage of the Shell_NotifyIcon function and state that it is not possible to use the messages without additional software. But you can. Essentially you create a hidden control on your form and use an unrequired message for controlling it."

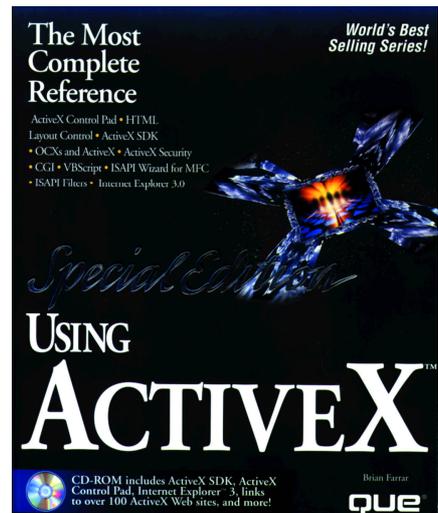
The system tray is controlled by an API call Shell_NotifyIcon, which takes a pointer to a NOTIFYICONDATA record as one of its parameters. This record includes fields for a window handle and a message identifier, the idea being that Windows sends that message to the specified window when the user clicks on an icon in the tray.

In C++ or Delphi you would use a custom message handler, but VB does not offer that facility. The workaround is to hijack an existing message handler, and

Books for Visual Programming

Using ActiveX by Brian Farrar

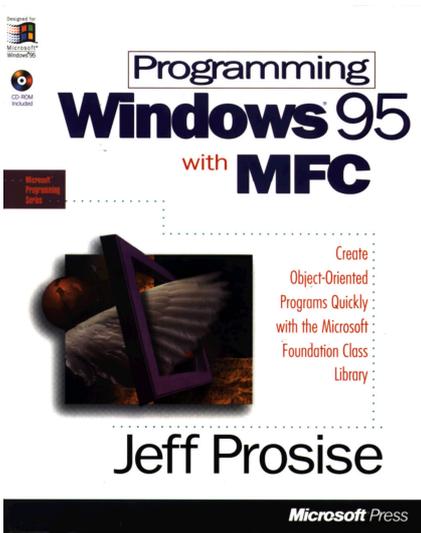
This is nearly very good. Aimed at those considering doing the web the Microsoft way, it presents all the main elements clearly and concisely, with examples. The book covers VB Script, ActiveX technology, the Control Pad, the Internet Control Pack, Internet Information Server and its ISAPI interface, and CGI scripting. It is fine as an introduction and overview, but does not go into enough depth to merit the "using" part of its title. For example, ActiveX security issues are skated over in a couple of pages. To be fair, Microsoft's ActiveX SDK, included on the supplied CD, does give the required detail; but most readers will have it already from another source. Buy this for an excellent overview, but expect to need further help very soon afterwards.



Using VBScript by Ron Schwarz and Ibrahim Malluf

This title is both longer and more tightly focused than its companion, Using ActiveX, in the same series. Without assuming much prior knowledge, the authors show how to program web pages using VB Script, touching on related areas like ActiveX and SQL Server web extensions. Considerable space is given to HTML itself, including an appendix documenting all HTML tags supported by Internet Explorer 3.0. There is a CD with all the examples from the book, and as a bonus, the full text of another Que title, Using Visual Basic 4.0. It is a nice extra, but ironically none of the web pages on the CD are well designed. Overall, it is a good introduction to VB Script, but do not expect it to answer all your Web queries.

Programming Windows 95 with MFC by Jeff Prossie



You have to respect someone who knows his limitations. Jeff Prossie is not a database programmer, nor is he an OLE enthusiast. "Certain parts of OLE are promising," he says, "but the OLE documents protocol is overly difficult to implement and of limited value in the real world." That explains why his book on Microsoft's Foundation Classes, the leading C++ Windows class library, covers neither MFC's database classes, nor OLE in any form. Instead, he gives a nuts-and-bolts description of how to program with MFC, starting with "Hello World" and progressing to documents, views, common controls and multi-threaded development. It is a valuable book, since most other tutorials focus more on using Visual C++ and its wizards, than on MFC itself. Look elsewhere for ActiveX, web development or database work, but buy this book to learn the fundamentals of Windows development using MFC.

James suggests using a hidden picture box and the WM_MOUSEMOVE message.

Then, you can write code in the MouseMove event that will respond to system tray events.

It works, and James has written an example notelet application, which is on the cover CD. It lets you store notes which pop up when you right-click an icon in the tray. Thanks, James – you have won a book token for your efforts.

PCW Details

Tim Anderson welcomes your Visual Programming comments and tips. He can be contacted at the usual PCW address or at visual@pcw.vnu.co.uk

Programming Windows 95 with MFC by Jeff Prossie (Microsoft Press), book and CD, £49.95 inc VAT.

Using ActiveX by Brian Farrar (Que), book and CD, £37.99 inc VAT.

Using VB Script by Ron Schwarz and Ibrahim Malluf (Que), book and CD, £46.99 inc VAT.



Spare parts

Tim Anderson compares components for Basic and Delphi. Plus, for those who can't see the wood for the trees there's a guide to choosing a visual programming package.

Visual programming means dropping components into your application and making them go by setting properties and calling methods. That, at least, is the plan and here is an evaluation of some recent components. Your views matter most though, so please let me know which components work well or badly for you.

Crystal Reports 5.0

Crystal Reports is hard to avoid, being widely bundled with products like Visual Basic and Visual dBase. Seagate naturally hopes that users of these bundled versions will want to upgrade. Version 5.0 is the latest release. The Standard version supports most desktop database formats like dBase, Access and Paradox, while the Professional version adds full ODBC and various native SQL formats.

There is a new interface for designing reports, with better drawing features and in-place OLE editing. Of most interest to developers is the new sub-report feature, which enables you to insert a report within a report. Normally, this would be linked to the main report for displaying child records, but it can also display unrelated data. Another new feature enables you to export HTML for adding to web sites.

Crystal is a powerful tool and has components for most development languages including 16- and 32-bit Visual Basic, Delphi and C++. But I do have reservations. One is the sheer size of the product: the main print engine is now over 3Mb; another is that Crystal has its own formula language and although reasonably capable, it is ugly and old-fashioned. Nevertheless, version 5.0 is a significant

upgrade and developers with better things to do than write their own report engine will find it invaluable.

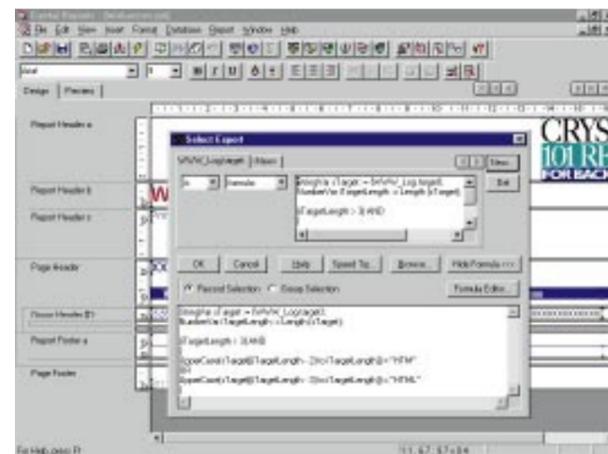
GeoPoint 1.0

Visual Components is responsible for some of the best ActiveX components around, including Formula 1 and Visual Writer. GeoPoint is a more specialist control. It

displays maps in MapInfo or Autocad format. By using it alongside the separate Legend control, you can programmatically control the text and shading of each area of the map. A technique called "binning" lets you categorise data into ranges, and then shade the map accordingly: a typical example would be a display of sales performance by region. The user can also

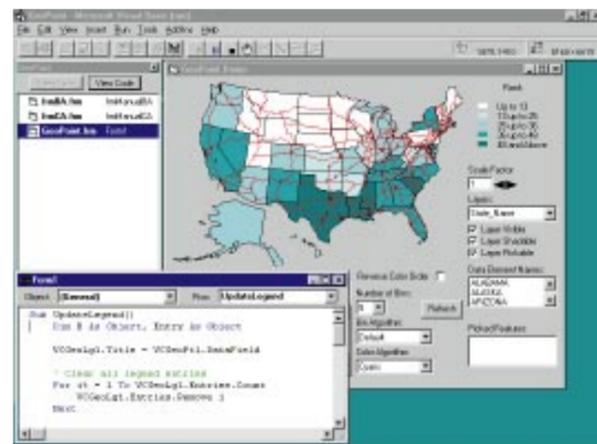
select an area of the map by clicking, so the application can show related data. GeoPoint can be bound to a data control to display your data.

This useful component is spoiled by its presentation. There is no printed manual and the online documentation is poor. The other snag is that the few supplied



Above The Crystal Select Expert lets you create custom fields using the formula editor

Right GeoPoint is a neat tool with which to analyse geographic data, but its USA map will not be of much use in the UK

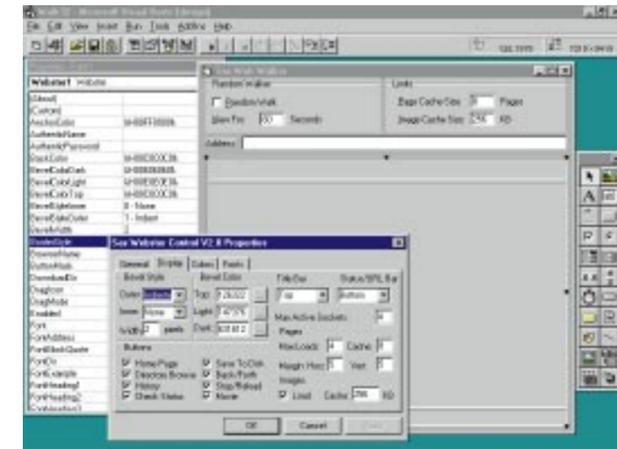


maps, heavily biased to the USA, are not likely to be what you want. That means purchasing add-on maps, or buying MapInfo or Autocad to create your own. Making full use of GeoPoint will be expensive.

Sax Webster or Internet Explorer?

The Sax Webster control displays HTML documents. Now at version 2.0, it comes as 16- or 32-bit OCX controls that you can drop into your application.

It is easy to use. You can, for example, display a web page by setting the PageURL property. The main change from the first Webster control is the HTML version supported, now version 3.0 but without frames. It works well and may be useful on 16-bit systems or where a small memory footprint is required. Otherwise, on 32-bit Windows a better option is Microsoft's freely available Internet Explorer 3.0. In Visual Basic 4.0, open the Custom Control dialogue and check Microsoft Internet Controls. This installs the WebBrowser component which is the HTML display part of Internet Explorer. It is just as easy to use as Webster, and far richer in terms of HTML support. You will need to get hold of the Internet Explorer object model, which is part of the ActiveX SDK available on the Microsoft web site.

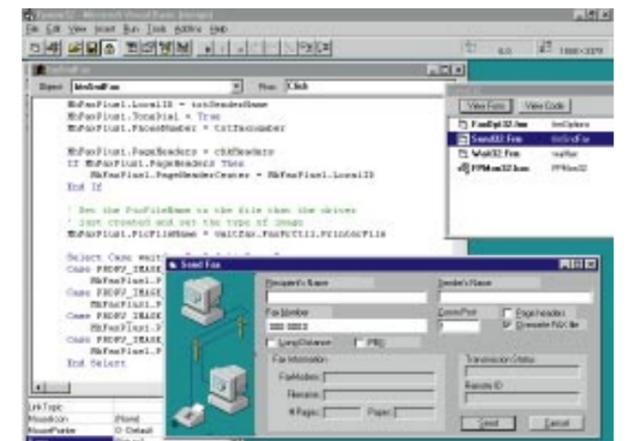


Left Sax Webster: it works, but why not use Internet Explorer instead?

Microhelp Fax Plus

Fax software may not be exciting but it is exceptionally useful, at least until the whole world gets webbed.

Fax Plus 2.0 is the new 32-bit version of Microhelp's Fax add-on. It is designed for Windows 3.1 and 95, but not NT. It consists of several controls, including a fax control



Below FaxPlus: does it have to be so complicated?

How to choose a visual language

Shaun Nicolson writes: "I am considering buying a visual programming package but cannot decide which one. I am considering Microsoft's Visual Basic and Visual C++. The language I choose would have to produce network applications. What are the pro's and con's of these?"

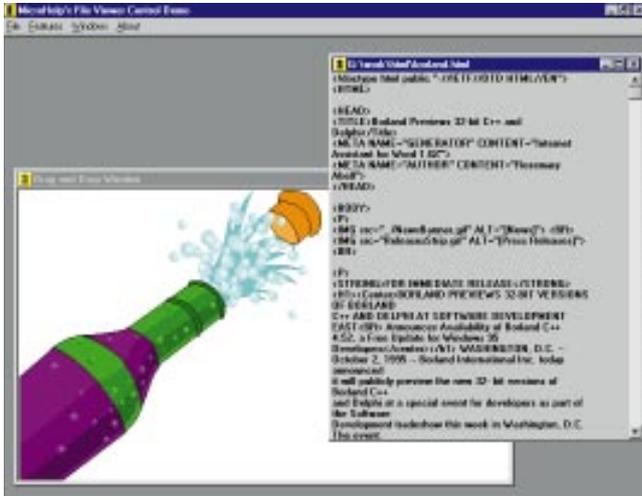
1. Ease of use. This is where Visual Basic scores highly, since you can have a simple utility up and running very fast. That does not mean VB will be the easiest for a large, complex application, since many other factors will then come into play. Delphi also scores well, while Visual C++ is hard to learn with limited visual tools.
2. Performance. This is where languages that compile to native executable code, like C++ or Delphi, generally win over interpreted languages like VB, FoxPro or Java. Database processing speed should be judged separately as all products use fully-compiled database engines. In some applications, performance is not an issue, or is determined more by factors like graphics or hard disk speed rather than the language used.
3. Power. The developer's nightmare is to spend months on a project, only to find that some intractable problem means that it cannot be completed with the current tool. Visual Basic is vulnerable since some features of Windows like callback functions or custom message handling are not available. There is usually a way around it by using a custom control or DLL but these must be written in another language. Version 5.0 should solve some of these problems. If you dread brick walls, C++ is the safest option, with Delphi a close second.
4. Database engine. Most languages have a native database engine, along with the ability to connect to other databases via SQL libraries or ODBC. If you know which database you will be working with, good connectivity is the first thing to check.
5. Availability of add-ons. This is where Visual Basic scores best. Most VBX and ActiveX controls are designed for VB, and may not work well elsewhere. There are also plenty of code libraries for C and C++, but native add-ons for other languages are more limited.
6. Reusable code. To protect your investment, you want to write code that will be reusable in future projects, perhaps even on other computer platforms. This is one of the benefits of object orientation, with Delphi scoring well, C++ fairly well, and Visual Basic less well. Best of all is Java, which forces you to write object-orientated code and runs on multiple platforms.

that handles communications, and a FaxImage control for creating and modifying fax bitmaps. There is a printer driver and control which lets you send faxes by printing from any Windows application. The Fax Plus driver creates a fax image and then fires a StartDoc event in the printer control, so that your code can handle sending the fax.

Unfortunately though, using FaxPlus is not as easy as it should be. In part this is because of fickle telephone lines and diverse hardware that turn faxing into a trouble-prone business. Other problems are down to FaxPlus itself, which is awkward to code and not entirely bug-free. For instance, at the time of writing, the VBX version is unable to correctly convert text files to fax images. Still, it beats trying to write your own fax driver.

Microhelp VB Viewer 2.0

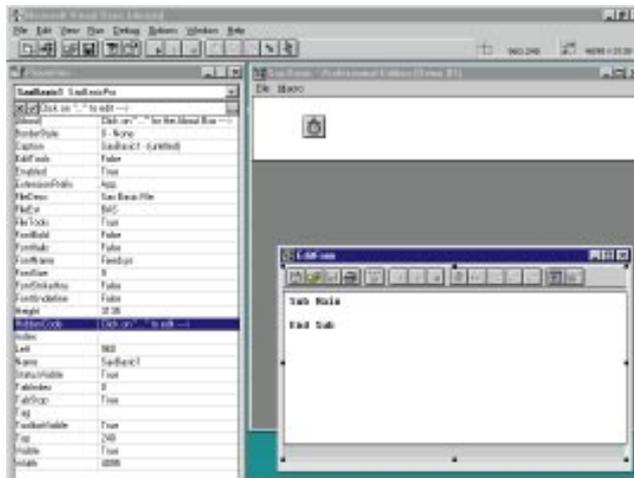
VB Viewer is a product of limited ambitions. Drop it onto a form and you can use it to display files of around 30 different types, including multimedia files. With text documents you can search for text and



Left VB Viewer can manage a picture, but struggles to display HTML

Below A language within a language, Sax Basic lets you deliver programmable applications

copy to the clipboard. But overall, VB Viewer is a disappointment. One let-down is that formatted documents in word processor formats are displayed as plain text only, unlike the much better QuickView utility that comes free with Windows 95. Some basic formats like HTML and Rich Text Format are not supported at all. You can set VB Viewer to use QuickView viewers, but files then appear in the QuickView window rather than embedded in your form.



has a more elegant solution. You write extensions in a VB class module, and then add them to Sax Basic using the control's AddExtension method. If your application needs a macro language, Sax Basic is ideal and a lot cheaper than licensing the genuine

Sax Basic Engine

One way to impress users is to supply an application with its own macro language, like Excel or Word. The Sax Basic Engine lets you do just that. The control has its own IDE, so getting started takes little more than placing it onto a form. The language itself is compatible with Visual Basic for Applications, with excellent support for OLE automation and class modules.

To make Sax Basic useful, you need to extend its language to communicate with your application. The way to do this differs, depending on whether you use the VBX or OCX version. With the VBX, you can add keywords that fire an event called AppExec. You can then write code using Select Case to interpret the command. The OCX version

Merging bitmaps using VB and the SRCAND bitwise operator

VBA from Microsoft.

Visual Basic

MT Emms writes: "Using Paint I've created four BMP files. Each is divided into four sectors, the other three being left transparent. I have written a program to merge these bitmaps into one but the last one dominates — in other words, the transparent sectors are not transparent. It was simple on the Mac and Archimedes; surely VB should be able to cope?"

Visual Basic can cope, but it is not as simple as it might be. The secret in this case is to use the PaintPicture method, on either a form or a picture box. The syntax for PaintPicture is:

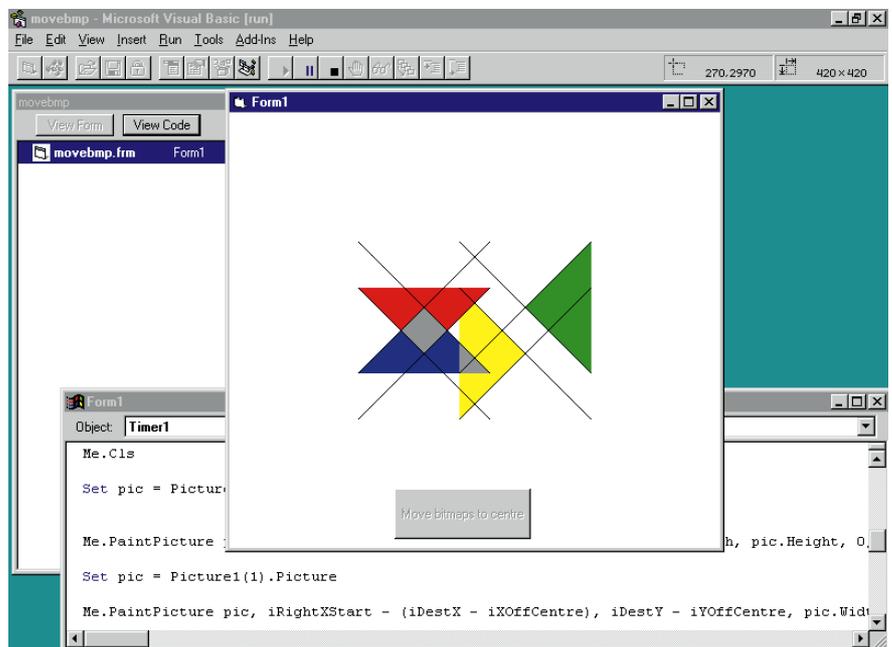
```
object.PaintPicture picture, x1, y1, width1, height1, x2, y2, width2, height2, opcode
```

The final parameter is a long value that defines a bit-wise operation which is performed on the picture as it is drawn. As the VB manual remarks, you can find a list of these operators in the BitBlt topic in the Windows SDK help.

The easy way to use them is to define them as constants in your VB application. For example:

```
Const SRCAND = &H8800C6 ' (DWORD)
  dest = source AND dest
Const SRCOPY = &HC0020 ' (DWORD)
  dest = source
```

If you then call PaintPicture with the SRCAND constant value, the bitmaps will merge in the way Mr Emms requires. Yes, it is more like programming in C than in Visual Basic, but at least it does the job. An example application is on our cover-



mounted CD, which also shows how to move the bitmaps across a form for an animated effect.

Delphi departure

In October, Anders Hejlsberg, the chief architect of Delphi, announced his departure from Borland for the safe pastures of Microsoft. Zack Urlocker, another key member of the Delphi team, has pointed out that "the architectural work that Anders covers is complete for Delphi 97. Anders' departure won't affect the shipping date or features going forward."

Even so, Anders is widely seen as the man without whom Delphi would never have happened, so his move is significant news. If anyone can knock VB into better shape as an OO language, he must be the man. Although Delphi is as good as ever, this weakens the case for migration from Microsoft tools. Personally, I hope that Borland can sustain Delphi's momentum, as it still delivers the best combination of rapid

Dear Santa...

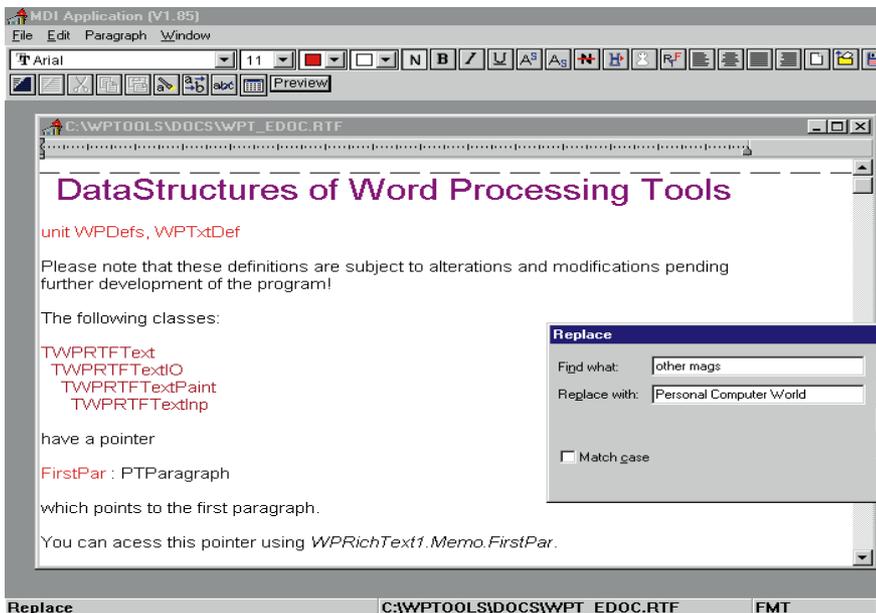
Sensible developers want an easy life. That means fast application building, reusable code, blinding performance and results that run everywhere. Well Santa, it seems you have a habit of giving with one hand and taking back with the other. Last year I asked for an end to the OS wars: a fanciful request, perhaps, but twelve months on and Java may provide an answer. Except that (dear Santa), we need easier, richer interface building, better performance, and decent support for platforms which Java finds difficult, like Windows 3.1 and the Macintosh. In the meantime, even developers who fix upon Windows have three versions with which to contend.

Forget the platform, then, let's look at the tools. First, there's Visual Basic, still the most popular all-round Windows language. Last year's wish-list included a compiler and better OO. The signs are that VB 5.0 delivers some of that, although it will never have the elegance of Delphi's component library. But Office 97 and VBA 5.0 are great news for developers of Office solutions. Thanks, Santa. And thanks for Optima ++, which is real visual development for C++ at last.

While I have your attention, there are a few things I'd like in my stocking for next year. Top of the list has to be a faster, better-organised internet. The web is irresistible for developers, both for technical support and as a platform in itself. But it has to get quicker and more reliable. *Please*.

Second, an un-present: Windows 3.1, please take it away. It's as bad as DOS, but worse, because people with working Windows 3.1 installations see no reason to change. I understand their point of view, but for developers this is a disaster. Develop two versions, with all the extra costs? Develop 16-bit only and waste all the advantages in 95 and NT? Develop 32-bit only and forget half the market? Hmmm, did I hear someone mention Java?

Third, I'd like better tools for troubleshooting OLE, ActiveX, COM, call it what you will. It's funny how quickly a Visual Basic 4.0 installation produces an "object server not correctly registered" message when you install custom controls. It means a registry problem and there's no easy way to fix it. This highlights a problem that will get worse if ActiveX continues to grow.



development, power and performance.

Word processing tools

Some months ago I mentioned a shareware product called WP Tools, a native Rich Text control for Delphi. On closer inspection, I am impressed. The feature list is good, with support for merge fields, graphics, tables and hyperlinks. The range of controls has a lightweight rich text label and a data-aware text box as well as the usual word processor, toolbar and status bar components. In tests, it has proved fairly reliable, though not entirely bug-free.

The advantage of WP Tools is that as a

native VCL supplied with source, you can track down bugs and amend the code if you can work out what is going wrong. Another benefit is richer functionality. You can access the data structures for both text and formatting, giving a fine degree of control. The Finder class offers sophisticated search and replace, including formatting properties. You can print to a canvas control in order to implement page preview.

Performance is good, on a par with rivals like Visual Writer, AllText and HighEdit. If you are developing for 16-bit Windows, a custom component is all but essential for

WP Tools is a fully-featured, shareware-rich text control for 16- or 32-bit Delphi

working with formatted text, while even in Windows 95 and NT it has advantages over the built-in rich text control.

There are problems. This is shareware, and the documentation is unclear. Advanced users need to be comfortable with such things as streaming and pointers, as WP Tools uses them extensively. To succeed with this product, you must be willing to pore over the source and not be put off by the odd mixture of English and German comments in the code. The extra effort and risk is rewarded by a product that works rather well.

PCW Contacts

Tim Anderson welcomes your Visual Programming comments and tips. He can be contacted at the usual PCW address or as freer@cix.co.uk

Components listed below are available from:
Contemporary Software 01344 873343; **Grey Matter** 01364 654100; and **QBS** 0181 956 8000.
Sax Webster £110 for the 16- or 32-bit version (£175 for both), plus VAT.
VB Viewer £110 (plus VAT)
FaxPlus £195 (plus VAT)
Seagate Crystal Reports 5.0 Standard £199, Professional £299 (both plus VAT)
Sax Basic 3.0 Pro £345 (plus VAT)

GeoPoint 1.0 costs £195 (plus VAT) from **Visual Components** 01892 834343
WP Tools is \$119 to register, available via CompuServe, or from the web at members.aol.com/JZIERSCHE/wptools



Future threat

What's behind Visual J++? Might it overtake C++ as a mainstream Windows application? Tim Anderson talks to Microsoft. Plus, screensavers in Delphi, and a free MSDN sample.

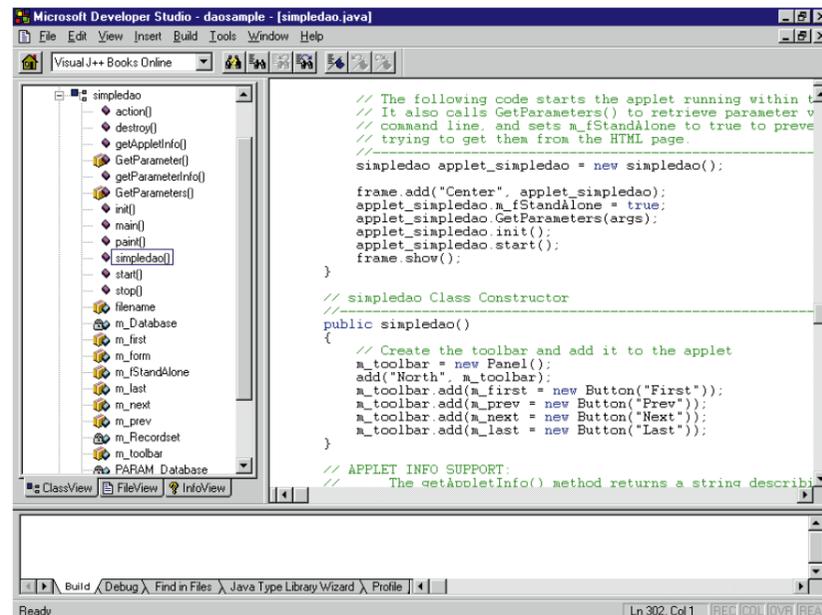
Microsoft's Visual Java tool is now out (see our review in "First Impressions"). But what is the company doing with a language that threatens Windows desktop dominance? Another consideration is that Sun's proposed Java Beans component model is at odds with Microsoft's ActiveX strategy. I interviewed Microsoft's development manager, Greg DeMichillie, to find out more.

Visual J++

PCW: I was disappointed by the lack of a visual environment for building an interface.
DeMichillie: "I totally agree. Our long-term direction is towards graphical interface builders but the reason we haven't got this is because of work going on with class



DeMichillie: Aiming for large-scale development in Visual J++



Visual J++ integrates seamlessly into Windows, but will other Java players accept Microsoft's ActiveX standards?

libraries. The question is whether the Abstract Window Toolkit is the long-term windowing model, or whether it will be an alternative? I'll be back in less than a year talking about a new version of Visual J++."

PCW: What are the problems with the AWT library?

De Michillie: "The first question is whether AWT will continue to only do things that can be done on 19 Unix variations plus the Mac, plus Windows 3.1, plus NT, plus 95. Second, AWT was developed by many different people and that comes through in the APIs that are exposed. There are different designs and they don't mesh well. There are problems with layout, which is entirely code-based. That means when I lay

out a form, the layout is stored in the code for the form's class. There's no separation of the code from the data. Maybe these points are addressable, but the larger architectural problems are more difficult.

"I think AWT was rushed out early. A lot of the fundamental Java technology was ready to go; the byte codes, the compiler and AWT got rushed. We support AWT because there are no viable alternatives."

PCW: How would you envisage your class library developing? Would you implement Windows-specific features, or go for compatibility?

DeMichillie: "My personal view is that the least common denominator solutions are not ultimately compelling. We want to

Books for Visual Programming

Visual J++ by Charles Wood

The best and worst thing about this book is that it exists. It was on sale before Visual J++ was officially released in the UK and unfortunately it shows signs of haste. Nearly a third of the book is a list of Java methods — something you can get from online help. There is only brief coverage of ActiveX and COM and the author refers us to his forthcoming, more advanced book on the subject. Data Access Objects, one of the key selling points of Visual J++, are hardly mentioned. The rather important resource wizard (which lets you convert Windows menus and dialogues to Java) is skimmed over as being "beyond the scope of this book". What's left is an unexceptional general introduction to Java programming. It's not really the author's fault, since he was working with a beta product. But my message to you is, beware — the first books to come out for a new product are often not the best.

expose all the richness and functionality of Windows but we want to do so in a way that enables us to port to other platforms.

"For example, take Direct3D and DirectX, our multimedia systems. Those take advantage of high-performance Windows graphics cards, but the API is generic enough to implement on other systems. Or database access — there's no reason database access APIs would only be on the Windows platform."

PCW: Is Microsoft happy to see Sun controlling the language, or would you like to see an ANSI Java, or something like that?

DeMichillie: "We don't necessarily need an ANSI committee, Sun has control over what is considered standard Java. But there are a number of vendors working on class libraries independent of Sun and over which Sun has no influence.

"I would expect Sun to be keenly involved in things like the byte code format, but I don't think class libraries are really one of those areas. In our relationship with Sun, we're competitive in many areas and I make no apology for that. But having said that, there are going to be huge areas of commonality. We don't want to see byte code format proliferation. I think the Java language will evolve."

PCW: What about the Java Beans proposals? Do they fall into the area that is competitive?

DeMichillie: "It's difficult to say, simply because Beans is just so vaguely defined at this point. We want to make sure that anything Beans does works well with COM.

"Our overriding goal is to make sure that Java developers have access to the thousands and thousands of COM components that already ship. ActiveX controls and OLE controls form the most successful component software market."

PCW: Why have you hooked into Internet Explorer (IE) and not made your product browser-independent?

DeMichillie: "We're not directly hooked into IE. We're hooked directly into MS's implementation of the Java Virtual Machine, which currently is hosted inside IE. But because the interpreter is itself an ActiveX control, that VM could be hosted inside any executable. The first reason is that the VM offers COM support and ActiveX support, and second, the VM supports a new set of debug interfaces. I would personally love Netscape to adopt the Microsoft VM so we could cut down on the number of VMs that are out there."

PCW: At some future point, might Visual J++ be able to compete with Visual C++ in creating mainstream Windows applications?

DeMichillie: "Sure. My ultimate goal is for large-scale development to be possible in J++. The growth in Java will come at the expense of C++. People have now dealt with C++ for a number of years and have

seen some aspects that are more complicated than they might like. Java offers a simplification that is very appealing. As the tools evolve, Java will be able to do many of the things that now would need C++.

"It's important to distinguish between component builders and component users. The majority of ActiveX controls out there are written with C++.

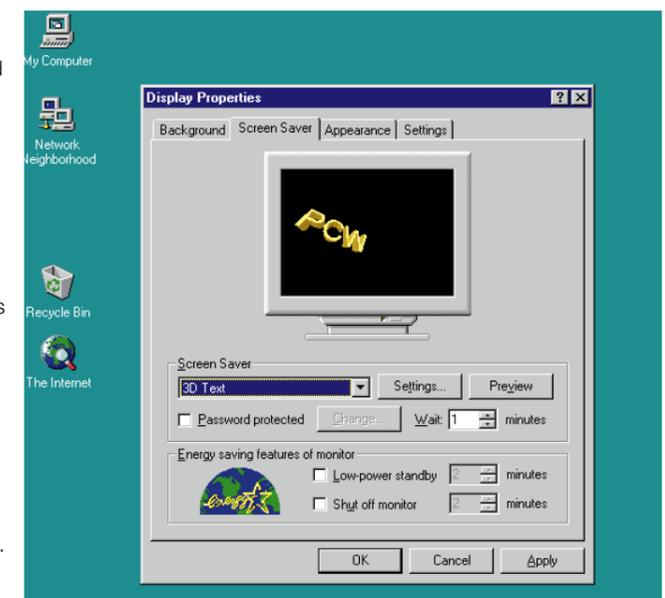
VB 5.0 will also create controls. It will be interesting to see where the component builders go. Component users gravitate towards VB and will eventually gravitate towards VJ++ as these very GUI-based, RAD-like tools appear."

PCW: If the lowest common denominator is not a long-term solution, does that mean cross-platform isn't either?

DeMichillie: "No. There will be a core subset that's the same everywhere. But there will also be extra capabilities, even in Java, where one platform has an extra class and another doesn't. For example, there are a number of capability differences between Windows and the Macintosh. Do you really want to restrict the class library to only those that are common? Or do you want a class library that has the room to contain components that maybe work on three or four platforms, and other components that work on a different three or four platforms? Microsoft understands that the market is not just Windows, but includes Macintosh as well as Unix. But it does not follow that you are only going to do things that can be implemented on every platform."

DELPHI

Anyone who writes a screensaver must have time on their hands. Screen burn is not common now, and in any case, perfectly functional savers are supplied with Windows. But they are fun! At least,



The Windows 95 screensaver control panel looks slick, but increases the work for developers

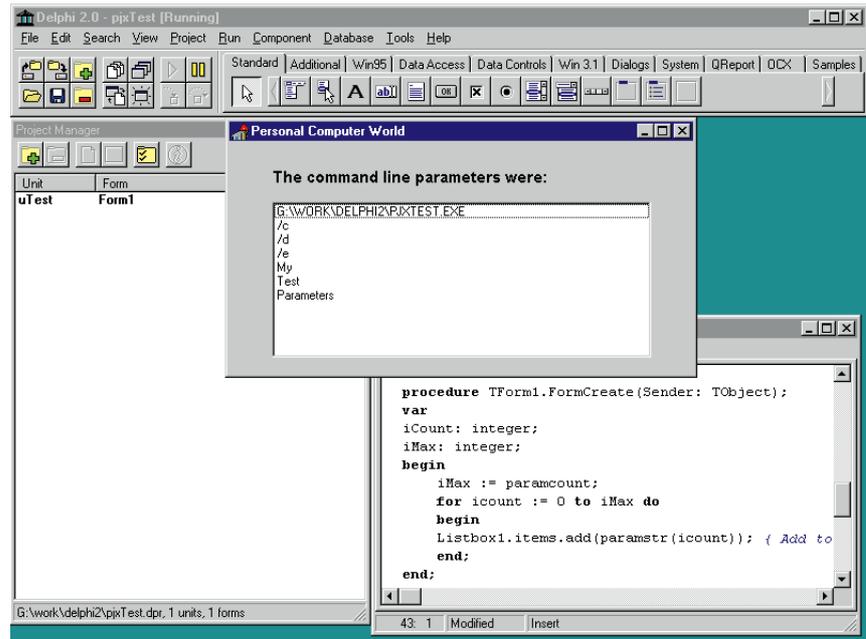
Delphi tip: Detecting command line parameters

It is often useful to supply parameters to an application at startup. For example, if an application handles documents, then passing the name of a document as a parameter should run the application and open the document. Delphi has two functions to make this possible. `ParamCount()` returns the number of command line parameters, and `ParamStr(Index: Integer)` returns a string representing the parameter that corresponds to `Index`. `ParamStr(0)` always returns the application name with full path.

The following routine detects command line parameters and writes them to a log file:

```
var
  iCount: integer;
  iMax: integer;
  F: textfile;

begin
  AssignFile(F, 'C:\TESTLOG.TXT');
  Rewrite(F);
  iMax := paramcount;
  for icount := 0 to iMax do
  begin
    WriteLn(F, paramstr(icontains)); {
      write to log }
  end;
  CloseFile(F);
end;
```



Andrew Jeffries must think so, since he asks: "I am having a few problems writing 32-bit screensavers using Delphi 2.0, running Win95 and NT: How do you do a small preview in 95 without the configuration form always appearing? How do you implement security? How do you make the screensaver's name appear in NT and 95?"

The problem with screensavers is that they are not well documented. The trusty Software Development Kits (SDKs) for the various Windows versions assume you will use C or C++, and that you will link your application with `SCRNSAVE.LIB`, a Microsoft-supplied library that holds the secrets of screensaver operation. Screensavers do not have to use `SCRNSAVE.LIB`, but avoiding it means extra work on the part of the programmer. Another snag is that screensavers work differently in each version of Windows.

Screensavers are executed by Windows in two ways: either when an interval of inactivity causes Windows to execute the screensaver, or when it is being configured in Control Panel. In Windows 95, screensavers have four modes of execution and these are selected by command line parameters:

- **Preview mode.** When you select the saver in Control Panel, Windows sends two

parameters, `/p HWND`, to select preview mode and to pass the handle of the preview window.

- **Configuration mode.** When you click Settings, Windows sends a parameter, `/c`, to select configuration mode. The saver responds by presenting a configuration dialogue.

- **Password mode.** When you click to change the password, Windows sends two parameters, `/a HWND`, to select password mode and to pass the handle of the parent window for your password dialogue.

- **Start mode.** When you click Preview, or when the saver is called for real, Windows sends a parameter, `/s`, to select start mode.

So the answer to Andrew's first question is that the application should check the command line parameters to see whether it should draw in the preview window or present a configuration dialogue. See the tip panel (above) for how to detect parameters.

Screensaver security is treated in different ways by Windows 95 and Windows NT. Under Windows 95, most screensavers call the Windows Master Password Router. This is a DLL called `MPR.DLL` which exports password functions like `PwdChangePassword`. They are usually called via another DLL, `PASSWORD.CPL`, which works as an

extension to the Control Panel. Neither of these libraries are fully documented in the Windows SDK, but some have worked out how to use them. The alternative is to implement your own password checking and throw up your own password dialogue when the saver is called in password mode.

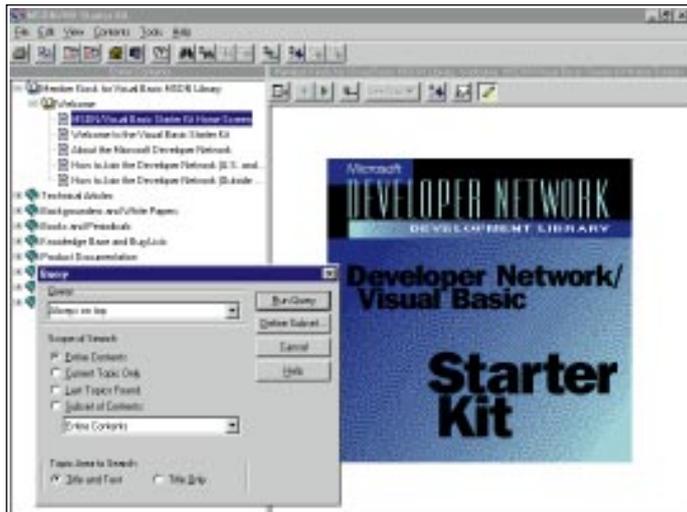
Windows NT is different. Passwords for NT screensavers are the same as those used for logging on to Windows. The Control Panel marks a registry entry to indicate a secure screensaver:

```
HKEY_CURRENT_USER\Control
Panel\Desktop\ScreenSaverIsSecure
```

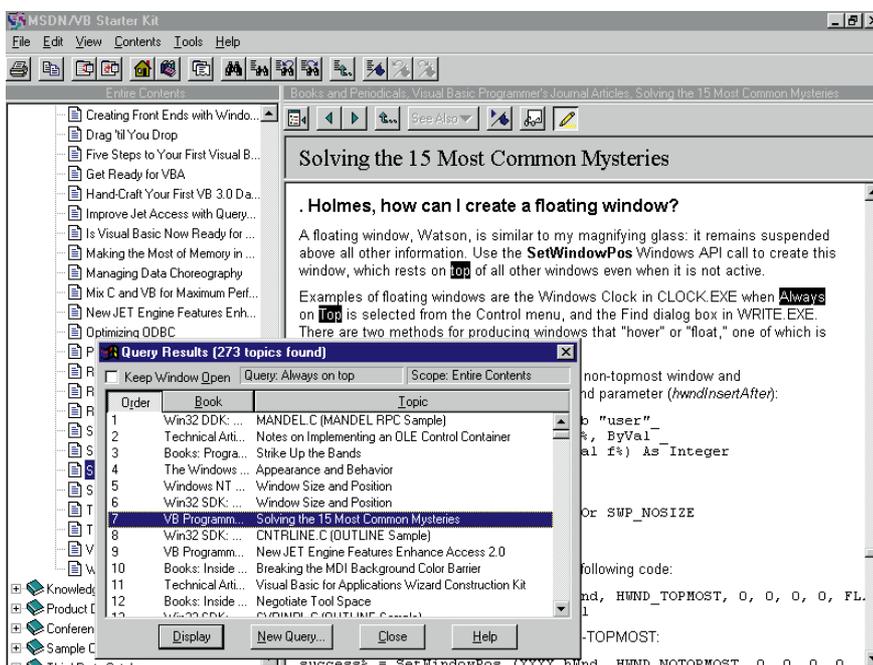
Finally, there is the matter of the description line. Confusingly, Microsoft has devised three ways of identifying this. Originally, it was the module description entry, which had to be of the form "SCRNSAVE : My Description."

Under Windows 95 and NT it became a resource string with an ID of 1 — and yes, Delphi can use standard Windows resources. This is the documented way; but actually, Windows 95 does not use it. It simply uses the long filename, less the `.SCR` extension. By the way, Windows will find any screensaver, identified by a `.SCR` extension, in the Windows or System folders, so at least installation is easy.

I've answered Andrew's questions, but I



(Left) Free on our cover-mounted CD: the Microsoft Developer Network starter edition is a mine of VB and Windows information (Below) MSDN query results are displayed in order of likely relevance and clicking a title in the list displays the selected article



free MSDN sample on our cover CD-ROM will soon pull up an example. Alternatively, if you prefer native BASIC, there are the neglected statements like Open, Print# and Input#. These are well-documented in the Visual Basic manuals, with examples.

Microsoft Developer Network

By special agreement and arm-twisting, we've included the MSDN starter edition for Visual Basic on this month's cover-mounted CD-ROM. I get regular enquiries about MSDN, and now you can try it for yourself.

Although this is only a starter edition, there is 125Mb of documentation, tips and tricks included, so no-one need feel short-changed. It even includes two complete books: Petzold's classic *Programming Windows 3.1*, and BrockSchmidt's tome explaining OLE 2. Much of the information covers VB 3.0 as well as 4.0.

The best thing about MSDN is its fast searching. For example, you might want to know how to set a window to be always on top. Click search, enter "Always on top", and then Run Query. A moment later, MSDN presents 273 topics ordered by likely relevance. Article 7, "Solving the 15 most common mysteries", has a section explaining exactly how to do it.

A subscription to the full MSDN comes at several levels and prices. Information is available on the CD, or call Microsoft for details.

■ See next month's *PCW* for a review of the new Crystal Reports 5.0, and components from Sax software and Microhelp.

Cover Disk

Files from last month's *Hands On Visual Programming* were, unfortunately, left off the CD, but they can be found on this month's cover disc.

PCW Contacts

Tim Anderson welcomes your Visual Programming comments and tips. He can be contacted at the usual *PCW* address, or at freer@cix.co.uk or www.cix.co.uk/~tim-anderson/

Learn Borland Delphi in 21 Days £34.99 (plus VAT), or £69 (plus VAT) for the Delphi 2.0 version. Borland is on 01734 320022.

Visual J++ is by Charles Wood, ISBN 07615-0814-7. £32.99 from Computer Manuals, 0121 706 6000.

Microsoft Developer Network is on 0800 960279.

do not mean to suggest that writing screensavers is easy. The main problem is poor documentation, especially if you are not using Visual C++. A hunt around CompuServe or the web will throw up Delphi examples and help files created by other frustrated users.

■ Psst! Want Delphi cheap? Borland is bundling Delphi 1.0 with a book, *Teach Yourself Delphi in 21 Days*, and offering it for just £34.99 (plus VAT). A similar package for Delphi 2.0 costs around £69. Borland says the packs are aimed at "students, hobbyists and programming beginners".

VISUAL BASIC

The web is buzzing with talk of VB 5.0, now likely to be released early in 1997. There's even a web site devoted to VB 5.0 news

and comments from anonymous beta testers. If the rumours are even half true, it looks like both performance and features will be hugely boosted.

In the meantime, Aaron Hodgson has contacted me with a question about Visual Basic: "I am trying to write a terminal program using SAXCOM.VBX. The terminal works fine but I would now like to add an automatic logon sequence, where the logon details are read from an initialisation file when the remote computer on the other end of the modem prompts the user to log in. If the answer to my questions involves complex things like DLLs please explain, because I don't know the first thing about using DLL files with Visual Basic."

Reading data from an initialisation file is not difficult. You can use API functions like GetPrivateProfileString, which uses a standard Windows .INI file. Searching the



All together now...

We're all web developers now, says Tim Anderson, who previews Visual J++, deals with Delphi components, lists VB Script limitations, and serves up the Windows 95 system tray.

Should you care about the internet? Over hyped and under-powered, at least for those suffering modem connections, it would be easy to dismiss it as being of little relevance

boundary between document and application. The final consideration is the sheer momentum of cross-industry support. For anyone planning a new software project, an HTML front-end must

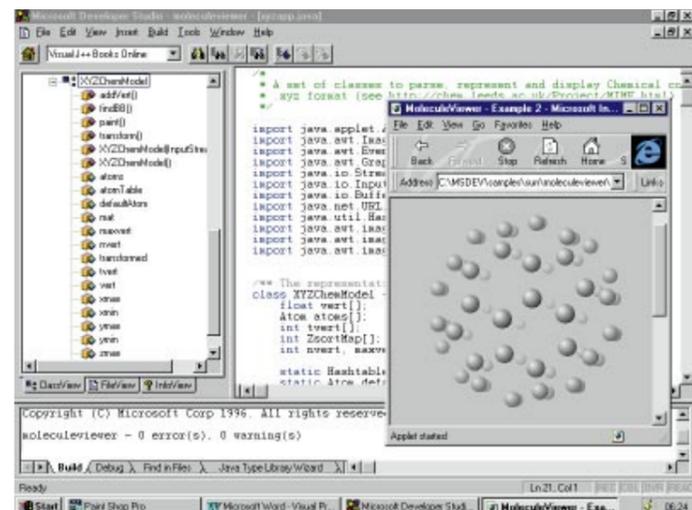
access are all comfortably handled by an intranet. If you buy a new server operating system, you are likely to find web-server software bundled with it, just as workstation software comes with a browser pre-installed. It is irresistible and companies that have not yet done so will inevitably install intranets alongside their Notes, Exchange, client-server or any other systems. Anyone developing software for use over a network should make it intranet-friendly.

So there are good reasons why software companies are falling over themselves to produce web software, and why you will see increasing coverage of web software tools in this column. The catch is that chaos is heading our way, with wars over standards, languages, objects and web servers. But we are all web developers now.

Visual J++ preview

A late beta of Microsoft's Java development tool has arrived in time for a brief preview. The package is hosted by the same Developer Studio used by Visual C++ and shares some of its tools. If you are comfortable with the Visual C++ environment, you will find the transition to Java easy. Each project can be viewed in a hierarchical class view, or file by file, and online documentation is fully integrated.

A clever touch is the resource wizard which converts compiled resources into equivalent Java code. There is still the problem, inherent to Java, that the Abstract Window Toolkit (AWT) Java class library does not support the range of controls available under Windows. Java projects can be either applets, hosted by a browser, or standalone applications which are executed



Visual J++ looks just like its C++ partner and creates cross-platform Java applications

for most developers. Easy, but wrong.

Here are three reasons why, to keep your skills marketable, you have to be web-savvy:

Firstly, HTML is here to stay. It is ironic that Hyper-Text Mark-up Language, designed to add a few simple formatting options for web display, is evolving into the new standard for rich-text documents. The closest previous contender was RTF, or Rich Text Format, used internally by the Windows clipboard. But HTML does forms as well as documents, can host Java applets or, in its Microsoft incarnation, ActiveX controls, and is scriptable with JavaScript or VB Script. It blurs the

be a strong contender, particularly for database applications.

Secondly, users like browsers. Maybe the network computer will catch on, or maybe PCs running Windows will remain dominant. Either way, the browser is going to be the primary user interface. Once users discover they can manage files, run applications, get help and surf the web, all from the comfort of their browser, they will be reluctant to learn other kinds of interface. For developers, that means creating applications which work well in that context.

Thirdly, networks are intranets. Company intranets solve a lot of problems. Publishing documents, email, and database

from the command line using the supplied JVIEW tool.

Microsoft is licensed to produce the 32-bit Windows reference version of the Java virtual machine. The key point of interest is the integration between Java and COM, the object model behind OLE. This enables you to treat Java applets as COM objects and vice-versa. Visual J++ has a type library wizard which creates Java .CLASS files as an interface to ActiveX controls or OLE servers.

You can also expose Java interfaces as COM interfaces and use a tool called JavaReg to register the Java class as an OLE object. This means the Java class becomes accessible to OLE clients like Visual Basic or Delphi, as well as in web applications. Another interesting point is that when a Java applet is running in Internet Explorer, all its public methods and variables automatically become available to Visual Basic Script or JavaScript for scripting, as if the Java applet were an ActiveX control. The snag is that this integration only works on platforms which implement COM, which essentially means Windows.

Visual J++ is a good Java development tool, whether or not you want the OLE features. Like Visual C++, it is not a visual environment in the same way as Visual Basic, Delphi or Optima. Borland's Latte promises something more along those lines. As proof that Microsoft is serious about Java, though, it is more than enough.

Visual Basic

Visual Basic and the System Tray

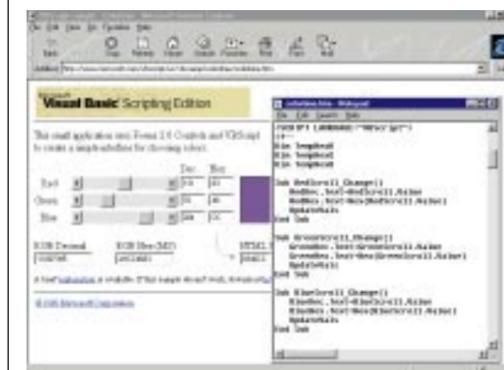
Sagar Shah writes: "I recently moved up to Visual Basic Pro 4.0 (32-bit). While creating some small utility applications, I ran into a problem with the system tray on the taskbar. I cannot find any entries in the manual which tell me how to add to the system tray. I found the function Shell_NotifyIcon in the WIN32APTI.TXT file, and using this function I can add and delete a blank space but nothing more."

The system tray is a corner of the taskbar reserved for status display, in Windows 95 or NT 4.0. It is also called the taskbar notification area. Typically, a utility installs itself as an icon in the system tray, which automatically updates. For example, if you install a modem, a modem icon appears in the system tray when you go

Visual Basic Script — what it does not do

Now that Microsoft Explorer 3.0 is around, VB Script has become useful, particularly on an intranet where you can ensure compatible browsers. Microsoft has also clarified its limitations, some of which are for security reasons while others are merely shortcomings of the language. Here are some of the things VB script cannot do:

- No data access: data access from a web page needs either CGI scripting or special features of particular web servers.
- No debugging: you cannot even step through code.
- No control arrays.
- No classes.
- No OLE automation except the OLE interface to Internet Explorer itself.
- No file operations.
- No types, other than variants.
- No access to system objects like the printer or the clipboard.



Visual Basic Script is fine for scripting, but with major limitations

Many of these limitations can be overcome by using ActiveX controls, which have unrestricted access to the system. A rogue ActiveX could cause lots of problems, which is why system administrators are watching nervously to see if the digital signature scheme for verifying ActiveX controls is successful in preventing viruses.

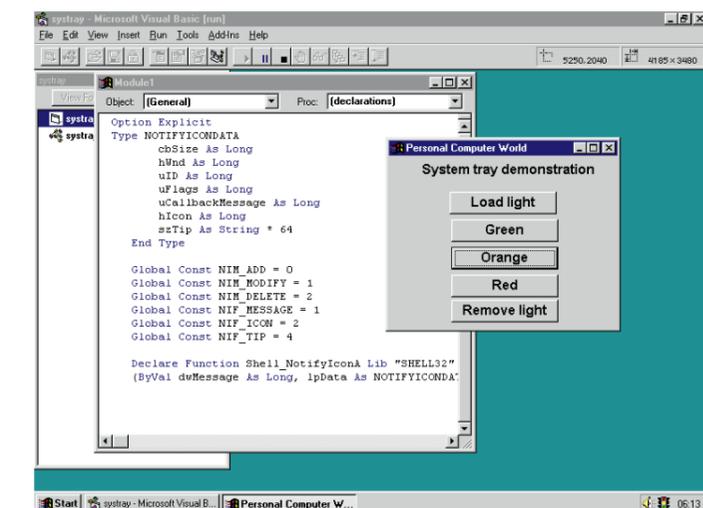
online and flashes when data is sent or received. When the mouse is over the icon, a tooltip shows more detailed information, in this case the number of bytes received. Double-clicking opens a dialogue of further options.

All this is done through the Shell_NotifyIcon function. The declaration in VB is:

```
Declare Function Shell_NotifyIconA Lib "SHELL32" _
(ByVal dwMessage As Long, lpData As NOTIFYICONDATA) As Integer
```

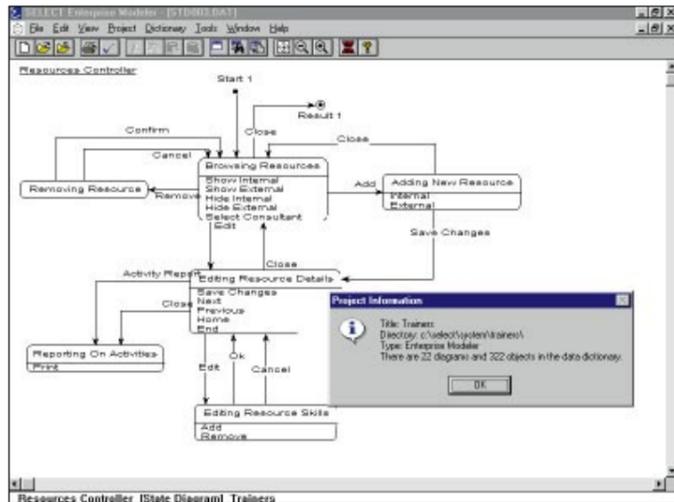
As NOTIFYICONDATA) As Integer

The dwMessage parameter is one of three constants which tell the system to add, delete or modify an icon in the system tray. The second parameter points to a record type which can include an icon handle, a string for the tooltip, a window handle and an application-defined message identifier. The idea is that you define a message handler in your application which responds when the user clicks the icon. The message parameters indicate what type of



VB can easily control icons in the system tray, but responding to mouse clicks is more difficult

SELECT for Visual Basic Enterprise: not for the faint-hearted



DELPHI

Using components in Delphi

When it comes to components, Delphi users have a difficult decision. The most obvious solution is to use VBX controls in Delphi 1.0, or OCX/ActiveX in Delphi 2.0. These component types are abundant, mainly thanks to the popularity of Visual Basic, and now that ActiveX plays a key role in Microsoft's internet strategy, you can expect them to proliferate.

Often, the component you want is only available in VBX or ActiveX form. The second advantage of these Microsoft standards is that they can be hosted by several different tools. For example, a VBX can be used by Delphi, Visual Basic and Visual C++, provided you use the 16-bit versions.

Unfortunately, these benefits are balanced by several problems, one of which is compatibility. Some VBX vendors assume that their controls will be hosted by Visual Basic and that not all their features work in Delphi. In particular, data-bound VBXs lose their data-aware functions in Delphi, if they

mouse event has occurred. Your application window can be hidden so that it only pops up when needed.

Unfortunately, not all this functionality is available from Visual Basic. You can easily make an icon appear and set the text for the tooltip. But since VB has no way to intercept custom messages, you cannot make a dialogue appear when the user clicks the icon.

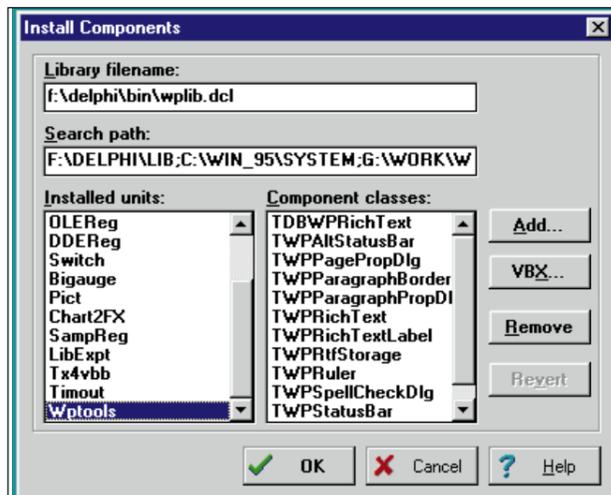
The way round this would be to use a control like MessageBlaster that adds this feature to Visual Basic. If you are content with more limited features, you can easily create an application like the example on our cover-mounted CD.

Note that you need to include an icon handle in the NOTIFYICONDATA record. You can obtain this in several ways: one is to use the icon or picture of a form or control; another is the LoadIcon API function; or there is the LoadResPicture function which works on icons stored in a resource file. The technique used in the example is to call LoadPicture to place an icon into an invisible image control, and then use its picture property to obtain an icon handle.

versions work with Forte and C++.

Unlike other tools which you can pick up and drop as required, committing to SELECT is almost a way of life. It replaces the free-and-easy VB style with a rigorous development process.

The best advice to those considering a system such as this is a careful evaluation procedure including full consultation with others actually using the system.



Delphi components are installed by rebuilding the component library. But which component type is best?

SELECT for Visual Basic

Bridging the gap between those who theorise about business object models and actual working systems is no trivial matter. SELECT is a set of tools based on object models developed by J Rumbaugh and Ivor Jacobson. There is a modelling tool, from which you can generate both SQL code and a set of Visual Basic forms and classes. There is also an automatic documentation feature that works with MS Word. SELECT needs the VB Enterprise edition, while other

Delphi Component Options

Component type	Pros	Cons
VBX, ActiveX, OCX	<ul style="list-style-type: none"> Widely available Shared between applications 	<ul style="list-style-type: none"> Not always compatible Cannot easily create in Delphi Version control problems
VCL	<ul style="list-style-type: none"> Best performance Can create or customise in Delphi No version control problems 	<ul style="list-style-type: none"> May not be available Not shared between applications

A book for visual programming

Rapid Development by Steve McConnell

Software projects are notorious for going wrong. The concept of rapid application development seems to offer a solution by using tools that dramatically cut programming time, but there are still plenty of problems.

Whereas his previous book, *Code Complete*, studied the detail of coding and debugging an application, this one analyses the whole development process, exploring common reasons for failure and offering tips for a successful strategy.

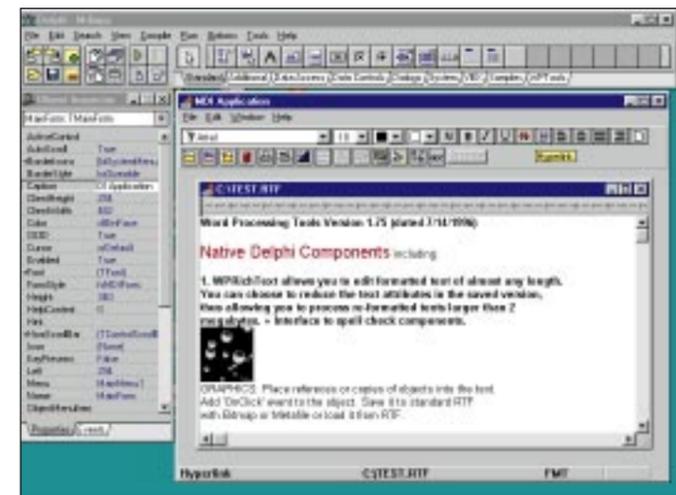
Really, the book is mis-titled. It is not only about rapid development, but any kind of software project. If your emerging application is suffering from feature-creep, unrealistic schedules, tools that do not work, problems integrating work from different team members, heavy overtime and low morale, then McConnell will tell you why, how to rescue the project, and how to avoid repeating history next time around.

Although the author does not focus on specific languages, there are some good observations about visual programming. The key advantage of products like Visual Basic and Delphi is reduced coding time, but there are associated risks. The main ones are over-estimated productivity savings, failure to scale well as the project expands, and the encouragement of sloppy programming. The answer is to be aware of the limitations of a particular tool and to allow time for working around them. Another key point is that the larger the project, the less time is spent on coding as opposed to other elements like designing and testing. Therefore, visual tools yield their biggest benefits on small projects.

There are plenty of case studies and examples in the book, although it is a shame that most are invented rather than real-life projects. It is repetitive in places and at times dispiriting, as the author tends to focus on how *not* to do things. But this is excellent reading for developers, especially those who work in a team.

work at all. Another problem is getting at certain VBX properties, such as string properties which Delphi converts to Pascal string types, cheerfully truncating them if needs be. OCX controls present a different set of problems, but since there are several different levels of OCX compliance, compatibility is by no means guaranteed. Finally, you cannot easily build either VBX or OCX components in Delphi. Borland has considered providing an OCX wizard to convert Delphi native components, but so far it has not emerged.

To avoid the problems with VBX and ActiveX you can extend Delphi's Visual Component Library either by coding your own custom components or obtaining additions from a third party. Coding your own is fairly easy, although harder than straight Delphi programming. The advantage of native components is full compatibility and the most efficient interface between the component and the rest of your application, and therefore the best performance. The disadvantages are that the component can only be used in Delphi, and if you have several applications using the same component, inefficient use of disk space because the whole component is compiled into every executable you create. If this



A third-party has produced a full-featured rich text control as a native VCL component

becomes a major problem, you can move parts of the code into dynamic linked libraries so it is shared between applications.

Frankly, in most projects a VCL component is preferable where available. Availability is the problem, but the situation is improving. Delphi 2.0 includes the QuickReport component which is vastly more efficient than wheeling in ReportSmith.

Crystal Reports 5.0 also comes with a VCL, although the huge Crystal DLLs do not

compare with QuickReport for efficiency. Other leading component vendors have been slow to support Delphi but may find themselves losing sales to smaller upstarts as a result. A good example is WPTools, a VCL-implementation of a rich text edit control. This one works in 16-bit Delphi as well as Delphi 2.0 and is not just a wrapper for the Windows 95 common RTF control: it supports large documents, fonts, styles, images and hypertext links, and full source is available to registered customers. Look out for a full assessment in a future article.

Delphi 2.0 16-bit?

Borland is considering an update to 16-bit Delphi, and is using the web to survey developers about what features they would like and whether they would buy it. TI would prefer to see resources go to developing Delphi for 32-bit Windows. There is still a big market for 16-bit development, but I sense that the tide has turned. Companies have both NT 4.0 and Windows 95 from which to choose and web developments favour 32-bit Windows. It is too late for 16-bit Delphi 2.0, and Borland risks getting it wrong again.

PCW Contacts

Tim Anderson welcomes your Visual Programming comments and tips. He can be contacted at the usual PCW address or as freer@cix.compulink.co.uk

Visual J++ (price not yet announced) from Microsoft 0345 002000
 WP Tools is shareware from Julian Ziersch, 100744.2101@compuserve.com
 SELECT VB Enterprise Edition £2,995. SELECT Software 01242 229700
 Rapid Development by Steve McConnell, (Microsoft Press). £32.49 from Computer Manuals 0121 706 6000