Strands of DNA

What's in store for the future of programming? Java is of course a hot development topic, and Microsoft, not to be outdone, is busy developing the intriguingly-named Windows DNA.

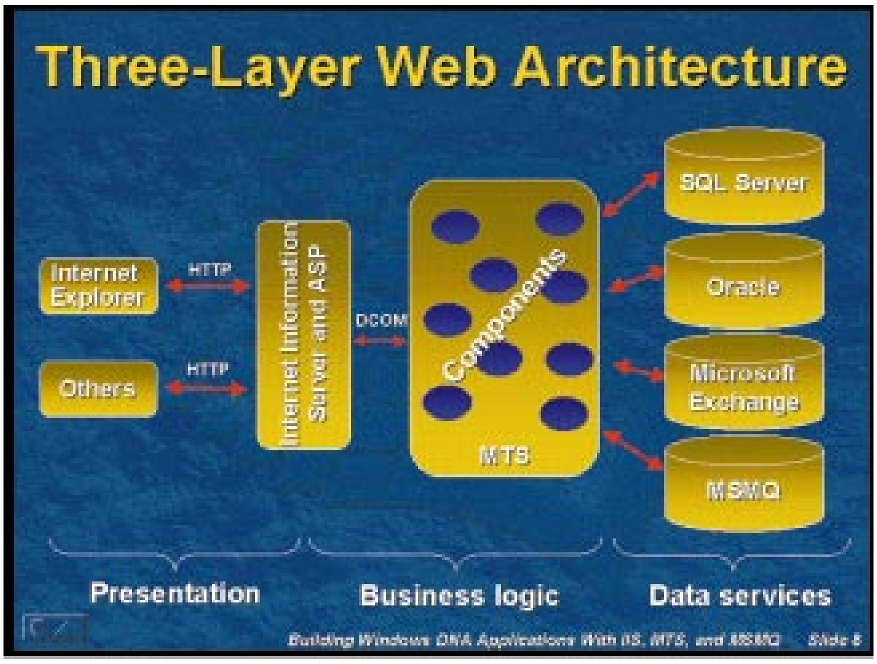
There is only one hot topic in the development world just now, and it begins with J. All the big questions (how to do internet development, distributed objects, which language to use, which operating system to run) are affected by Java and its future.

At the time of writing, two interesting events have occurred. One is Microsoft's PDC (Professional Developer's Conference) held in San Diego, from which I have just returned. The PDC is where Microsoft unveils its software development strategy for the coming year. The other event is that Sun Microsystems initiated legal action against Microsoft for alleged breach of contract in respect of Java's implementation in Internet Explorer 4.0.

This column is not about industry politics, though, but about practical issues facing developers, and Java is just such an issue. You have a software project: should it be implemented in Java, Visual Basic or C++? A decision in favour of Java is particularly fundamental. In effect, you are no longer creating a Windows application but a Java one. Java has become your platform. You will very likely continue to run it on Windows. But consider what difference it makes when you next replace your computer. The priority may be not how well it runs Windows, but how well it runs Java.

Sun owns Java, and Java is developed on Sun computers running Solaris. In other words, the best platform for Java may not be a Windows system at all. It is not surprising that Microsoft and Sun don't see eye to eye on the subject.

There is an uneasy alliance of software and hardware companies, including Sun, Netscape and Oracle, which is sometimes described as "Anything but Microsoft". The theme of the PDC could equally be called "Anything but Java".



Three-tier web applications the Microsoft way involve Internet Explorer and COM components. No sign of the J word

Windows DNA

Microsoft has a Java development tool and a Java virtual machine, but it is a company with nothing to gain and everything to lose from Java's success. The focus of the PDC was a new framework called Windows DNA (Distributed Internet Applications Architecture). It is hard to summarise what Microsoft means by "Windows DNA" but the gist of it is: applications using Internet Explorer, clients with Windows NT and Internet Information Server at the server end. The client application uses HTML beefed up by scripting and the new features of Dynamic HTML.

The real work of the application is handled by application components running on the server or elsewhere on the network. Communication between components, including dynamic database access, is through COM, the ActiveX component model. In this scenario Java still has a place, but as another way to write a client-side control or a server-side component.

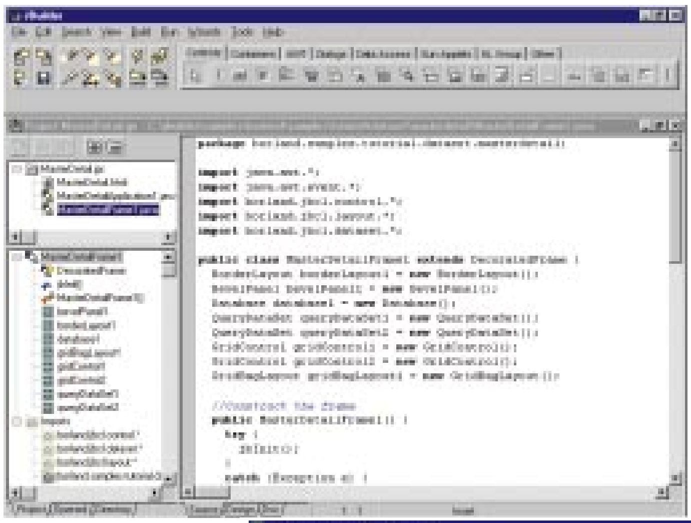
Much of the infrastructure for writing Windows DNA applications is already in place. A key part of it is Microsoft Transaction Server, which provides intelligent management of COM objects. Another important element is the Active Directory, coming in Windows NT 5.0, which makes sense of user management.

As an aside: from what we saw at the

How to get started with Java

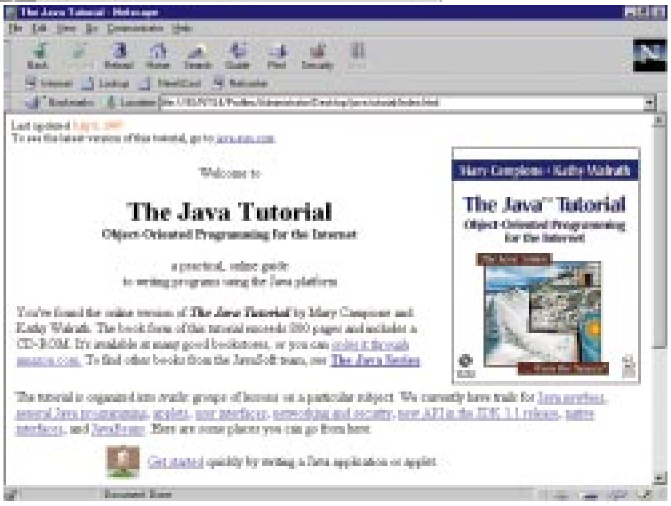
You will love coding with Java. It is clean, object-orientated, productive, powerful and many other things (as fans of Java are keen to point out). Java is not all plain sailing, though. Here are some tips for getting started.

- Visit www.java.sun.com and download the latest Java Development kit (JDK) and the online Java tutorial. Both are free. The advantages of starting with the JDK are that you have the most up-to-date implementation, you achieve a good understanding of Java basics uncluttered by the characteristics of a particular IDE, and the command-line tools are less system-hungry than other Java tools.
- Understand the two kinds of Java project. Applications run standalone (using the Java runtime library) like a Visual Basic program. Applets run in a browser, or in Sun's AppletViewer. Applications have unrestricted access to files and other system resources, whereas applets are limited, for security reasons, in what they can do. For instance, applets cannot use the JDBC-ODBC bridge to access databases, since ODBC drivers run in native code on the client machine.
- Expect a culture shock if you are a Windows developer. For instance, Java tutorials are full of stuff about layout managers of which a Windows developer has never heard. This is the price you pay for Java's cross-platform approach. Positioning of controls needs to be relative, not absolute.
- There is a new generation of Java development tools which use JavaBean components for fast visual development. Examples are Borland JBuilder, IBM VisualAge Java, PowerSoft PowerJ and Symantec Visual Café 2.0. Despite many excellent features, all these products are system hungry and not always stable. I would recommend Windows NT with 64Mb RAM or more as a minimum.
- Be clear about why you are using Java. It is outstanding for cross-platform work, for learning object-orientated programming, for distributed applications and for riding what is probably the wave of the future. It is currently poor for those applications where performance is critical, or those which need a rich graphical interface or multimedia capabilities.
- Java deployment can be as easy as putting an applet on a web server. On the other hand, be warned that browsers vary in how they implement Java. If you want to distribute a Java application, use Sun's Java Runtime Environment (JRE) to set up a Java Virtual Machine along with your application.


JBuilder
File Edit Search View Build Run Wizards Tools Help
Controls Containers AWT Dialogs Data Access Run Applets KL Group Other
Source Design Doc

```
package borland.samples.tutorial.dataset.masterdetail;

import java.awt.*;
import java.awt.event.*;
import borland.jbcl.control.*;
import borland.jbcl.layout.*;
import borland.jbcl.dataset.*;

public class MasterDetailFrame1 extends DecoratedFrame {
  BorderLayout borderLayout1 = new BorderLayout();
  BevelPanel bevelPanel1 = new BevelPanel();
  Database database1 = new Database();
  QueryDataSet queryDataSet1 = new QueryDataSet();
  QueryDataSet queryDataSet2 = new QueryDataSet();
  GridControl gridControl1 = new GridControl();
  GridControl gridControl2 = new GridControl();
  GridBagLayout gridBagLayout1 = new GridBagLayout();

  //Construct the frame
  public MasterDetailFrame1() {
    try {
      jbInit();
    }
    catch (Exception e) {
```

Left A package like JBuilder is slicker than using command-line tools, but beware instability and system requirements


The Java Tutorial - Netscape
File Edit View Go Communicator Help
Back Reload Home Search Guide Print Security
Bookmarks Location
Document: Done

Welcome to

The Java Tutorial

Object-Oriented Programming for the Internet

a practical, online guide to writing programs using the Java platform

You've found the online version of *The Java Tutorial* by Mary Campione and Kathy Walrath. The book form of this tutorial exceeds 800 pages and includes a CD-ROM. It's available at many good bookstores, or you can order it through amazon.com. To find other books from the JavaSoft team, see The Java Series.

The tutorial is organized into *trails*: groups of lessons on a particular subject. We currently have trails for Java products, general Java programming, applets, user interfaces, networking and security, new API in the JDK 1.1 release, native interfaces, and JavaBeans. Here are some places you can go from here:

Get started quickly by writing a Java application or applet.

Below Want to learn Java? This online tutorial is both up-to-date and free to download

PDC, NT 5.0 looks excellent: easier to manage, more scalable, and plugs the gap between NT and Windows 95/98 by including plug-and-play, better games and multimedia support.

Will DNA do it?

Windows DNA is packed with clever ideas, but will it work? Undoubtedly it can be *made* to work, and it does have advantages. One is that performance may be better than a Java-based system, because most processing is carried out by compiled components that either run on the server or are only downloaded once. Another is that migration of existing code may be smoother because all you need to do is expose its interface through COM. Tools like Visual Basic, Visual C++ and Delphi make this easy.

There are several problems, one of which is the complexity of building applications which incorporate so many diverse elements — at a minimum, HTML, scripting, and COM components. Anyone who has worked with moderately complicated web applications will know that using HTML as an application interface is a maintenance nightmare. Also, COM has messy aspects, in particular the need to register client-side components.

The Windows registry is hard to manage. COM components run natively, which is good for performance but makes integration with Mac or Unix systems hard. The solution is to use COM only on the server, putting all the client-side logic into HTML, scripting, or indeed Java. It is not ideal though.

Another factor is that Netscape Navigator and Internet Explorer have many differences. These include how Dynamic HTML is handled and Navigator's lack of support for ActiveX. If users run Navigator, it is difficult to create full-featured Windows DNA applications. Overall, my suspicion is that building good, manageable Windows DNA applications will be possible, but hard.

What about Java?

Some of these considerations, like the need to incorporate HTML and scripting into your application, also apply to Java. But Java still has performance problems.

Where Java shines is for distributed or web applications. The language itself is elegant, productive and likely to result in applications that are easier to maintain than

p315 ➢

Anderson answers…

Q I am using Delphi and have designed a database form with several DBEdit fields on it. When the user selects one of these fields I want its background to change colour. The problem arises when I try to create common event handlers for all the DBEdit components on the form. I have tried to change the background colour of the component using either Sender or ActiveControl [as the code in Listing 1 illustrates].

Unfortunately, the only thing these routines succeed in doing is to alter the colour of the form and not the DBEdit.

Kevin Parsons

A This question illustrates several of Delphi's features. Event procedures have a Sender parameter, making available the object which called the event. So why can't you change its Color property? The answer is that Sender has a type of TObject, which doesn't have a Color property. Neither does ActiveControl, which is of type TWinControl. In this example, the event procedure is a method of TForm1, so Delphi correctly changes the Color property of TForm1 instead.

The solution is to use a typecast. This tells Delphi that Sender is actually a TEdit control, which does have a Color property. The amended code is like that shown in Listing 2. The line "if sender is TDBEdit" is not essential in this case, because you can ensure that the event is only attached to DBEdit controls. It is useful if you have code which you want to attach to more than one type of object.

Listing 1

```
procedure TForm1.DBEdit1Enter (Sender :TObject);
begin
  with Sender do Color := clRed;
end;
```

Listing 2

```
procedure TForm1.DBEdit1Enter(Sender: TObject);
var
thisEdit: TDBEdit;
begin
if sender is TDBEdit then
 begin
 thisEdit := TDBEdit(Sender);
 thisEdit.Color := clRed;
 end;
end;
```

Q I am still learning Visual Basic and use the v5.0 learning edition. I am trying to write a program that will make heavy use of playing WAV files, but when I use the OLE function the WAV file plays, although there is an error message in hex format.

Could you please tell me how to play a WAV or MIDI file when an event happens? I can only play one, by double-clicking on the icon.

Nathan W.

A The easiest way to play WAV files in VB is by using the Multimedia control. Set the filename to point to the WAV file, and set the command property to Play in your code. I suspect from your question that the Multimedia control is not supplied with the Learning Edition. In that case, you can use the OLE control.

First, link or embed the WAV file and then play it using the DoVerb method. Playing is the default action of a sound, so you can use

```
OLE1.DoVerb (vbOLEPrimary)
```

You may feel all this is over-the-top just to play a sound file, and I would not quibble with that. The solution yet again is an API function. This is the declare shown in Listing 3, so now you can write code like that shown in Listing 4. This also has the advantage that no applet window opens, the sound just plays. You should look up the function in an API help file to explore the parameter options available.

Listing 3

```
Declare Function PlaySound Lib "winmm.dll" Alias "PlaySoundA" (ByVal lpszName As ➢
String, ByVal hModule As Long, ByVal dwFlags As Long) As Long
```

[Key: ➢ listing continued on next line]

Listing 4

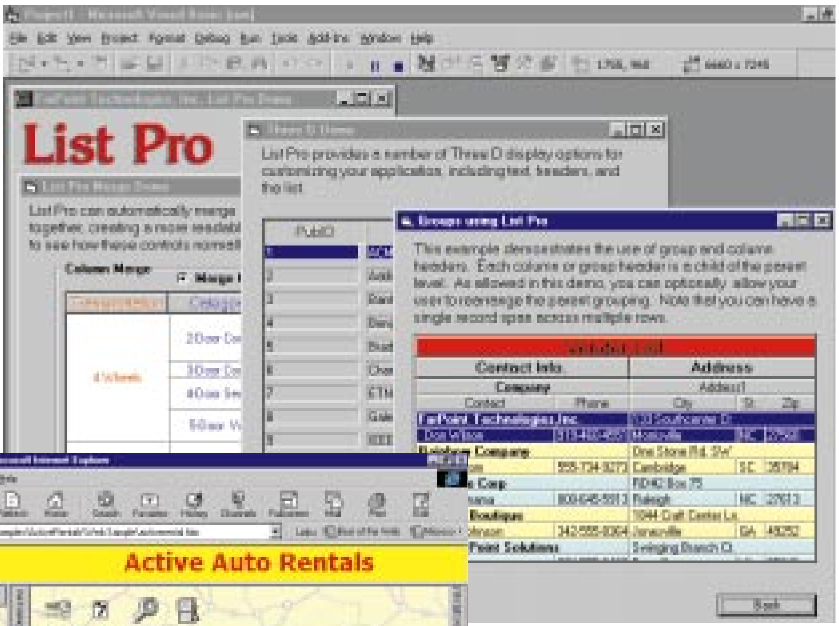
```
Dim lRetVal As Long
lRetVal = PlaySound("C:\WINDOWS\MEDIA\CHIMES.WAV", 0, 0)
```

scripts, Dynamic HTML or even Visual Basic. You can use Java on the client, on the server and across diverse platforms. Remote Method Invocation (RMI) makes it possible for Java objects to communicate across the network. Microsoft's competitors love Java because they are no longer forced to support Windows. Like Windows, Java is a proprietary technology owned by a single company (in this case, Sun) so it is possible that at some later date this ownership may, too, be resented. But at present, Sun's open approach to managing Java is winning industry-wide support, which gives developers confidence that investment in Java will not be wasted. Time to learn Java, then: see "How to get started with Java", page 313, for some tips and problems.

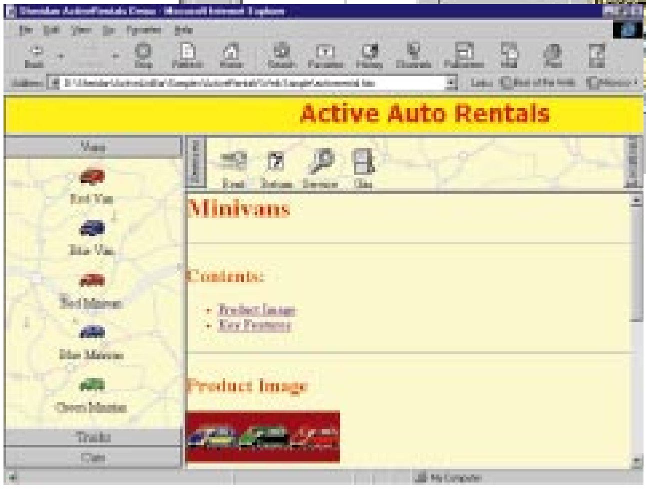
Get listed

Java is exciting stuff, but for many, Windows is where the real work gets done. Two recently released ActiveX components will help. FarPoint's List Pro is a model of how an add-on should be done. It is an update of an earlier product, called Aware/VBX, and comes in four forms, all in one box: 16- and 32-bit DLLs, VBX and ActiveX.

p316 ➢



FarPoint ListPro is a comprehensive solution to listbox problems



Sheridan ActiveListBar makes a fun way to navigate a web site

I can recommend List Pro. There is a list control and a combo box, with many advantages over the standard Visual Basic or Delphi equivalents. Both claim to handle up to two billion list items. Another key feature is the ability to merge cells in the list, to avoid the continuance of repeating values. There are hundreds of properties, enabling you to customise the appearance of the list. You can organise items into groups. You can also set font and colour properties for individual cells and include both text and graphics. Also in the box is a detailed printed manual.

Sheridan's ActiveListBar is more specialist than List Pro. If you have Office 97 you will already know the Outlook Bar (a scrolling toolbar which is also divided into groups, like a tabbed dialog). ActiveListBar lets you put one of these in your own application or web site. In fact, its natural home *is* a web page and Sheridan supplies a 138Kb signed CAB file for easy deployment, the main snag being that users need 32-bit Windows and Internet Explorer, or an ActiveX plug-in.

You can have sound and 3D effects as the user selects icons. It makes a nice navigation tool, but I'm less happy about Sheridan's presentation. The box is smart, but there is nothing inside except a few leaflets and a CD. Apparently, Sheridan no longer produces printed documentation for its products. Another gripe is that many of the example files do not work unless you have Active ThreeD (another Sheridan product) installed. But it is worth considering if you like the effect of the Outlook Bar.

Books in brief

■ ***Java 1.1 Unleashed* by Michael Morrison and others**

Continuing the Java theme, here is a 1,300-page handbook which explains all, from the basics to remote objects and the CORBA industry standard for distributed objects. Thirty-one contributing authors are listed. This is a good way to get an up-to-date book published quickly, with chapters written by specialists, but the downside is uneven style and repetition. For instance: the chapter on Remote Method Invocation overlaps with a one by a different author, about remote objects; and layout managers are introduced twice, in chapters on Applets and AWT. By way of compensation, there is detailed explanation with examples on most Java topics. The book is a fine reference tool but too disjointed to give the overall picture of how to design and build a Java application.

■ ***Delphi 3 Client/Server Developer's Guide* by Ken Henderson**

I suspect there are too many books attempting to give comprehensive coverage of complex developer tools and too few which focus on a particular area. That means many shallow and repetitive titles. This one, by contrast, is focused on client/server database development with Delphi. It is a topical subject now that PCs capable of running server databases are commonplace.

The book is in four sections. The first provides an overview of both Delphi and client/server database design. Next is a detailed tutorial covering entity modelling, form design and reporting. The third section has chapters specific to popular databases, Microsoft and Sybase SQL Server, Oracle and Interbase. Finally, the fourth part tackles advanced topics including multi-tier applications, advanced SQL and ActiveX internet applications. An appendix has useful tips for migrants from Visual Basic and Delphi. This is a useful, common-sense book suitable for Delphi developers with no previous knowledge of client/server issues.

• *See PCW Contacts box for prices and availability.*

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

Computer Manuals 0121 706 6000 for books reviewed here: ***Java 1.1 Unleashed*** (£54.95 book and CD); ***Delphi 3 Client/Server Developer's Guide*** (£54.95 book and CD).
Contemporary Software 01344 873434 for **FarPoint ListPro** (£175 ex VAT) and **Sheridan ActiveListBar** (£99 ex VAT).
Java information or to download the JDK visit www.java.sun.com or www.javasoft.com

Patch works

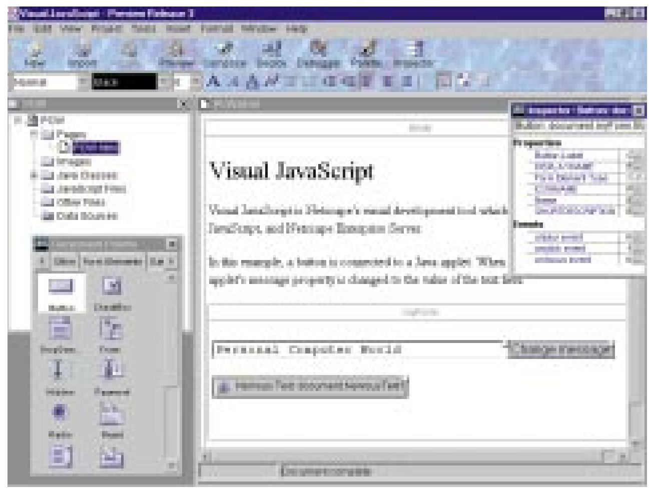
Tim Anderson reviews the latest patches for Visual Basic and Delphi. Netscape's Visual JavaScript offers a couple of good features and CodeBase for Java is a nice little runner.

Microsoft's Service Pack 1 was a disappointment for users of Visual Basic 5.0. Some painful and easily reproduced bugs were left unfixed, while more mysterious problems with instability in the IDE seemed to be little improved.

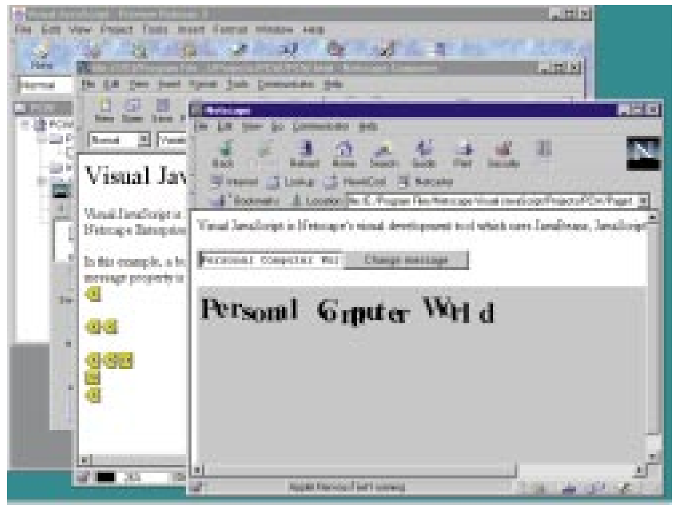
Service Pack 2.0 (SP2), which is available from the Microsoft web site, finally provides some fixes. You can now change the font of several objects at once without it crashing. The problems which have been sorted include add-ins causing menu corruption, overflow errors when compiling, crashes with user controls, and errors displaying help for custom controls.

Unfortunately, one control has been broken in the process. It appears that the Internet Transfer Control no longer supports usernames and passwords, and Microsoft suggests reinstalling the old one.

Another thing to watch out for is that applications compiled under SP2 will not work with the old runtime DLL (MSVBM50.DLL). More worrying is that some users have reported problems with applications or ActiveX DLLs, compiled under the earlier version, not working properly with the new runtime. If you have the source code, recompilation is recommended. It is well worth it, though: so far, SP2 has proved much more reliable.



Left Visual JavaScript is an application builder for web pages and Java or JavaScript components. You can see the project manager, the component palette, the property inspector, and the HTML page being edited



Left Visual JavaScript slots neatly into Netscape Communicator, making use of Navigator for preview and Composer for page editing

Service Pack 2 is not only a bug-fix. The new patch makes Forms, UserControls, UserDocuments and ActiveX Designers thread-safe, so you can use Apartment Model threading in ActiveX projects even if they contain forms and controls — this was not previously possible. This will be of most value to those building client-server systems with Visual Basic.

A patch for Delphi, too

Borland has released Delphi 3.01. Everyone wants to know what it fixes, but the information is hard to discover. Software companies are often reticent about bug-fixes, presumably because they would rather not acknowledge the existence of bugs in their product. This attitude makes no sense at all for a developer product.

Microsoft is better in this respect. All Borland has said is that the help files are more reliable, although it took me no time at all to get the usual "This topic does not exist" message. The best plan is to create a full-text index and use that.

Rumour has it that other changes include better COM and DCOM support, an updated QuickReport component, significant improvements to the MIDAS distributed database technology in the client-server version, and who knows what else. My copy arrived this morning, so I will be exploring it with interest.

JavaScript gets visual

JavaScript is Netscape's browser scripting language and, apart from the name, has no more to do with Java than Microsoft's VBScript. But in both cases there is a relationship with Java, since you can use scripts to control Java applets (and soon, Java Beans).

The problem with VBScript is that at this stage in the browser wars it looks unlikely to be supported by anyone but Microsoft, whereas JavaScript is already present in Internet Explorer (under the name JScript) and is gaining ground as a standard. For client-side scripting, JavaScript is the one to use. Visual JavaScript is a new tool for developing interactive web sites, in contrast to Netscape Composer which is for laying out static pages. Although it works in a different way, its target market is similar to Microsoft's Visual Interdev; and just as Interdev ties in to Internet Information Server, Visual JavaScript is essentially a tool for Netscape's Enterprise server and uses its features for database connectivity.

Visual JavaScript is, itself, a Java application and runs with Symantec's Just-In-Time compiler. It still runs frustratingly slowly, at least in the Preview 3 release which I tried. The main elements are a project manager where items such as HTML pages, Java classes, images and data sources are listed, an HTML editor for assembling the application, and a component palette. This palette can contain Java applets or beans, JavaScript components, or CORBA objects.

ActiveX is nowhere to be seen. To construct an application, you drag components onto an HTML page. Double-clicking a component opens a property editor. Each object has a small tab at the top right: click here and drag to another object to open the connection builder, which lets you write code that is triggered by one of the events supported by the source object.

An interesting feature is a database wizard which sets up a single record or master-detail form by linking to a data source supplied by Netscape's Enterprise server. You can apply further editing in Netscape Composer and preview your work in Navigator, so making good use of the various parts of the Communicator suite. A strong point is that if your code has errors, the debugger will pop up within Navigator and show the problem line.

Visual JavaScript is a nice idea, but judging by this preview there are plenty of limitations. It's a good utility for assembling pre-built components, but sadly it's a poor coding environment. The Java-based environment is sluggish to use and crude compared with native Windows applications: there is neither syntax-highlighting nor context-sensitive help. Dependence on Netscape's Enterprise server will limit its appeal. But Visual JavaScript shows the potential of using HTML pages in the same way that Visual Basic developers use forms, with JavaBeans taking the place of ActiveX components.

The technology works, it's just a matter of improving performance and providing slicker tools. An interesting question is how much application logic developers need to put into JavaScript, rather than building all the increasingly functionality into a Java applet which can simply be inserted on a web page? An all-Java solution is preferable, if you can make it work.

CodeBase: xBase meets Java

In the era of Microsoft Office, it is nice to know that some vendors still believe in small applications. I am referring to CodeBase, the long-established database library from Sequiter Software, distributed in the UK by Highlander Software (*see "PCW Contacts", p310*). It still uses the venerable DBF file format, but nevertheless, CodeBase is moving with the times. Version 6.0 introduced a client-server version, with the server executable occupying just 350Kb and running on Windows 95 or NT. Version 6.2 added Java support, while 6.3 offers ActiveX data-bound controls to supplement the existing 16-bit VBX set.

CodeBase is a solution for those who would rather avoid database technologies like ODBC, the Borland Database Engine or Microsoft JET. Although these systems are rich in features and, in theory, easy to work with, they introduce extra layers of software, upping runtime requirements and making it difficult to troubleshoot problems. CodeBase does not use any of them. Instead, it provides its own library of database functions, for C, C++, Delphi, Visual Basic and now Java.

Unfortunately, the price of this simplicity is that the built-in database features of these environments do not work with CodeBase: bound controls in Delphi are not supported except via the special CodeBase-supplied components, for example. The benefit is fast data access and easy runtime deployment.

I was particularly interested in the Java version. CodeBase now comes on one CD which supports all the above-mentioned environments. (Unix and Mac versions are separate purchases.) Java support comes in the form of around 30 small classes (occupying about 50Kb) which let you connect to the CodeBase server. Data access is achieved by constructing a Code4 object which represents the connection, then Data4 objects which represent tables, and Field4 objects and their descendants which represent fields. Fig 1 (*above*) illustrates a minimal example of searching an address table.

Using CodeBase, it was not difficult to create a simple applet for looking (*p310*➢)

Fig 1 Searching a database with CodeBase Java

```
void findTel() {
try {
Code4 client = new Code4();
client.connect(“wollaton”, 23165, “username”, “password”);
Data4 dataFile = new Data4(client, “C:\\ADDRESS\\ADDRESS”);
Field4stringBuffer s_name = new Field4stringBuffer(dataFile,”SURNAME”);
Field4stringBuffer f_tel = new Field4stringBuffer(dataFile,”TELEPHONE”);
int rc=dataFile.top();
  dataFile.select(“SURNAME”); /* select SURNAME index */
  rc=dataFile.seek(“Jones”);
   if (rc==Status4.SUCCESS) dataLabel.setText(“Tel: “ + f_tel.contents.toString());
        else dataLabel.setText(“Name not found”);
dataFile.close();
}
catch etc…
```

p309 ➢

Anderson answers…

Using PostMessage to close an application

```
Private Sub Command1_Click()
Dim hwndTarget As Long
Dim lRetVal As Long

hwndTarget = FindWindow(vbNullString, “Registry Editor”)

If hwndTarget <> 0 Then ‘ found it

lRetVal = PostMessage(hwndTarget, WM_CLOSE, 0&, 0&)

    If lRetVal = 0 Then
    MsgBox “Error posting messge”
    End If

Else

MsgBox “The registry editor is not open”

End If
```

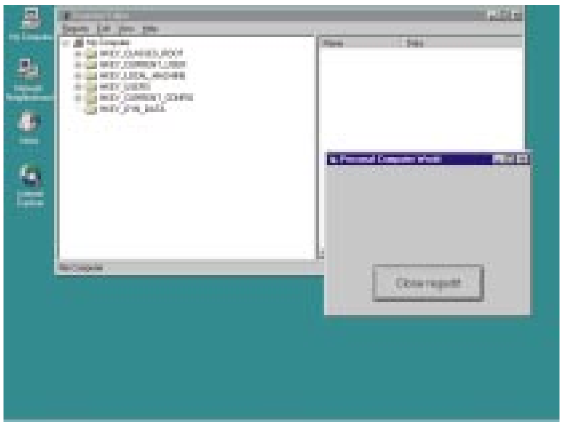


Fig 2 (above & left) **Closing Regedit from Visual Basic. Take my word for it: one click and it’s gone**

Q I am trying to write a program that uses a timer and a form and scans for the Registry Editor program. When it finds it, the program is to send a WM_CLOSE command to it via the SendMessage API call. I have seen the PostMessage API call used but I think the SendMessage call is the correct one. Am I right? The major problem I am facing is a lack of API knowledge. I need to get the classname of RegEdit and the handle, and then I can send the WM_CLOSE command to it. Using the Spy++ program I have found out the classname on the Regedit Program, yet am unable to get the SendMessage call to work. It just sits there doing nothing.

Philip Bevan

A Windows is a message-based operating system. The message API functions are useful to Visual Basic programmers and are easy to use. Whichever message you use, the starting point is the same. You have to find the handle, which is an ID number that identifies the target window. If this is a window in your VB application, you can simply inspect the **hWnd** (Windows handle) property of the form or control. Note that controls such as the PictureBox or ListBox are, themselves, windows with their own windows handle. If it is a window belonging to another application, finding the handle takes a little more work.

A useful function is **FindWindow**, which searches for a top-level window matching either (or both) the classname or the caption which you specify. Take care, though: both these characteristics can change with different versions of the application. Another catch is that MDI applications, like Word and Excel, include a document name in the window caption when a document is maximised.

Once you have the right handle, use **PostMessage** or **SendMessage** to dispatch the message of your choice. The key difference between these two functions is that PostMessage puts a message in the queue for Windows to process when it is ready to do so, whereas SendMessage is immediate. You will often not notice the difference, but it is significant for your code.

With PostMessage, your application will have moved on to other things by the time the message is processed, whereas with SendMessage, no further code will execute until the function has returned. If you want to check the return value, you must use SendMessage. The return value of PostMessage merely tells you whether the message was dispatched successfully, not whether it achieved its aim.

In general, if you are sending messages to other windows in your application, you should use SendMessage. If you are sending messages to windows in other applications, use PostMessage unless you need SendMessage for a specific reason.

There is one final factor to consider. In 32-bit Windows, if you use SendMessage for a window whose thread of execution has hung, your application will hang, too. There is a separate function which allows for this possibility, **SendMessageTimeout**, and times out if there is a problem. See also Fig 2.

Q How do I use the VB ImageList control? I’m totally confused.

Mark Hankison

A The ImageList control, which is one of the Windows 95 common controls, is a way of storing images for use by other controls. There are other ways to store images, such as loading them from disk as needed, or putting them in invisible image controls. These techniques are slow and difficult to manage, however, particularly where you need a lot of images — for an animation, say, or for a toolbar with many buttons, each of which has different images for different states.

Visual Basic has better options. The ImageList control stores separate bitmaps in an invisible list. The Picture Clip control

p310 ➢

stores a single large bitmap and lets you access sections as if they were separate images, and you can store images in a resource file and access them by a resource ID. Several VB controls require you to use an ImageList to supply their bitmaps; these include the Toolbar and the Treeview controls. For example, this is how you use the ImageList with a Toolbar:

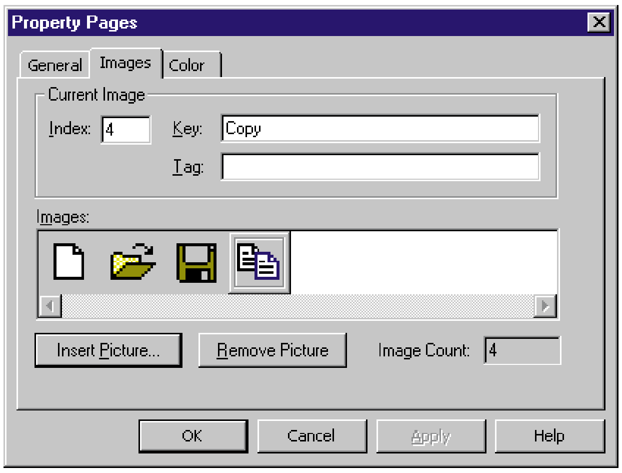
1. Place a Toolbar and an ImageList on a form. Right-click the image, choose **Properties**, click the **Images** tab, and use **Insert Picture** to add images (Fig 3). A useful selection for toolbars is in the **Graphics\Bitmaps\tlbr_w95** sub-directory of a full VB installation.

• *Tip: When you add images, include a unique descriptive word in the Key property. Otherwise, you will have to refer to the image by Index, which might change if you later amend the image list*.

2. Select the **Toolbar**, right-click, choose **Properties/General** tab, and set the ImageList drop-down to point to the ImageList control (Fig 4).

3. Next, click the **Buttons** tab and click the **Insert Button**. In the **Image** property, enter the key word you entered for the image in the ImageList (Fig 5). Alternatively, you could enter an index number.

4. Run the application to see your toolbar (Fig 6). ■


Property Pages — General | Images | Color

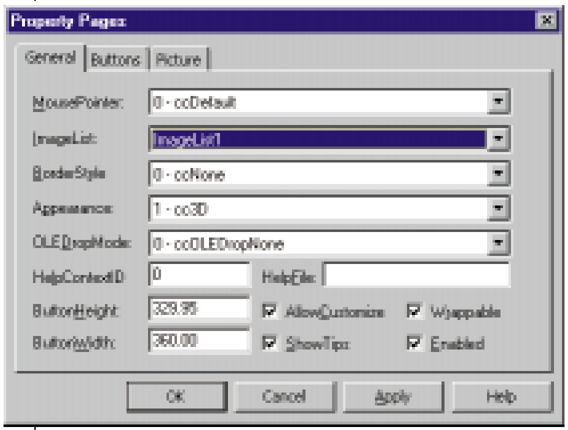
Current Image — Index: 4 Key: Copy Tag:

Images:

Insert Picture... | Remove Picture | Image Count: 4

OK | Cancel | Apply | Help

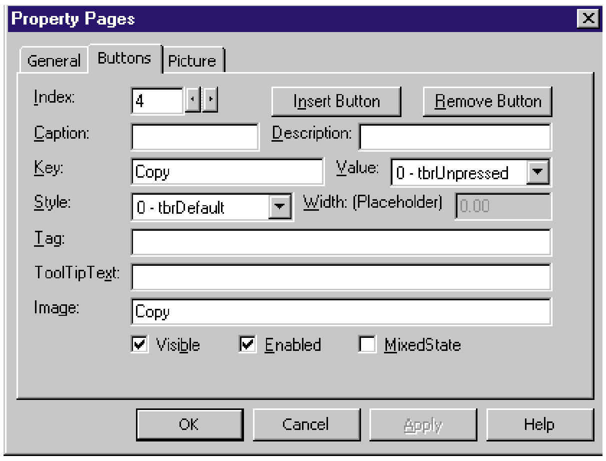
Fig 3 Use the Properties dialog to add images at design time


Property Pages — General | Buttons | Picture

MousePointer: 0 - ccDefault
ImageList: ImageList1
BorderStyle: 0 - ccNone
Appearance: 1 - cc3D
OLEDropMode: 0 - ccOLEDropNone
HelpContextID: 0 HelpFile:
ButtonHeight: 329.95 ☑ AllowCustomize ☑ Wrappable
ButtonWidth: 360.00 ☑ ShowTips ☑ Enabled

OK | Cancel | Apply | Help

Fig 4 Set the Toolbar to point to the ImageList


Property Pages — General | Buttons | Picture

Index: 4 Insert Button | Remove Button
Caption: Description:
Key: Copy Value: 0 - tbrUnpressed
Style: 0 - tbrDefault Width: (Placeholder) 0.00
Tag:
ToolTipText:
Image: Copy
☑ Visible ☑ Enabled ☐ MixedState

OK | Cancel | Apply | Help

Fig 5 Click Insert Button to add buttons to the toolbar

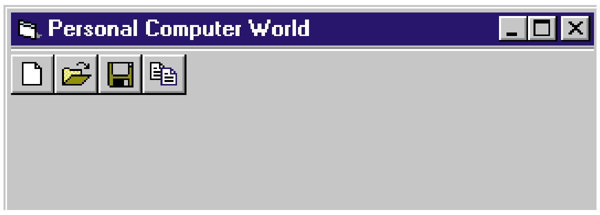


Fig 6 Running the toolbar application

up phone numbers, and to run it successfully in Netscape and Internet Explorer on both Windows and the Mac.

There are some problems, though. One is that only a subset of the CodeBase functions available in other languages are supported in Java. In particular, the query and relation functions are missing, so database searches are more or less limited to seek(). Another is that the product is rough and ready in places and you need to be prepared to experiment.

The Java classes are in a package called CodeBase, which should be renamed com.sequiter.codebase to follow Sun's guidelines for avoiding the risk of duplicate package names. Source is supplied, though, so you can go ahead and make the change. Most people will want to stay with the mainstream Java standards, using JDBC and SQL databases, but CodeBase is worth considering, especially if you need to integrate with an existing xBase system or want a lightweight solution on client and server.

All you need to jive with Java

Sun has released the Java Runtime Environment (JRE) version 1.1.3. It is a 2.5Mb self-extracting executable and you'll find a free download at www.java.sun.com. As its name implies, this is all you need to run Java applications without the parts of the JDK, which are only required for development. The idea is that you can just install the JRE, plus your own code, to get an application up and running on a target system.

Taken together with the latest wave of development tools, like Borland's excellent JBuilder or IBM's VisualAge, Java is becoming a realistic alternative to the likes of Visual Basic or Delphi for building and deploying applications.

I would be interested to hear from anyone who has successfully made that switch, with real applications as opposed to window dressing for web sites.

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**.

CodeBase 6.3 costs around £250 (ex VAT) from Highlander Software 0181 316 5001 www.highlander.co.uk; www.sequiter.com
Visual JavaScript is on the Netscape web site at www.netscape.com

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

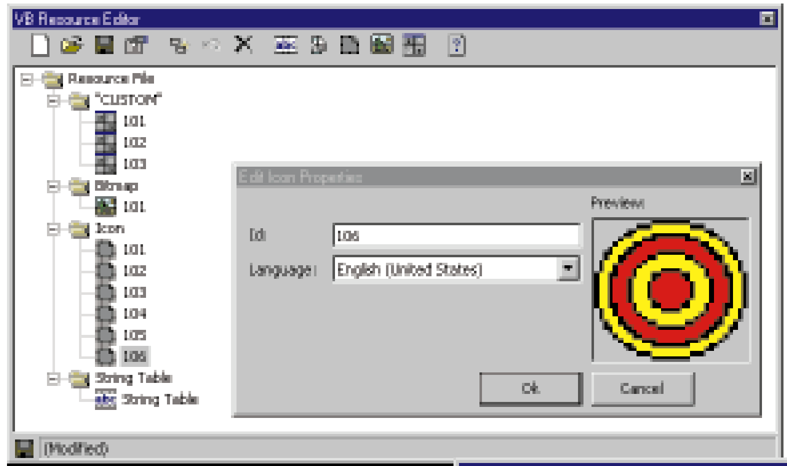
The meat of this package is in the controls themselves which are relatively lightweight, with some interesting features for the serious VB developer. An example is the list box constituent (Fig 2) that can be set to the owner-draw style, allowing you to create virtually any kind of fancy list box.

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

p309 ➢



VB Resource Editor

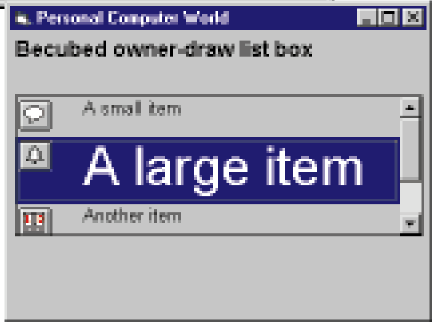
Resource File
- "CUSTOM"
  - 101
  - 102
  - 103
- Bitmap
  - 101
- Icon
  - 101
  - 102
  - 103
  - 104
  - 105
  - 106
- String Table
  - String Table

Edit Icon Properties

Id: 106

Language: English (United States)

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



Personal Computer World

Becubed owner-draw list box

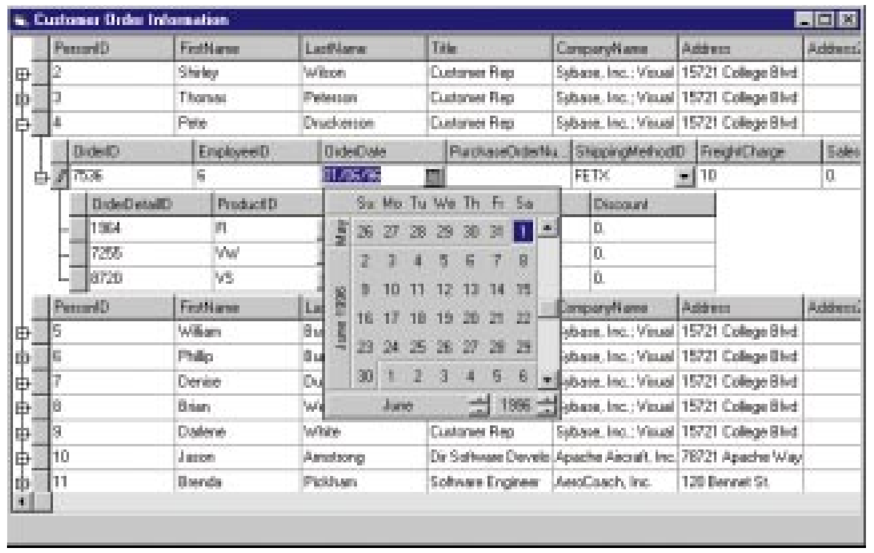
A small item

A large item

Another item

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



Customer Order Information

PersonID	FirstName	LastName	Title	CompanyName	Address	Address2
2	Shirley	Wilson	Customer Rep	Sybase, Inc.: Visual	15721 College Blvd	
3	Thomas	Peterson	Customer Rep	Sybase, Inc.: Visual	15721 College Blvd	
4	Pete	Druckerson	Customer Rep	Sybase, Inc.: Visual	15721 College Blvd	

OrderID	EmployeeID	OrderDate	PurchaseOrderNu	ShippingMethodID	FreightCharge	Sales
7536	6	01/06/96		FETX	10	0

OrderDetailID	ProductID	Discount
1964	PI	0.
7255	VW	0.
8720	VS	0.

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

June 1996

PersonID	FirstName	La…	CompanyName	Address	Address2
5	William	Bu	ybase, Inc.: Visual	15721 College Blvd	
6	Philip	Bu	ybase, Inc.: Visual	15721 College Blvd	
7	Denise	Du	ybase, Inc.: Visual	15721 College Blvd	
8	Brian	W	ybase, Inc.: Visual	15721 College Blvd	

PersonID	FirstName	LastName	Title	CompanyName	Address	Address2
9	Darlene	White	Customer Rep	Sybase, Inc.: Visual	15721 College Blvd	
10	Jason	Armstrong	Dir Software Develo	Apache Aircraft, Inc.	78721 Apache Way	
11	Brenda	Pickham	Software Engineer	AeroCoach, Inc.	120 Bennet St	

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

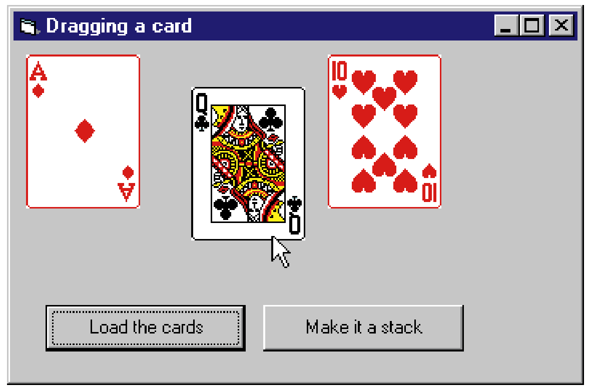
I suppose you could write extra code to drag a stack of cards. This is not the way to do it, though. If one card lands on another and you want to treat the two cards as a stack, it would be better to delete the lower image and draw a new image illustrating the top card with some artistic shading added, 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)



Dragging a card

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

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).

Q I 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?

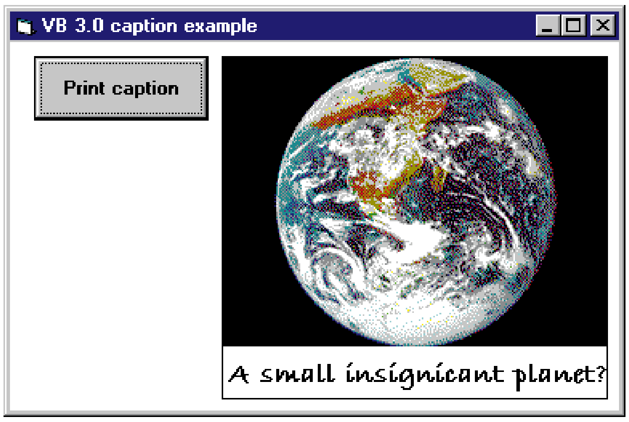
Edward Lea

A With 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.

p310 ➢

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 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.

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.

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

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

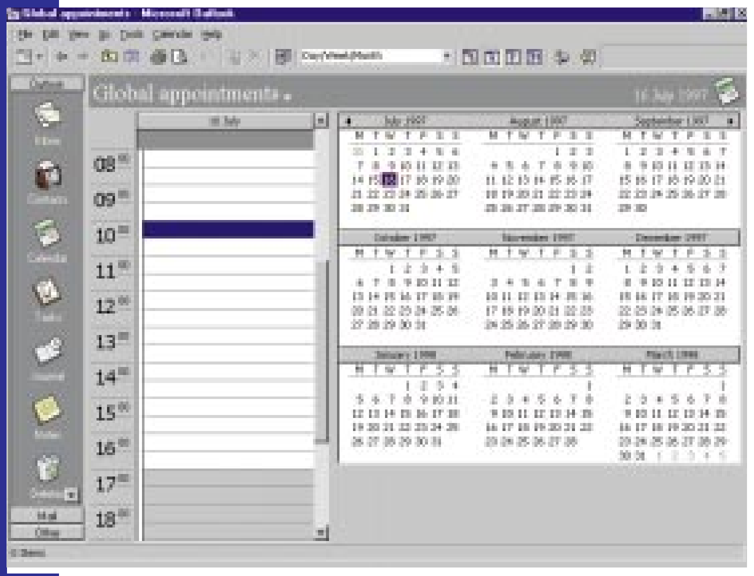
Standard issue

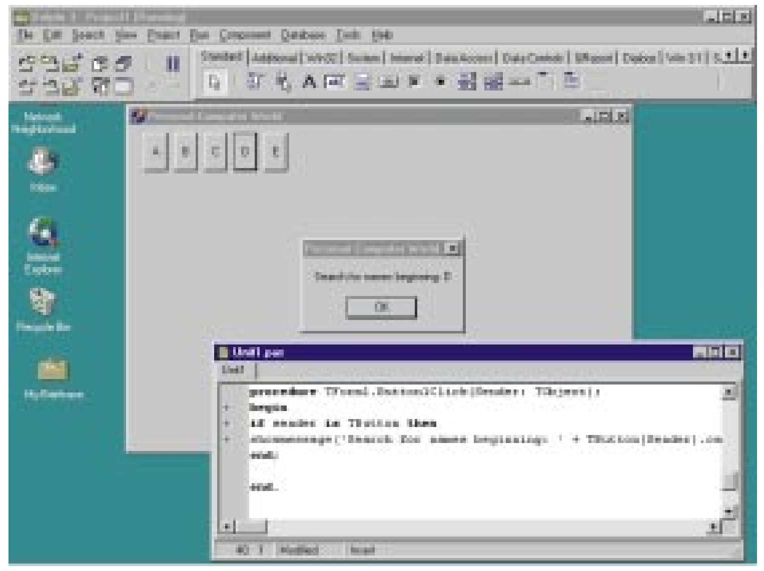
The problem is not confined to RMI. If you look at the Javasoft web pages (www.javasoft.com) you will see a host of APIs, standards and proposed standards for Java, including the Java Native Interface, JAR compression, multimedia APIs and many more. But the only way to find out what is and is not actually 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 ➢





```
procedure TForm1.Button1Click(Sender: TObject);
begin
if sender is TButton then
showmessage('Search for names beginning: ' + TButton(Sender).ca
end;

end.
```

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 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 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 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;
mybutton.OnClick := MyButtonClick;
end;
```

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

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

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

Fig 2 The MyButtonClick procedure

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

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 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.

Fig 3 Code for the OnClick event

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

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*

p296 ➢

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.

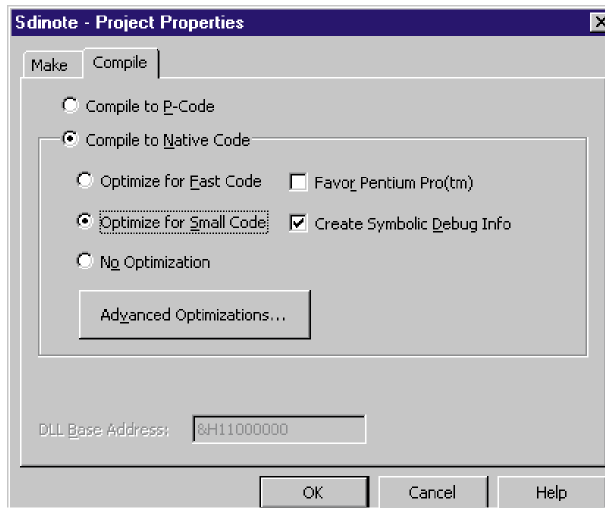
Here's how to do it. The first step is to compile your application to native code from Visual Basic. In the project properties, make sure that the options for Compile to native code and Create Symbolic Debug Info are both checked.

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 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.

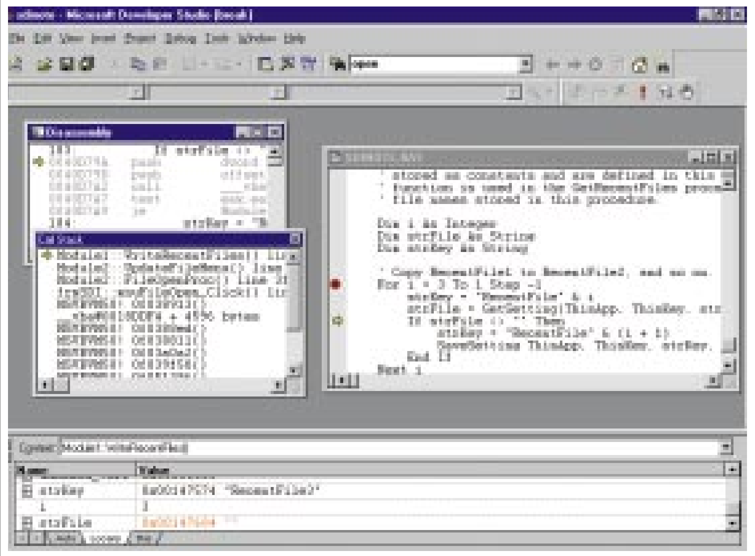


Sdinote - Project Properties

Make | Compile

( ) Compile to P-Code
(•) Compile to Native Code
( ) Optimize for Fast Code
[ ] Favor Pentium Pro(tm)
(•) Optimize for Small Code
[✓] Create Symbolic Debug Info
( ) No Optimization

DLL Base Address: &H11000000



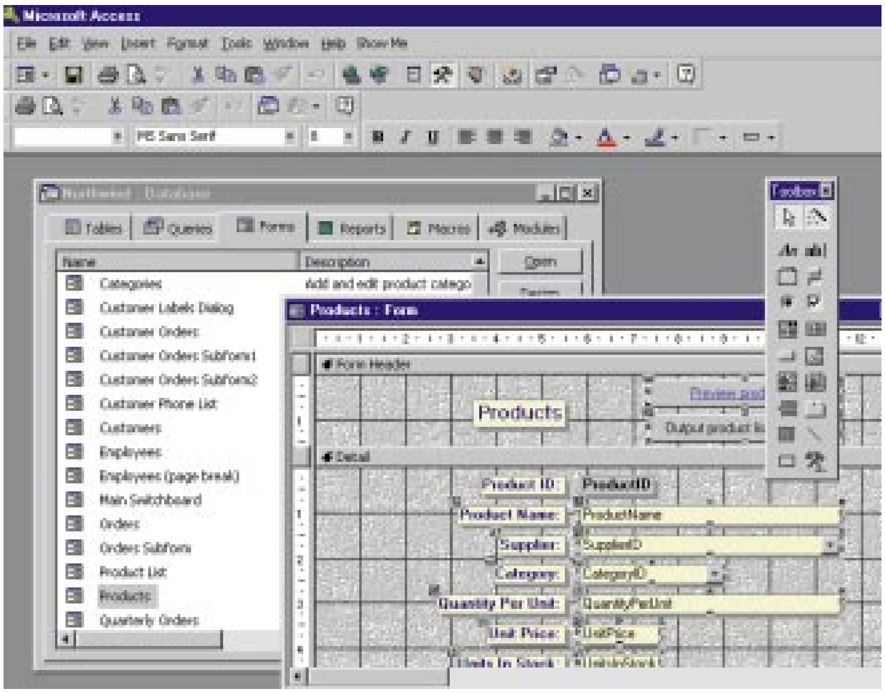
```
' stored as constants and are defined in this
' function is used in the GetRecentFiles proce
' file names stored in this procedure.

Dim i As Integer
Dim strFile As String
Dim strKey As String

' Copy RecentFile1 to RecentFile2, and so on.
For i = 3 To 1 Step -1
    strKey = "RecentFile" & i
    strFile = GetSetting(ThisApp, ThisKey, str
    If strFile <> "" Then
        strKey = "RecentFile" & (i + 1)
        SaveSetting ThisApp, ThisKey, strKey,
    End If
Next i
```

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

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 thunking 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.

Thunking 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.

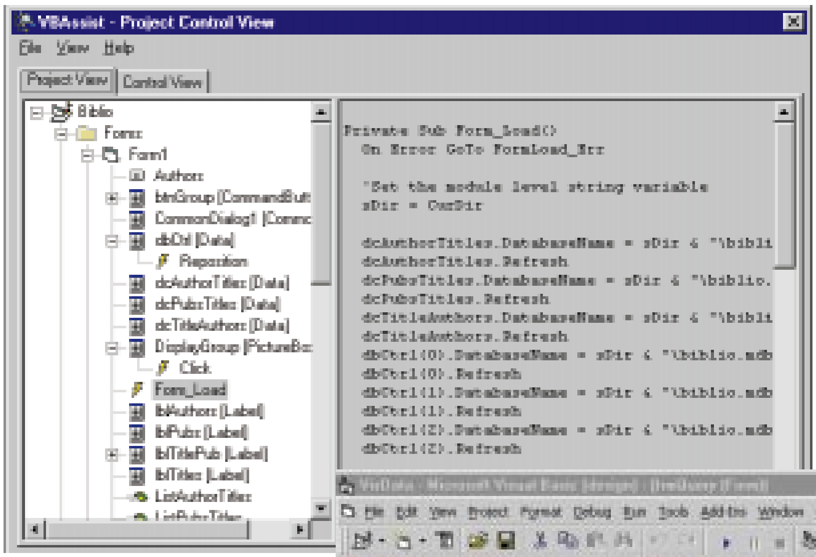
There are some excellent ideas. For example, the Project Control Assistant (Fig 1) analyses a VB project into an Explorer-style view so you can quickly see what event-driven code you have written. Accelerator Assistant (Fig 2) deals with the awkward business of assigning unique accelerator keys to menus 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 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 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.

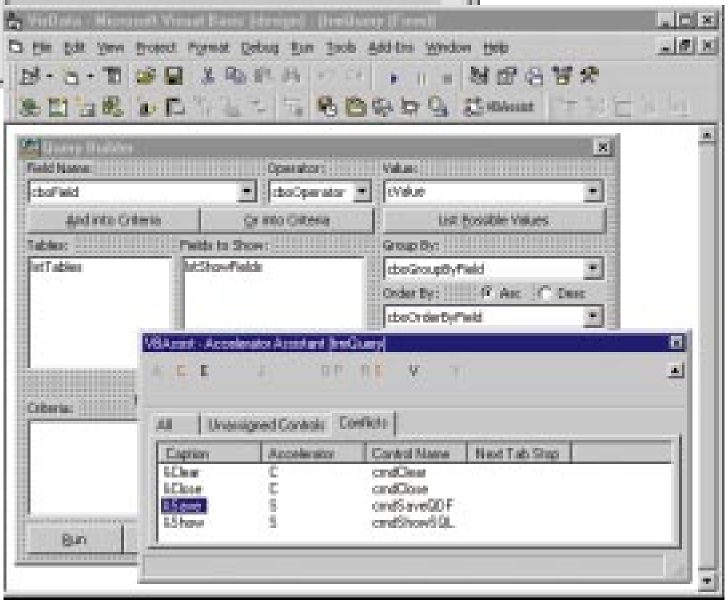
The useful tab-order Assistant exposes the current tab order visually and lets you

p264 ➢



```
Private Sub Form_Load()
  On Error GoTo FormLoad_Err

  'Set the module level string variable
  sDir = CurDir

  dcAuthorTitles.DatabaseName = sDir & "\bibli
  dcAuthorTitles.Refresh
  dcPubsTitles.DatabaseName = sDir & "\biblio.
  dcPubsTitles.Refresh
  dcTitleAuthors.DatabaseName = sDir & "\bibli
  dcTitleAuthors.Refresh
  dbCtrl(0).DatabaseName = sDir & "\biblio.mdb
  dbCtrl(0).Refresh
  dbCtrl(1).DatabaseName = sDir & "\biblio.mdb
  dbCtrl(1).Refresh
  dbCtrl(2).DatabaseName = sDir & "\biblio.mdb
  dbCtrl(2).Refresh
```

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



Caption	Accelerator	Control Name	Next Tab Stop
&Clear	C	cmdClear	
&Close	C	cmdClose	
&Save	S	cmdSaveQDF	
&Show	S	cmdShowSQL	

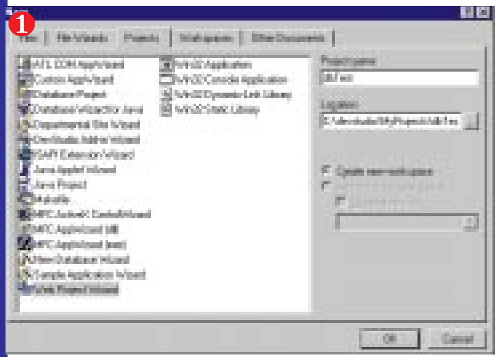
Fig 2 (below) Assist's Accelerator Assistant detects two sets of duplicate keyboard shortcuts on this Visual Basic form. Note the VB Assist toolbar below the standard one change it by clicking with the mouse. The common dialog wizard automatically writes code for using the common dialog control, while 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.

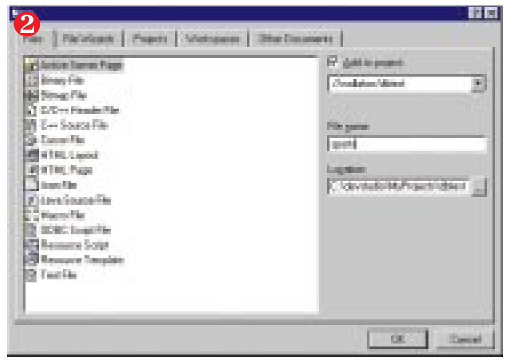
As an example, what follows is how to publish an Access MDB with Visual InterDev. For this to work, you need to be using a Microsoft web server: either Internet Information Server running on NT, or the Personal Web Server for Windows 95. The example given creates a read-only


1

telephone directory but at least it is dynamically linked to a database, so it is easy to maintain and update. Once you grasp the concepts, you can add logic to provide read-write access to the data. For use on an intranet of significant size it would be better to use NT for the web server; and for the database, a client-server database such as SQL Server.

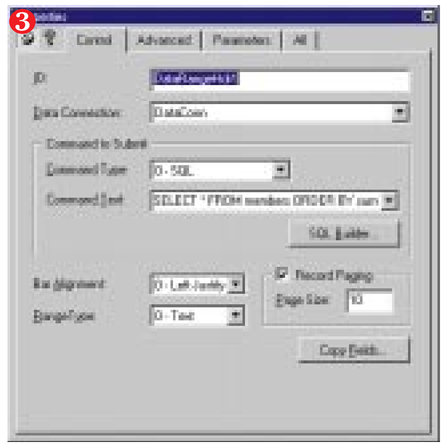
The exciting aspect is that by publishing the database through an intranet, you avoid the need to distribute any client applications. Any connected computer can simply point a web browser at the page in question, and that includes non-Windows machines.

1. The first step is to configure an ODBC data source. This is because Visual InterDev uses ActiveX Data Objects, which use


2

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

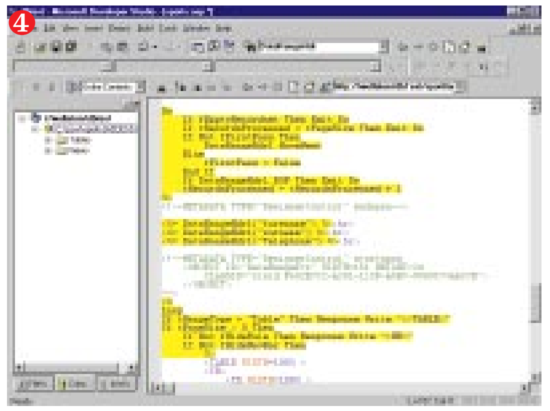

3

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.

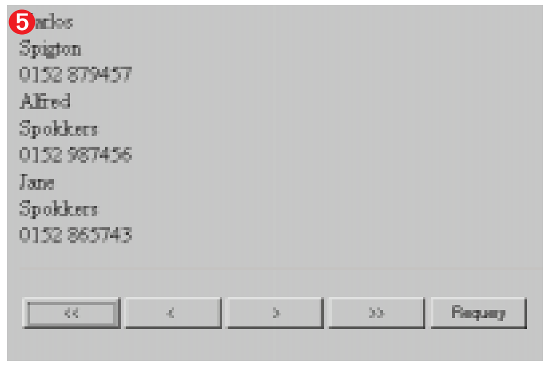
4. Next, choose File – New and create a new Active Server Page *(screenshot 2)*. This is an HTML page capable of running server-side scripts on a Microsoft web server. Visual InterDev opens the new page in the HTML editor. This example is called SPORTS.ASP.

5. Move the edit cursor to just after the


4

<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


5

Carlos
Spigton
0152 879457
Alfred
Spokkers
0152 987456
Jane
Spokkers
0152 865743

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 (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.

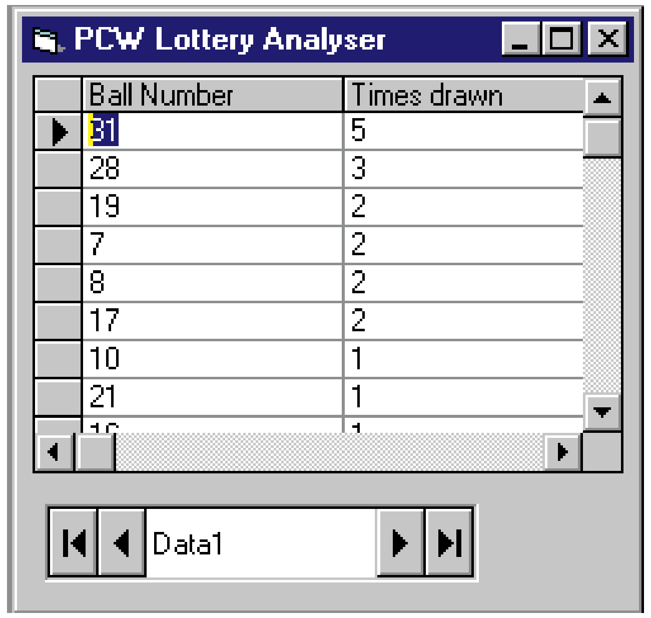
As a further check, the WISE installer has a clever feature whereby it runs in the background at runtime, capturing details of all the files used. And finally, a good tip is

p267 ➢

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.

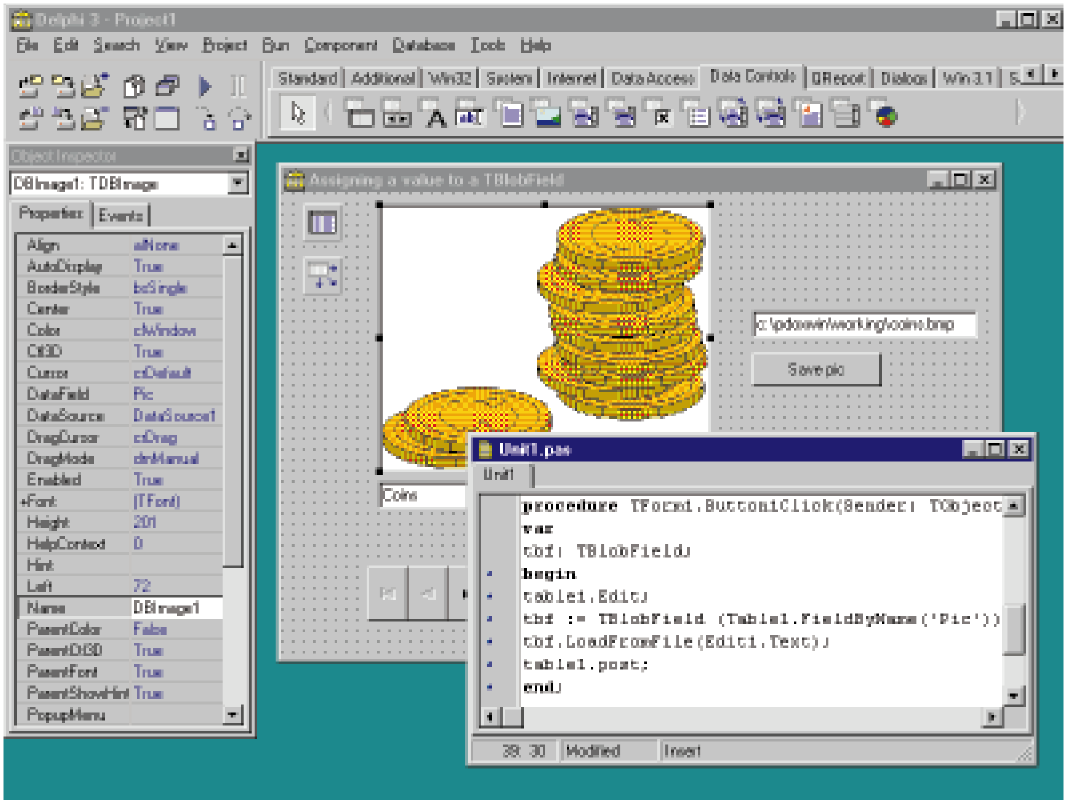

PCW Lottery Analyser

Ball Number	Times drawn
31	5
28	3
19	2
7	2
8	2
17	2
10	1
21	1

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**

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;
```


Delphi 3 - Project1
File Edit Search View Project Run Component Database Tools Help
Standard Additional Win32 System Internet Data Access Data Controls QReport Dialogs Win 3.1
Object Inspector
DBImage1: TDBImage
Properties Events
Assigning a value to a TBlobField
c:\pdoxwin\working\coins.bmp
Save pic
Coins
Unit1.pas
Unit
39: 30 Modified Insert

Align	alNone
AutoDisplay	True
BorderStyle	bsSingle
Center	True
Color	clWindow
Ctl3D	True
Cursor	crDefault
DataField	Pic
DataSource	DataSource1
DragCursor	crDrag
DragMode	dmManual
Enabled	True
+Font	(TFont)
Height	201
HelpContext	0
Hint	
Left	72
Name	DBImage1
ParentColor	False
ParentCtl3D	True
ParentFont	True
ParentShowHint	True
PopupMenu	

```
procedure TForm1.Button1Click(Sender: TObject
var
tbf: TBlobField;
begin
table1.Edit;
tbf := TBlobField (Table1.FieldByName('Pic'))
tbf.LoadFromFile(Edit1.Text);
table1.post;
end;
```

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(Edit
1.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:

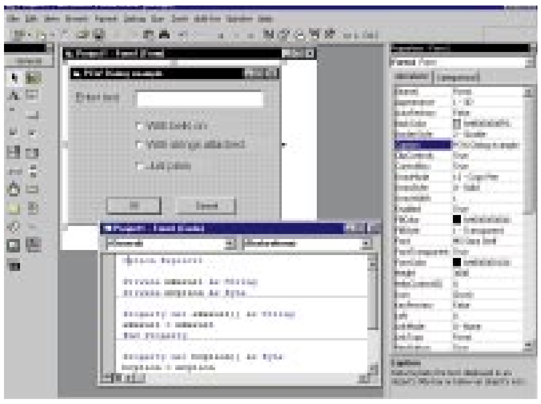
- Check "When a program starts — Save Changes" in the Options-Environment dialog.
- Do a fresh installation of Windows and VB, preferably onto a newly-formatted hard disk. This brute force approach corrects two common problems: mismatched system DLLs and/or a corrupt registry.
- Double-check API calls 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.

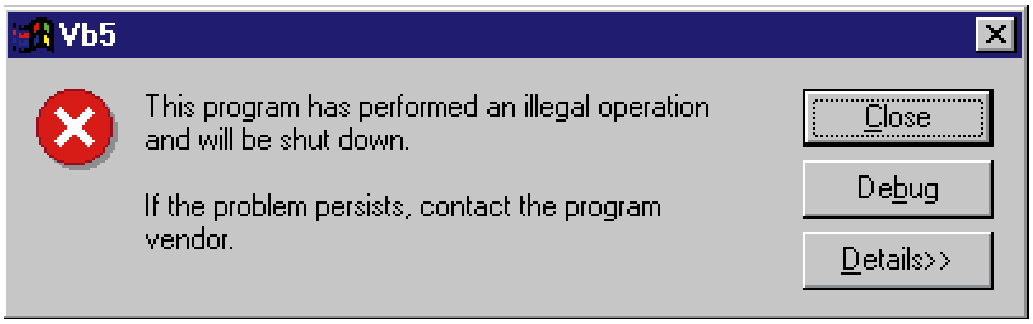
Dealing with dialogs

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,





Vb5

This program has performed an illegal operation and will be shut down.

If the problem persists, contact the program vendor.

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

p280 ➢

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.

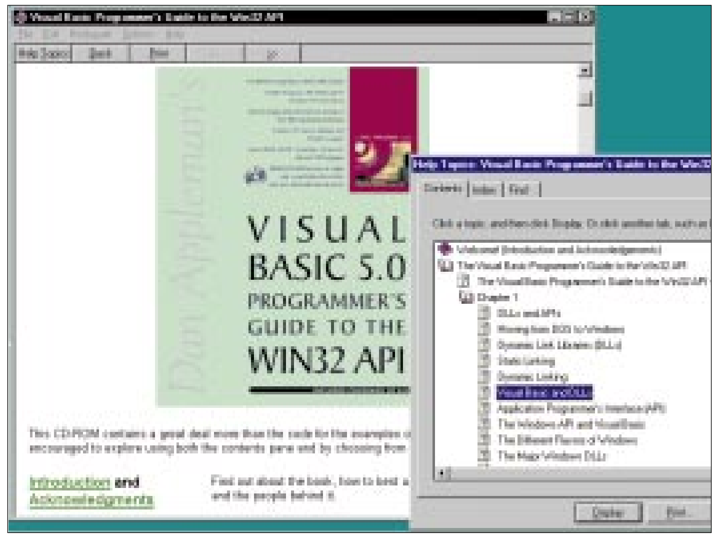
It is a basic introduction, useful for anyone wanting to get up to speed on what JavaBeans is all about.

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.


VISUAL BASIC 5.0 PROGRAMMER'S GUIDE TO THE WIN32 API
Introduction and Acknowledgments

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

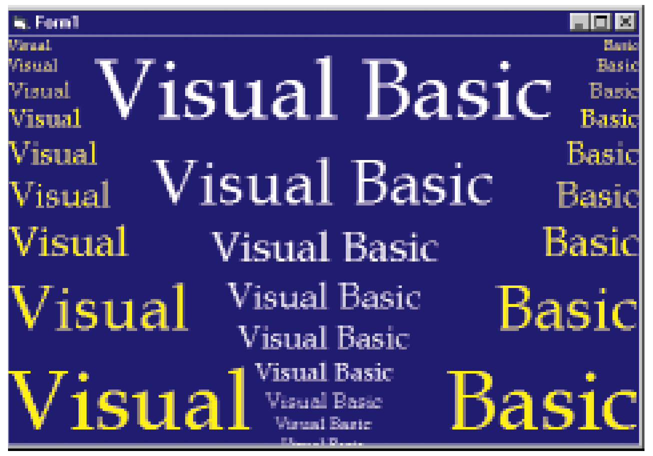
declare private variables for the dialog's values. Next, create property procedures to access the variables. In your dialog's OK procedure, write the values from the dialog controls back to the private variables. Finally, you can create a ShowDialog method which takes parameters to initialise the dialog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



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

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 tool like Visual C++ has a complex interface and a product-specific book is helpful. Fourth, most C++ programmers work 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).

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)
```

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.

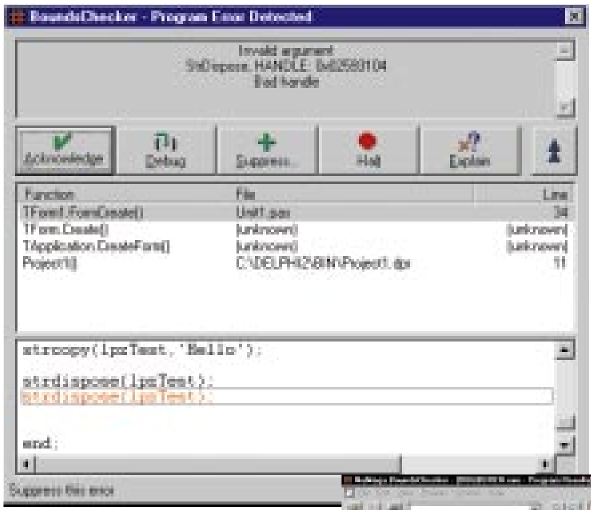
The natural question to ask is: what can BoundsChecker do that Delphi's own debugger cannot? One of the issues is detecting memory leaks. For example, your program might include the following code:

```
var
nullstring: pchar;
begin
nullstring := strAlloc(50);
strcopy(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

p283 ➢



Invalid argument
StrDispose: HANDLE: 0x02580104
Bad handle

Function	File	Line
TForm1.FormCreate()	Unit1.pas	34
TForm.Create()	(unknown)	(unknown)
TApplication.CreateForm()	(unknown)	(unknown)
Project1()	C:\DELPHI2\BIN\Project1.dpr	11

```
strcopy(lpzText,'Hello');

strdispose(lpzText);
strdispose(lpzText);


end;
```

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.

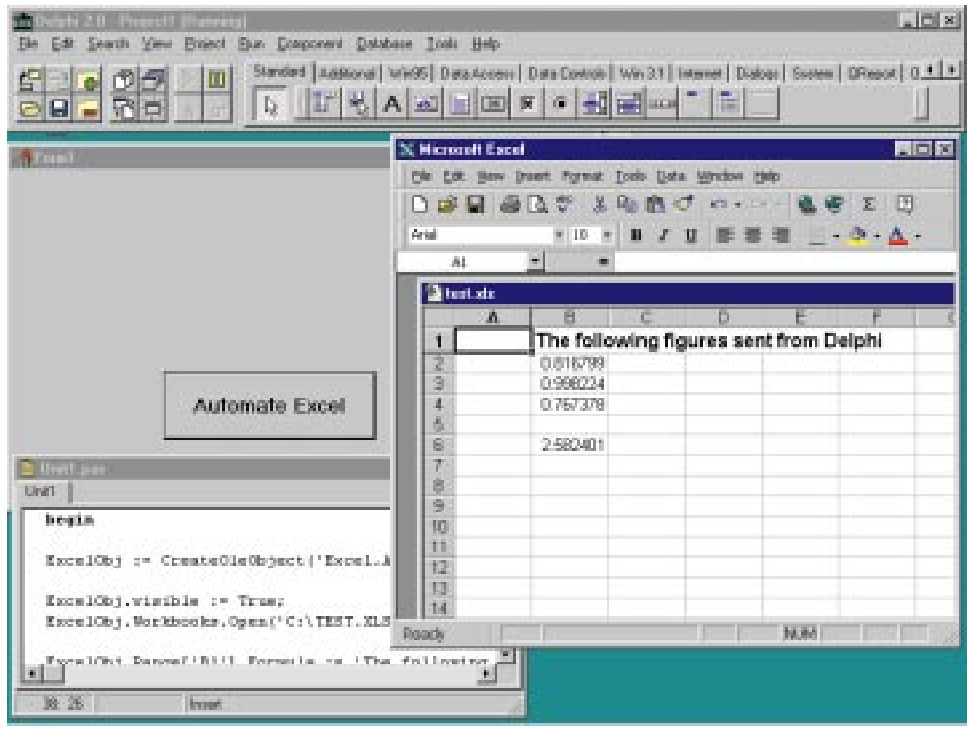
The following books are available from Computer Manuals 0121 706 6000:
- *Visual Basic 5.0 Programmer's Guide to the Win32 API* by Dan Appleman, Ziff-Davis Press. Book and CD £54.95 (inc VAT). Further information at www.desaware.com.
- *Using Delphi 3.0* by Todd Miller, David Powell and others, Que. 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.



```
begin

ExcelObj := CreateOleObject('Excel.A

ExcelObj.visible := True;
ExcelObj.Workbooks.Open('C:\TEST.XLS

ExcelObj.Range['B1'].Formula := 'The following
```

	A	B	C	D	E	F
1		The following figures sent from Delphi				
2		0.816799				
3		0.998224				
4		0.767378				
5						
6		2.582401				

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

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."*

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, 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 RICHTX32.OCX component which makes it work, complete with an updated help file. This later version is required in order to work with pictures.

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');
...
```

Programming Outlook 97

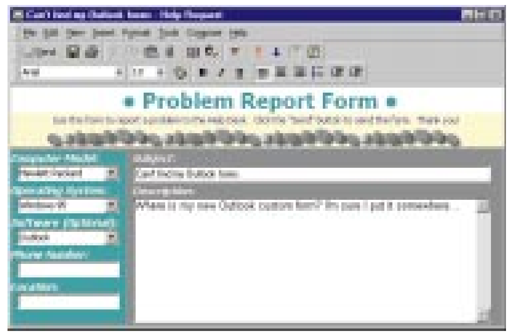
Outlook is an excellent starting point for an Office 97 application. Most people need an email reader and an address book or contact manager, and Outlook does both. The natural next step is to add functionality. For example, Outlook's contact menu already has an option to start a new letter to the current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

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

p299 ➢

Book Reviews

■ **Building Applications with Outlook 97**
£37.49 (book and CD)
Microsoft Press



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

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 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.*

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 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

p300 ➢

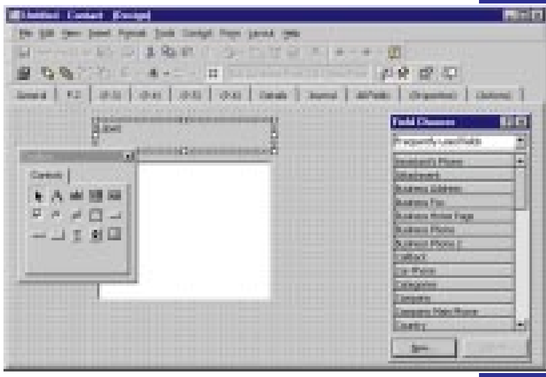


Fig 2 Create a new contact and then choose Design Outlook Form to open the form designer

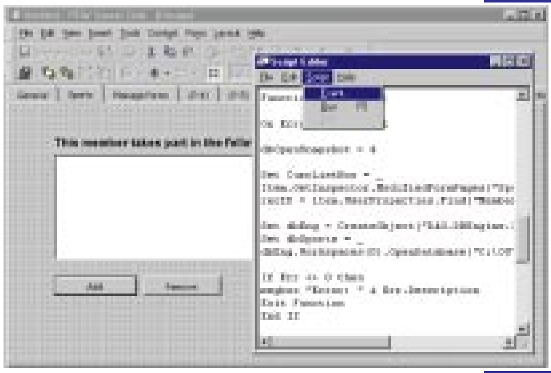


Fig 3 Choose View Code to open the Script Editor. The cunningly-titled Run option does not run the code but merely checks the syntax

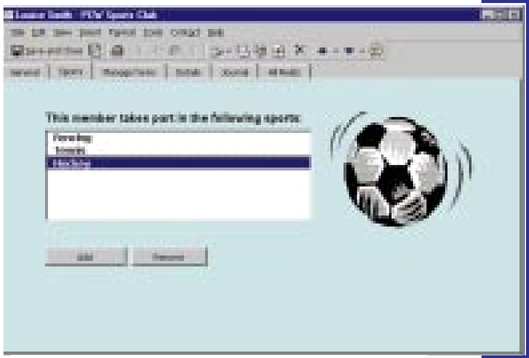
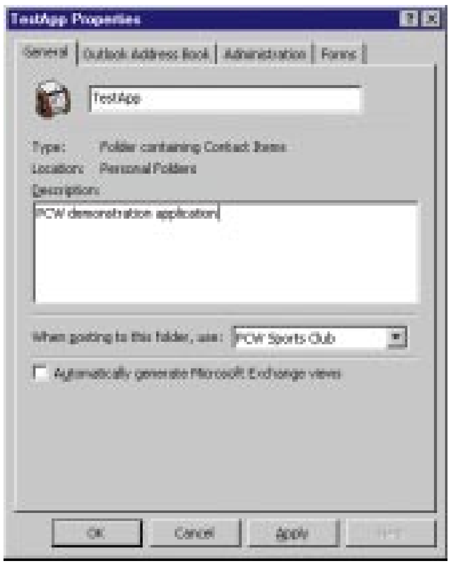


Fig 4 When the form opens, Outlook looks up the list of sports from an external database

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.


TestApp Properties

General | Outlook Address Book | Administration | Forms

TestApp

Type: Folder containing Contact Items

Location: Personal Folders

Description:

PCW demonstration application

When posting to this folder, use: PCW Sports Club

[ ] Automatically generate Microsoft Exchange views

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

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
```

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):

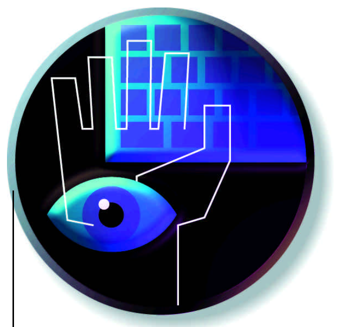
1. Open the contacts folder and choose New Contact from the Contact menu. This opens the built-in Contacts form.
2. From the Tools menu choose Design Outlook Form. This opens the form in design mode with six spare tabs available. Click the second tab, rename it Sports, and add a list box control. Name 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**.

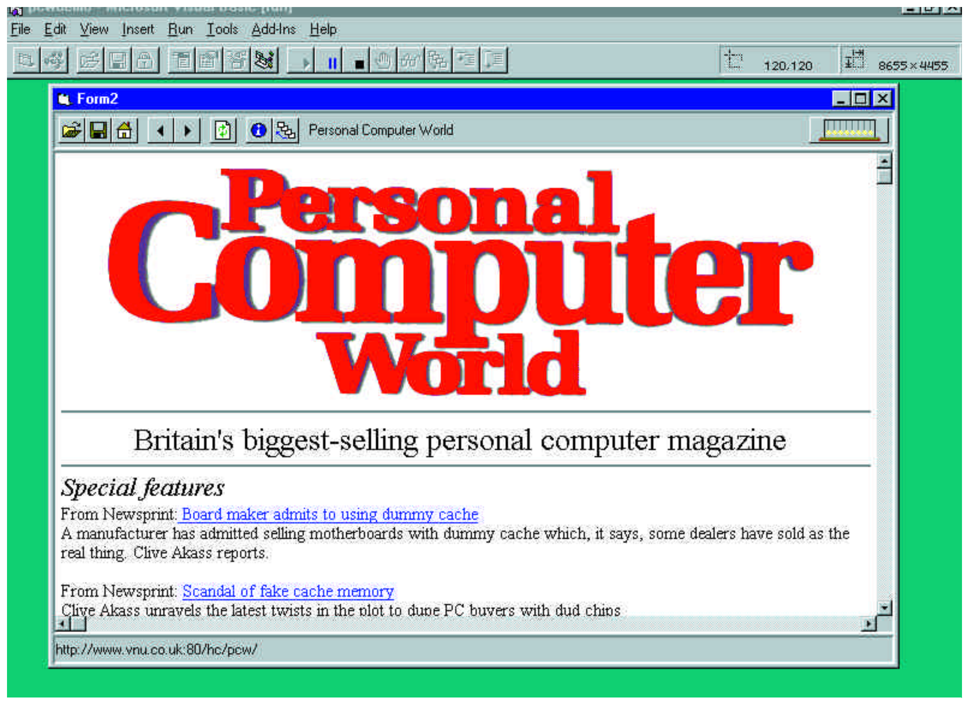


Sax appeal

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 Intenet 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 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.



Personal Computer World

Britain's biggest-selling personal computer magazine

Special features

From Newsprint: Board maker admits to using dummy cache
A manufacturer has admitted selling motherboards with dummy cache which, it says, some dealers have sold as the real thing. Clive Akass reports.

From Newsprint: Scandal of fake cache memory
Clive Akass unravels the latest twists in the plot to dupe PC buyers with dud chips

http://www.vnu.co.uk:80/hc/pcw/

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

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 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 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.

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 project with a 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)
```

Widgets for your data

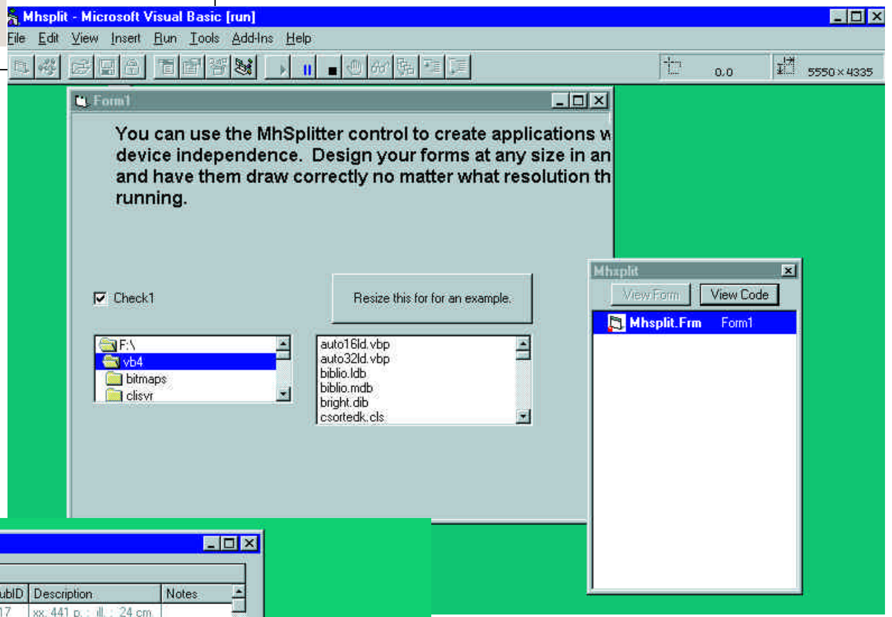
Sheridan's Data Widgets has long been one of the most popular Visual Basic add-ons, particularly since the VB 3.0 grid is so poor. The data-bound controls in VB 4.0 are better, but still leave room for third-party enhancements. Version 2.0 brings the expected conversion to 16- and 32-bit OCX format, 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:

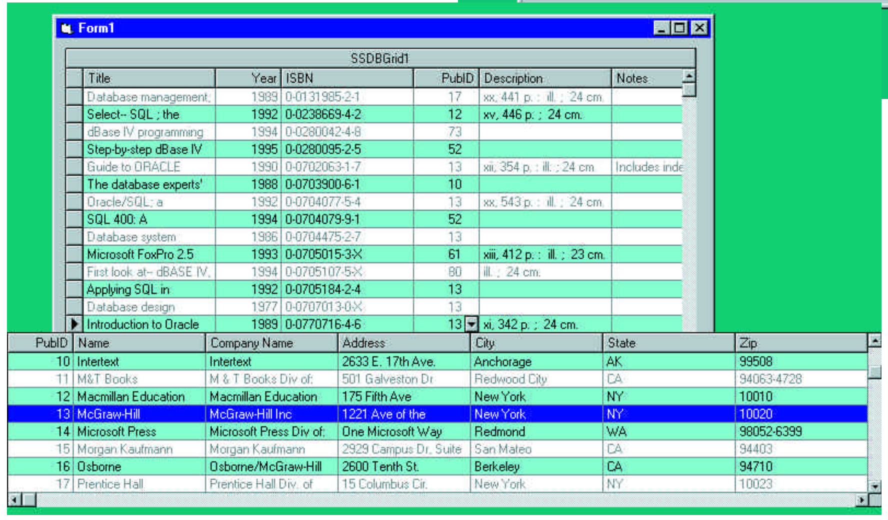
```
SSDbGrid1.Columns(2).Style = 1
' edit button.
```

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).



You can use the MhSplitter control to create applications w device independence. Design your forms at any size in an and have them draw correctly no matter what resolution th running.

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



Title	Year	ISBN	PubID	Description	Notes
Database management;	1989	0-0131985-2-1	17	xx, 441 p. : ill. ; 24 cm.	
Select-- SQL ; the	1992	0-0238669-4-2	12	xv, 446 p. ; 24 cm.	
dBase IV programming	1994	0-0280042-4-8	73		
Step-by-step dBase IV	1995	0-0280095-2-5	52		
Guide to ORACLE	1990	0-0702063-1-7	13	xii, 354 p. : ill. ; 24 cm.	Includes inde
The database experts'	1988	0-0703900-6-1	10		
Oracle/SQL; a	1992	0-0704077-5-4	13	xx, 543 p. : ill. ; 24 cm.	
SQL 400: A	1994	0-0704079-9-1	52		
Database system	1986	0-0704475-2-7	13		
Microsoft FoxPro 2.5	1993	0-0705015-3-X	61	xiii, 412 p. : ill. ; 23 cm.	
First look at-- dBASE IV,	1994	0-0705107-5-X	80	ill. ; 24 cm.	
Applying SQL in	1992	0-0705184-2-4	13		
Database design	1977	0-0707013-0-X	13		
Introduction to Oracle	1989	0-0770716-4-6	13	xi, 342 p. ; 24 cm.	

PubID	Name	Company Name	Address	City	State	Zip
10	Intertext	Intertext	2633 E. 17th Ave.	Anchorage	AK	99508
11	M&T Books	M & T Books Div of:	501 Galveston Dr	Redwood City	CA	94063-4728
12	Macmillan Education	Macmillan Education	175 Fifth Ave	New York	NY	10010
13	McGraw-Hill	McGraw-Hill Inc	1221 Ave of the	New York	NY	10020
14	Microsoft Press	Microsoft Press Div of:	One Microsoft Way	Redmond	WA	98052-6399
15	Morgan Kaufmann	Morgan Kaufmann	2929 Campus Dr. Suite	San Mateo	CA	94403
16	Osborne	Osborne/McGraw-Hill	2600 Tenth St.	Berkeley	CA	94710
17	Prentice Hall	Prentice Hall Div. of	15 Columbus Cir.	New York	NY	10023

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**

p285 ➢

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). MhRealInput 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 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.

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_GETSCREENS AVEACTIVE, 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_SETSCREENS AVEACTIVE, bActive,
ByVal pvParam, 0)

If lRetVal = False Then
MsgBox “Call to SystemParametersInfo failed”
End If

End Sub
```

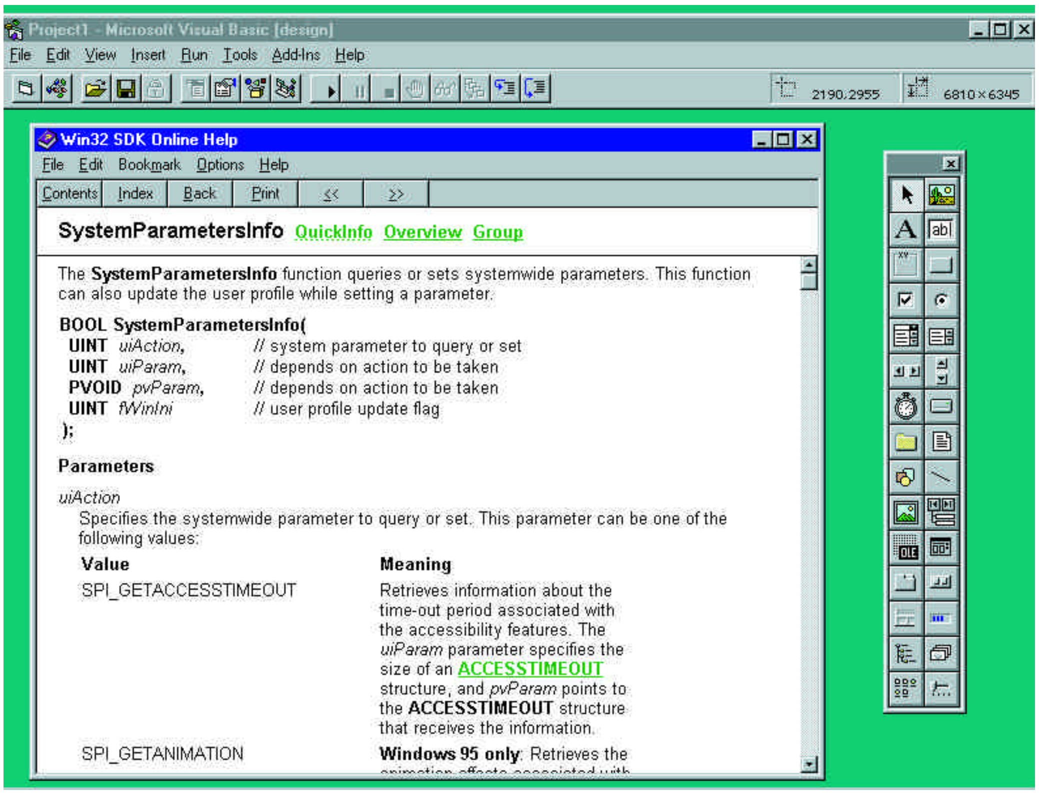
Hacking the system in Windows 95
Mark Horton writes: *"I've just bought a new system with Windows 95 and VB 4.0. My computer has a WIn/TV card, and I wanted to write a program that would turn the screensaver off and on without having to go into the display properties tab. How or where can I find out 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.

p286 ➢

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.


Project1 - Microsoft Visual Basic [design]
File Edit View Insert Run Tools Add-Ins Help
2190,2955
6810 x 6345
Win32 SDK Online Help
File Edit Bookmark Options Help
Contents Index Back Print << >>

SystemParametersInfo QuickInfo Overview Group

The **SystemParametersInfo** function queries or sets systemwide parameters. This function can also update the user profile while setting a parameter.

```
BOOL SystemParametersInfo(
  UINT  uiAction,    // system parameter to query or set
  UINT  uiParam,     // depends on action to be taken
  PVOID pvParam,     // depends on action to be taken
  UINT  fWinIni      // user profile update flag
);
```

Parameters

uiAction

Specifies the systemwide parameter to query or set. This parameter can be one of the following values:

Value	Meaning
SPI_GETACCESSTIMEOUT	Retrieves information about the time-out period associated with the accessibility features. The *uiParam* parameter specifies the size of an ACCESSTIMEOUT structure, and *pvParam* points to the **ACCESSTIMEOUT** structure that receives the information.
SPI_GETANIMATION	**Windows 95 only**: Retrieves the

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?

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 1A 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)

Listing 3: ParamCount

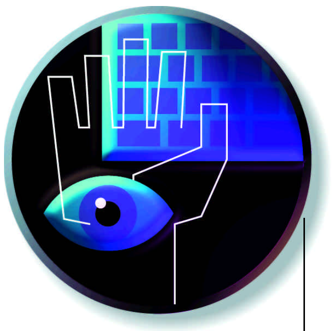
```
procedure TForm1.Button1Click(Sender: TObject);
var
i: integer;

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.

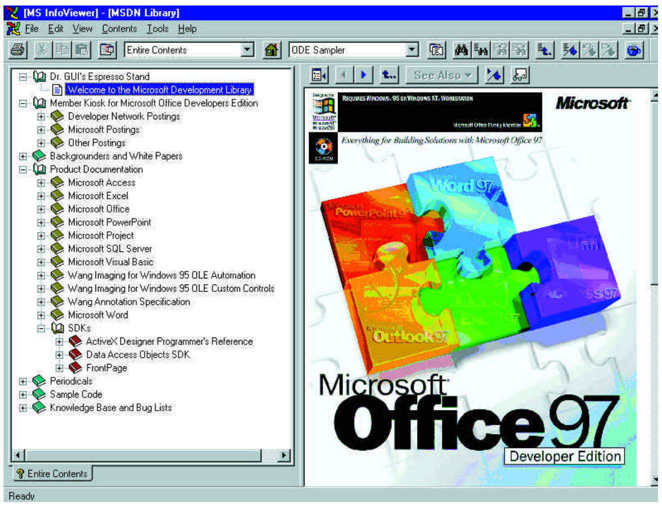
To do these wonderful things, you need appropriate resources. This might or might not include the Office 97 Developer Edition, Microsoft's one-stop solution. What you get is the Office Professional CD, plus a further CD of developer tools. There are also two books: the *Office 97 Visual Basic Programmer's 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 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.

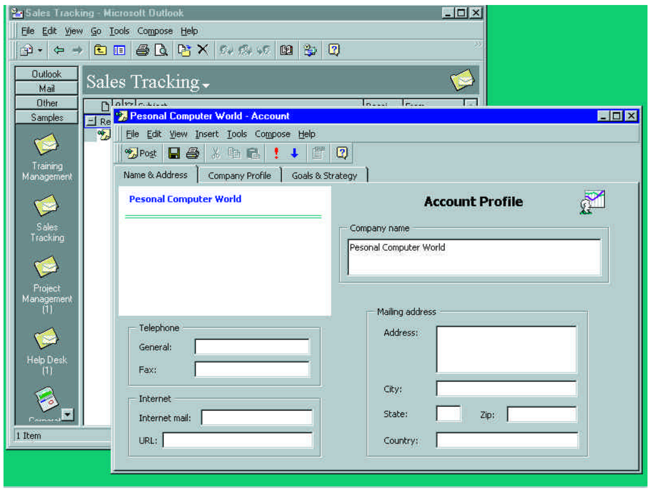
The most essential developer reference, the VBA reference for each application, is actually on the Office 97 Professional CD so it is questionable whether the Developer Edition is worth having. Most of the material can also be found on the Microsoft web site, often in updated form. There is only one 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

p302 ➢



Dr. GUI's Espresso Stand
Welcome to the Microsoft Development Library
Member Kiosk for Microsoft Office Developers Edition
Developer Network Postings
Microsoft Postings
Other Postings
Backgrounders and White Papers
Product Documentation
Microsoft Access
Microsoft Excel
Microsoft Office
Microsoft PowerPoint
Microsoft Project
Microsoft SQL Server
Microsoft Visual Basic
Wang Imaging for Windows 95 OLE Automation
Wang Imaging for Windows 95 OLE Custom Controls
Wang Annotation Specification
Microsoft Word
SDKs
ActiveX Designer Programmer's Reference
Data Access Objects SDK
FrontPage
Periodicals
Sample Code
Knowledge Base and Bug Lists

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



Pesonal Computer World

Company name: Pesonal Computer World

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

Fig 1 Using the clipboard

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

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 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 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?”*

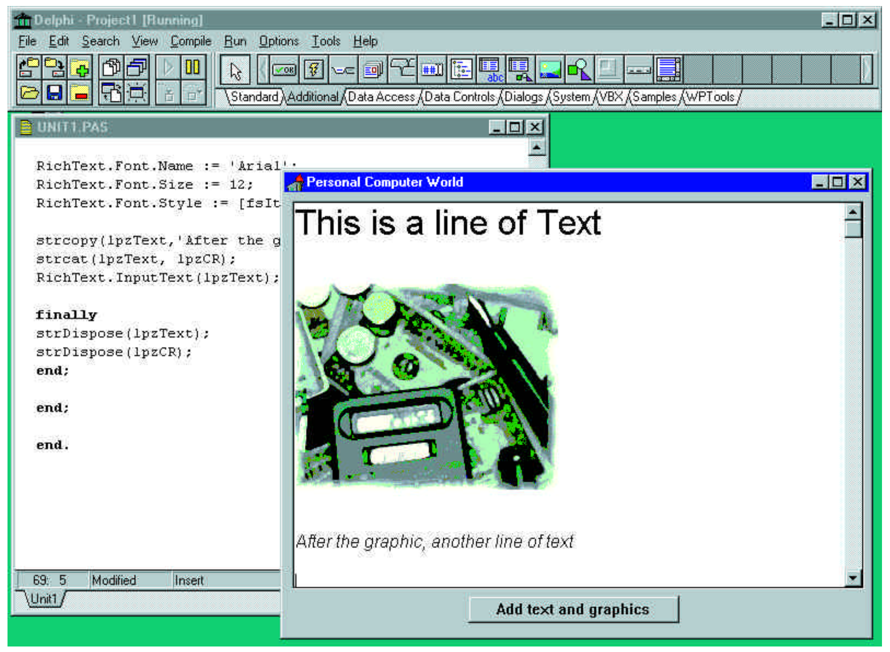
If you are working in Windows 95 or NT, the standard rich text control can display formatted text and graphics. It is easy to miss this functionality in Visual Basic, since there is no InsertPicture method. You can insert OLE objects, though, using the OLEObjects collection. For example, this code inserts 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

Fig 2 Using WPTools

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

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

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

RichText.clear;
RichText.Font.Name := ‘Arial’;
RichText.Font.Size := 24;

strcopy(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];

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

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

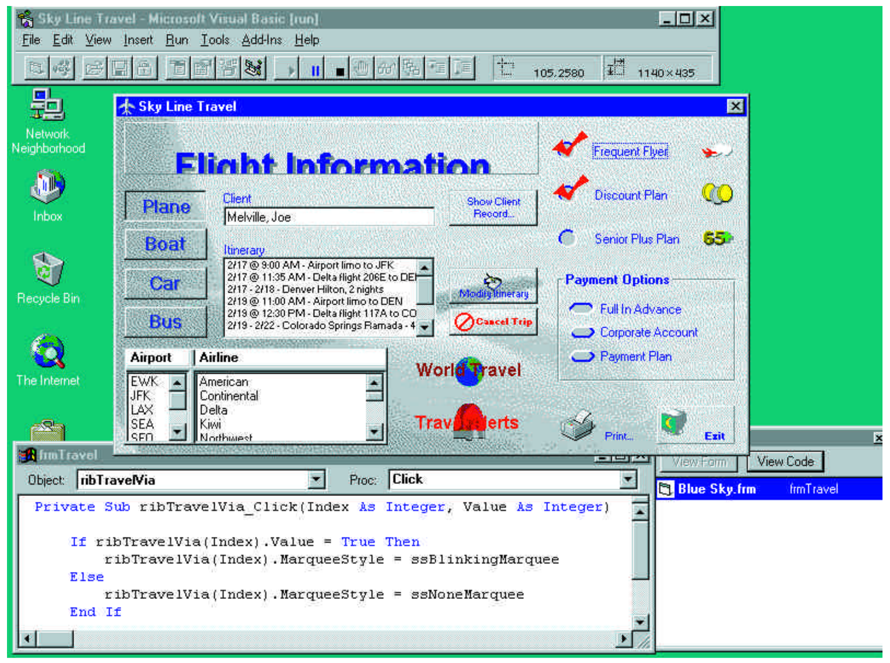

Delphi - Project1 [Running]
File Edit Search View Compile Run Options Tools Help
Standard Additional Data Access Data Controls Dialogs System VBX Samples WPTools
UNIT1.PAS
Personal Computer World
This is a line of Text
After the graphic, another line of text
Add text and graphics
69: 5 Modified Insert
Unit1

```
RichText.Font.Name := 'Arial';
RichText.Font.Size := 12;
RichText.Font.Style := [fsIt

strcopy(lpzText,'After the g
strcat(lpzText, lpzCR);
RichText.InputText(lpzText);

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

end;

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.



Client: Melville, Joe

Itinerary:
2/17 @ 9:00 AM - Airport limo to JFK
2/17 @ 11:35 AM - Delta flight 206E to DEN
2/17 - 2/18 - Denver Hilton, 2 nights
2/19 @ 11:00 AM - Airport limo to DEN
2/19 @ 12:30 PM - Delta flight 117A to CO
2/19 - 2/22 - Colorado Springs Ramada - 4

Airport	Airline
EWK	American
JFK	Continental
LAX	Delta
SEA	Kiwi
SFO	Northwest

Object: ribTravelVia Proc: Click

```
Private Sub ribTravelVia_Click(Index As Integer, Value As Integer)

    If ribTravelVia(Index).Value = True Then
        ribTravelVia(Index).MarqueeStyle = ssBlinkingMarquee
    Else
        ribTravelVia(Index).MarqueeStyle = ssNoneMarquee
    End If
```

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

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 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 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
```

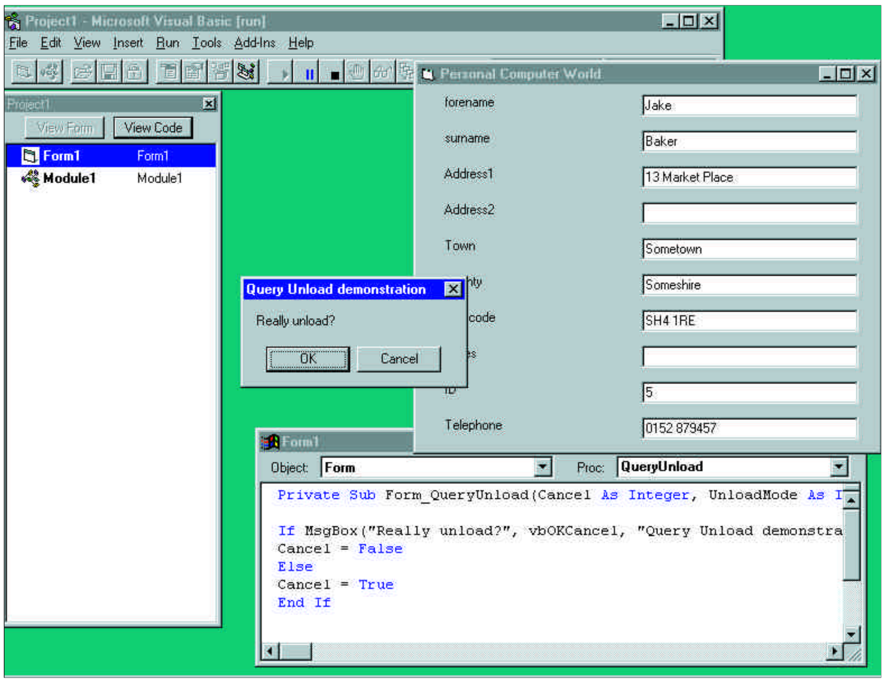
“What should the CurrPerson.Counter part look like?”

Andrew’s idea is to write a loop that 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.

p305 ➢



forename	Jake
surname	Baker
Address1	13 Market Place
Address2	
Town	Sometown
...nty	Someshire
...code	SH4 1RE
...s	
ID	5
Telephone	0152 879457

Really unload?

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As I

If MsgBox("Really unload?", vbOKCancel, "Query Unload demonstra
Cancel = False
Else
Cancel = True
End If
```

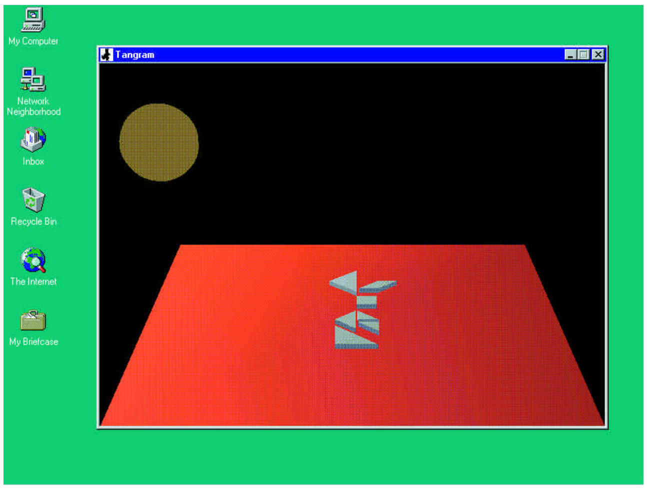


Fig 3 (above) Using the QueryUnload event to confirm a close decision

Left (see "Inside Com") This application from Inside COM demonstrates agregation, 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 Threed £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.

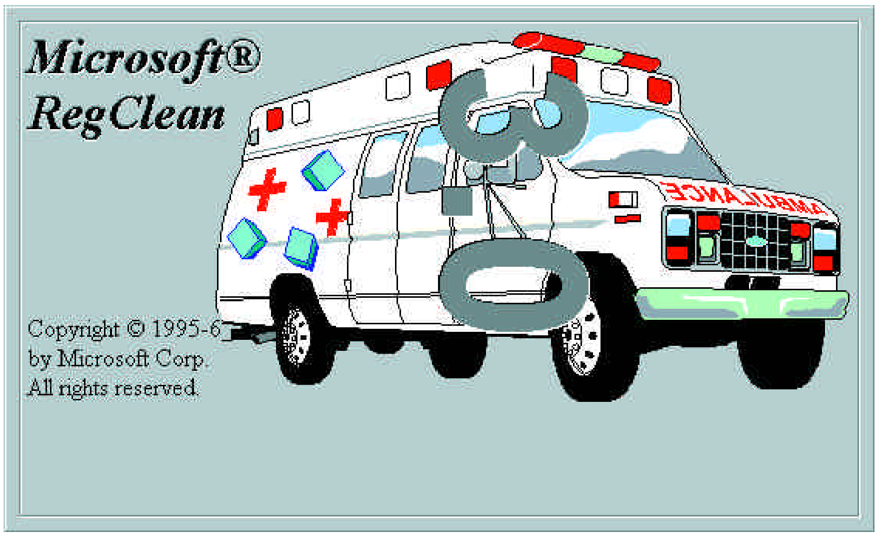
Sounds fun? This is exactly what can happen with Visual Basic 4.0. The reason, as you will have guessed, is that both VB itself and the many OCX controls which come with it depend on numerous registry settings. If the registry gets scrambled, this is the kind of thing that can happen.

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."

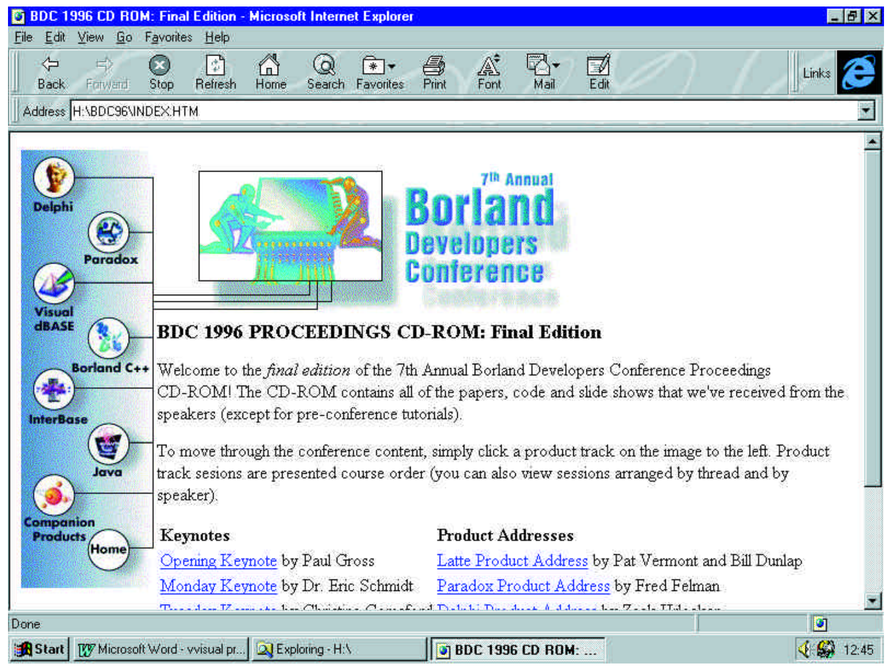
As for Regclean, a utility that comes with VB, I have come to mistrust it deeply. In a misguided moment I ran the latest version 3.0 which you can download from www.microsoft.com. The idea was to fix the annoying messages VB gives you when something is awry in the registry: "Object server not correctly 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. 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 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



Microsoft's RegClean 3.0: proceed at your own risk

p308 ➢



BDC 1996 CD ROM: Final Edition - Microsoft Internet Explorer

Address H:\BDC96\INDEX.HTM

BDC 1996 PROCEEDINGS CD-ROM: Final Edition

Welcome to the *final edition* of the 7th Annual Borland Developers Conference Proceedings CD-ROM! The CD-ROM contains all of the papers, code and slide shows that we've received from the speakers (except for pre-conference tutorials).

To move through the conference content, simply click a product track on the image to the left. Product track sesions are presented course order (you can also view sessions arranged by thread and by speaker).

Keynotes	Product Addresses
Opening Keynote by Paul Gross	Latte Product Address by Pat Vermont and Bill Dunlap
Monday Keynote by Dr. Eric Schmidt	Paradox Product Address by Fred Felman

File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Print Font Mail Edit
Links
7th Annual Borland Developers Conference
Delphi
Paradox
Visual dBASE
Borland C++
InterBase
Java
Companion Products
Home
Done
Start
Microsoft Word - vvisual pr...
Exploring - H:\
BDC 1996 CD ROM: ...
12:45

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

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 TstAcess, 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."

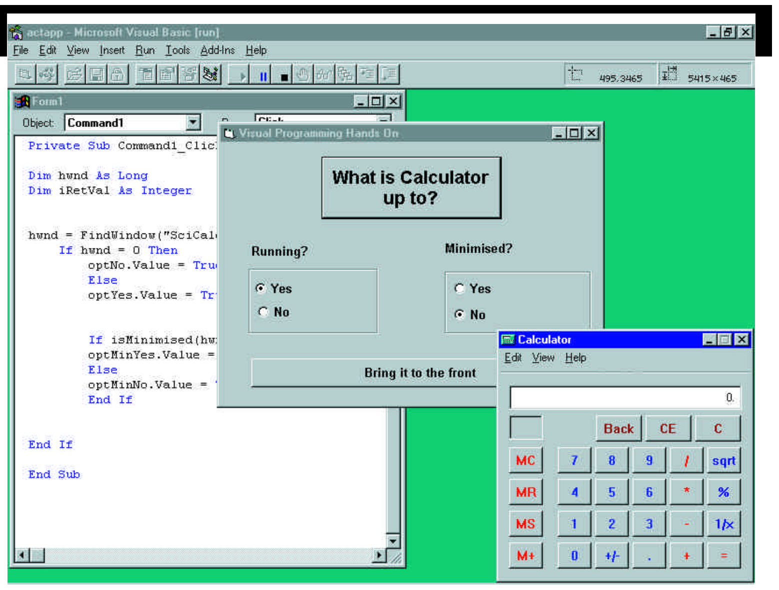
Thanks to the popularity of Microsoft Office Professional and Visual Basic, desktop data is frequently stored in Access MDB files. This creates a problem for other applications which need to get at the data, especially since Microsoft has never documented the structure of an MDB. In any case, the format 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

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



```
Private Sub Command1_Clic

Dim hwnd As Long
Dim iRetVal As Integer


hwnd = FindWindow("SciCal
     If hwnd = 0 Then
          optNo.Value = Tru
          Else
          optYes.Value = Tr


          If isMinimised(hw
          optMinYes.Value =
          Else
          optMinNo.Value =
          End If


End If

End Sub
```

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

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
Public Const SW_RESTORE
```

You can now discover whether a non-VB window is minimised like this:

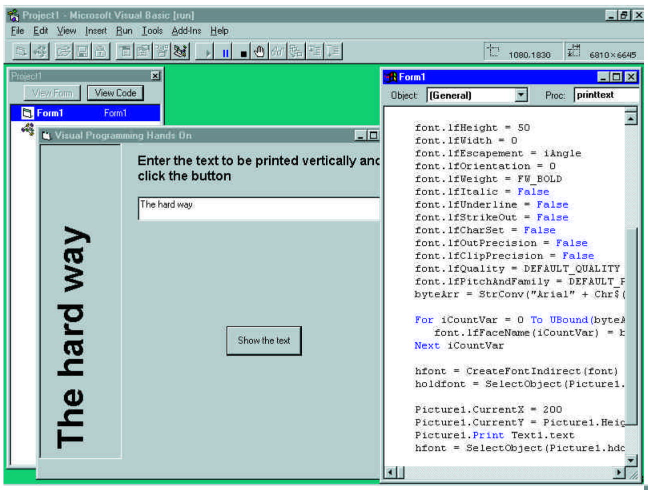
```
Function isMinimised(hwnd) As Boolean

    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.

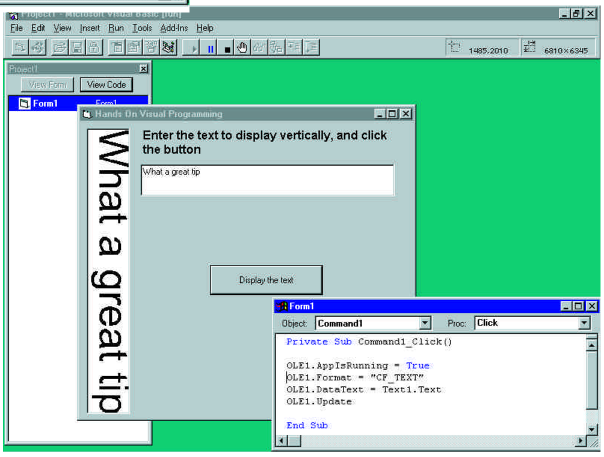
p311 ➢


Project1 - Microsoft Visual Basic [run]
File Edit View Insert Run Tools Add-Ins Help
1080,1830
6810 x 6645
Project1
View Form
View Code
Form1
Visual Programming Hands On
Enter the text to be printed vertically and click the button
The hard way
Show the text
The hard way
Form1
Object: (General)
Proc: printtext

```
font.lfHeight = 50
font.lfWidth = 0
font.lfEscapement = iAngle
font.lfOrientation = 0
font.lfWeight = FW_BOLD
font.lfItalic = False
font.lfUnderline = False
font.lfStrikeOut = False
font.lfCharSet = False
font.lfOutPrecision = False
font.lfClipPrecision = False
font.lfQuality = DEFAULT_QUALITY
font.lfPitchAndFamily = DEFAULT_F
byteArr = StrConv("Arial" + Chr$(

For iCountVar = 0 To UBound(byteA
   font.lfFaceName(iCountVar) = b
Next iCountVar

hfont = CreateFontIndirect(font)
holdfont = SelectObject(Picture1.

Picture1.CurrentX = 200
Picture1.CurrentY = Picture1.Heig
Picture1.Print Text1.text
hfont = SelectObject(Picture1.hdc
```

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

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


Project1 - Microsoft Visual Basic [run]
File Edit View Insert Run Tools Add-Ins Help
1485,2010
6810 x 6345
Project1
View Form
View Code
Form1
Hands On Visual Programming
Enter the text to display vertically, and click the button
What a great tip
Display the text
What a great tip
Form1
Object: Command1
Proc: Click

```
Private Sub Command1_Click()

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

End Sub
```

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 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 Create-FontIndirect. 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 Delete-Object 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