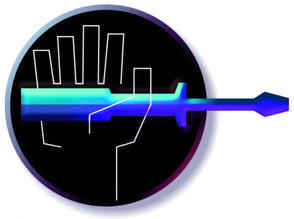
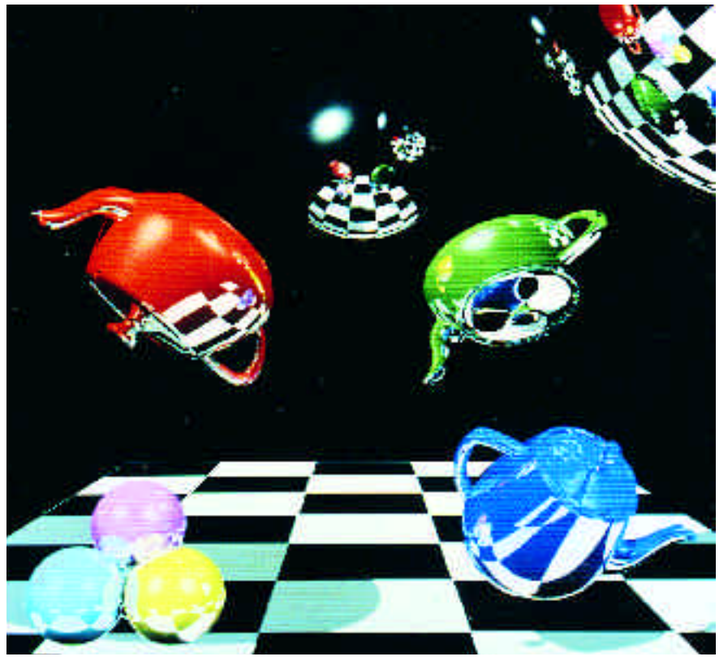
NEW!



Object of the exercise

Tim Anderson takes you through the first in a four-part teach-in about Visual Basic. You'll learn how to get to grips with VB objects and snap together a powerful database application using a few lines of code.



Like it or loathe it, you can hardly avoid it — Visual Basic is the most popular Windows development language. It is also the macro language of Microsoft Office, and with Microsoft now willing to license it to third parties, VB will more frequently appear in third-party products such as the Visio charting package. So time spent learning Visual Basic (VB) soon repays the effort, giving you program control over many powerful applications.

This workshop will show you how to make the most of VB: including data access, automation of other applications like Word and Excel, and using ActiveX components for rapid application development. And along the way you will build a useful application. The software is for managing a sports club but could easily be adapted for a contact manager, book or CD collection, customer database and many other purposes.

Our workshop uses Visual Basic 4.0. It makes use of features introduced in that version so you will not be able to follow the workshop using older editions. A little knowledge of VB is assumed, so complete beginners are advised to become familiar with the product before starting on the workshop.

Objects in focus

Visual Basic makes extensive use of objects. What is confusing, though, is that the word is used in several different ways. Here are three kinds of VB objects:

1. Internal objects and controls. For example, there is a global App object which has useful properties like Title and Path. There are also VB's built-in controls like command buttons and text boxes, represented by objects with properties, methods and events. These objects are VB's essential building blocks.
2. OLE objects. These include ActiveX controls, also known as OCX controls, and applications like Excel which expose functionality in the form of objects you can access from Visual Basic. The advantage of OLE objects is that they are system-wide and not just limited to one application.
3. User-defined objects. You create these by inserting class modules into your project. You can also customise forms by adding your own properties and methods.

If you have used VB at all, you will already have worked with the first two kinds of object but may not have used the third. It is possible to write major VB applications without using them, especially if either the application or the developer started in Visual Basic 3.0, where they did not exist. In fact, the Visual Basic environment does not encourage you to use them.

The obvious way to build an application is to draw buttons and controls onto a form, setting their properties and writing code for their events. With that approach you may not see the need to define your own objects. It is worth making the effort. Here are three reasons why:

1. Object-orientated programs are more robust and easier to debug. One reason for this is that you can avoid global variables, which are notoriously error prone, and use object properties instead.
2. Well-designed objects can be used in more than one application.
3. To exploit the power of OLE (Object Linking and Embedding) you need to define objects that can be made available to other applications.

This workshop will explore how to make use of VB's class modules, which create user-defined objects, in order to derive these benefits.

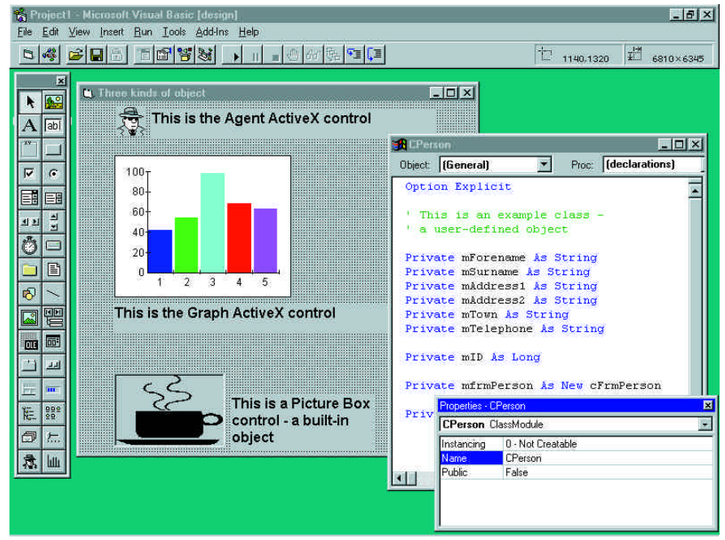


Fig 1 Not all VB objects are the same. This application shows three kinds: built-in, OLE, and user-defined

Building a database application

Anyone can build a Visual Basic database application. Just place a data control on a form, set the databasename and recordsource properties, add some bound text boxes to display the fields, and it's done. There is even an add-in that will do it for you; the data form designer. The typical result is shown in Fig 2.

The speed of development is impressive, but in other respects applications built like this are poor. For a start, a visible data control is not the world's most stylish graphical interface. Worse, it encourages a navigational approach to viewing data. If the visible record is not the one you want, click Next until you find it. It may work for half a dozen records but it's hopeless for large database tables. It is also fundamentally at odds with the set-based strategy of SQL, the native query language of VB's database engine. Additionally, working with bound controls increases the risk of inadvertently changing the data. All these problems can be overcome by adding code for searching, validation, and so on. Another option is to use an entirely different approach.

A particularly powerful technique uses a listbox and a text box to create a database searcher. The user types one or more letters into the text box and presses Enter. The listbox then fills with all the matching records. By double-clicking an entry in the list, the full details of the record can be displayed. It allows control over the precision of the search, and it is fast, with no need to enter criteria into a search dialogue.

Will the real VB stand up?

Visual Basic exists in various forms. The standalone product comes in three editions: Standard, Professional and Enterprise.

The Standard edition is cheap but not all that cheerful. It is fine for learning the Basic language but data access is limited to the data control. Few custom controls are included and it is unsuitable for creating applications for distribution. It works only on Windows 95 or NT. The Professional edition fills the gaps, includes a 16-bit version, full data access, important OLE features and a wide range of custom controls. For general-purpose work, the Professional edition is all you need. The Enterprise edition adds features for client-server work and team development.

That leaves two other types of VB. Visual Basic for Applications is the version that ships with Microsoft Office and now a number of third-party applications, too. VBA in Office 95 has no forms engine, which limits its power, but VBA 5.0 in Office 97 is almost the same as the standalone version. The main difference is that you cannot compile a standalone executable. VB Script is a stripped-down language for Internet Explorer. Microsoft hopes that other web browsers will adopt VB Script, too, although so far this has not happened.

p248 ➢

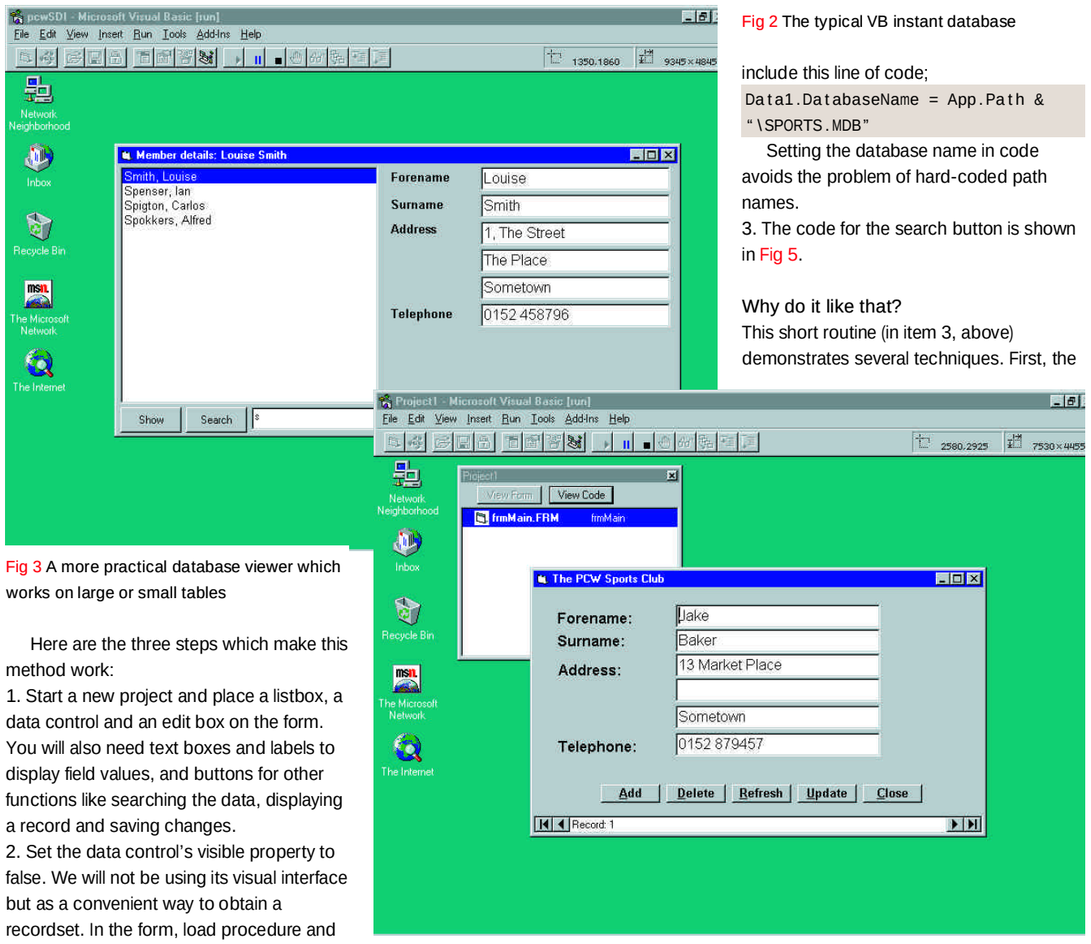


Fig 3 A more practical database viewer which works on large or small tables

Here are the three steps which make this method work:

1. Start a new project and place a listbox, a data control and an edit box on the form. You will also need text boxes and labels to display field values, and buttons for other functions like searching the data, displaying a record and saving changes.
2. Set the data control's visible property to false. We will not be using its visual interface but as a convenient way to obtain a recordset. In the form, load procedure and include this line of code;

```
Data1.DatabaseName = App.Path &
"\SPORTS.MDB"
```

Setting the database name in code avoids the problem of hard-coded path names.

3. The code for the search button is shown in Fig 5.

Fig 2 The typical VB instant database

Why do it like that?

This short routine (in item 3, above) demonstrates several techniques. First, the code uses SQL to create a dynaset-type recordset based on the text the user has entered. By adding the star character to the string and using the Like keyword, we find all the surname fields which begin with that string. JET, the Visual Basic database engine, is not case-sensitive, which simplifies matters. A nice feature is that the user can enter wildcards. For example, the string "??i" finds all surnames with a third letter i. Your users will think this is very clever, but it is VB's SQL that has done the work for you.

Second, the code uses a standard list box rather than the databound list box or the bound grid control. Using a databound control would save the few lines of code which fill the list. But unfortunately, the bound list control can only display one field, limiting its use. The databound grid is a viable option but is, frankly, overkill in view of what's required. In version 4.0, Microsoft enhanced VB's list box by adding the ItemData property and this is ideal since it lets you store an ID number against each item in the list. It is then easy to look up the correct record when the user selects the item.

The underlying principle is not to use a complicated ActiveX control where a simple, lightweight VB control will do just as well.

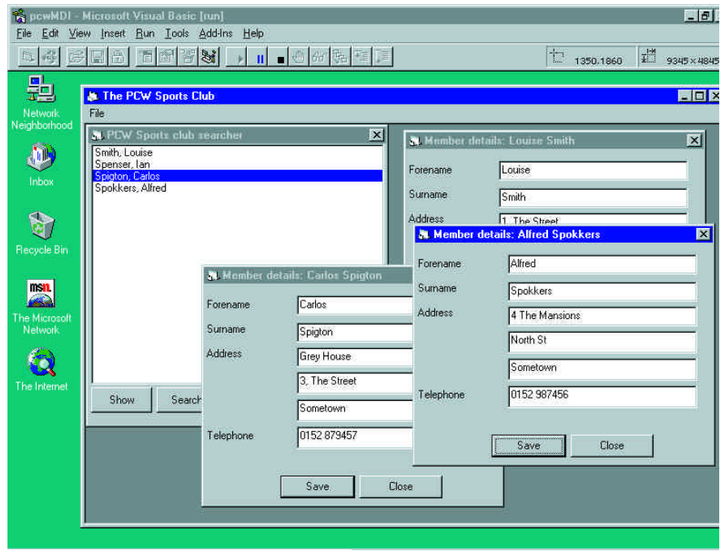


Fig 4 This alternative approach lets you view several records at once

Fig 5 Code for the search button

```
List1.Clear

' now do the search
Data1.RecordSource = _
"select * from members where members.surname like '" _
& Trim$(txSearch.Text) & "*' order by members.surname"
Data1.Refresh

' now fill the list box
If Not (Data1.Recordset.BOF And Data1.Recordset.EOF) Then
' there are matching records

Data1.Recordset.MoveFirst

Do While Not Data1.Recordset.EOF
List1.AddItem (Data1.Recordset!surname & ", " &
Data1.Recordset!forename)
List1.ItemData(List1.NewIndex) = Data1.Recordset!ID_NO
' stores the ID in the list box

Data1.Recordset.MoveNext
Loop

List1.ListIndex = 0 ' select first matching record
cbShow_Click ' show the first record

Else
' add code here to clear the form's fields, report no match, etc

End If
```

Putting objects to work

Not all the code is shown here (for reasons of space) but if you look at the example project on our cover-mounted CD you will notice a class module, CPerson.

The application maintains an instance of the CPerson class and obtains member details by inspecting its properties. The Save button works by calling the save method of the currPerson object. This approach will bring several advantages as the application develops. For example, a weakness of traditional database forms is that they only show one record at a time. Fig 4 is a database application which uses an enhanced CPerson class that has the capability to display itself. That makes it easy to simultaneously view the details of several individuals.

■ *Next month: A closer look at VB class modules.*

•PCW Contacts

Tim Anderson welcomes your VB comments and tips. Contact him either by post c/o *PCW* or email at visual@pcw.vnu.co.uk

For more information about Visual Basic, contact Microsoft on 0345 002000.