Dealing with **databases**

In part II of his Delphi tutorial, Tim Anderson examines the program's abilities as a data management tool and gets you started on building your own database applications.

Delphi is a great all-purpose development tool, but it is particularly strong for database work and can handle anything from a simple address book to a large multi-user system. This workshop explains how Delphi manages data and will get you started with building your own database applications.

A quick-start Delphi database

Try the following example to see how Delphi deals with data.

1. When Delphi is installed, an example Paradox database called DBDEMOS is installed with it. If you run the BDE configuration or administration utility, you should find it listed. DBDEMOS is called an Alias. What that means is that instead of referring to the exact location of the data files in your application, you can use the name DBDEMOS instead. Another advantage is that you can configure the way the BDE accesses this particular database just once. Use of an alias is not compulsory, but it does simplify data access. This example uses DBDEMOS, so it is a good idea to check that it is installed (Fig 1).
2. Start a new Delphi project. Place a DataSource and a Table component on a form. Select the Table component and set its DatabaseName property to DBDEMOS and its TableName to CUSTOMER.DB.

Now select the DataSource and set its DataSet property to Table1.

3. Place a DBGrid on the form and set its DataSource property to DataSource1.
4. Select the Table component and set its Active property to True. All going well, the grid fills with data, even at design time.
5. Place a DBNavigator control on the form and set its DataSource property to DataSource1.

• *Tip: Set the Align property of the DBNavigator to Bottom. When the user resizes the form, the navigator will automatically resize itself to fit neatly along the bottom edge.*

6. Run the application. You can scroll up and down the grid to view all the records in the table. You can use the Insert Record button on the navigator control to insert a new record, and the

Code for the search facility

```
procedure TForm1.Button1Click(Sender: TObject);
var
sSearchVal: string;
begin
sSearchVal := InputBox('Search','Enter a company name','');
Table1.SetKey;
if sSearchVal <> '' then Table1.FindNearest([sSearchVal]);
end;
```

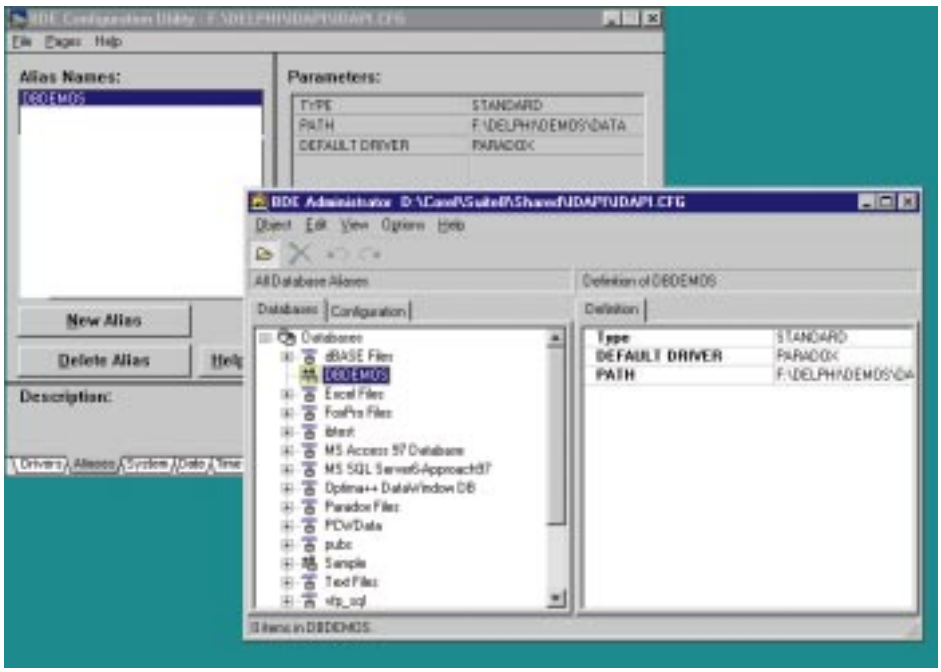


Fig 1 The rear window shows the BDE configuration utility for Delphi 1.0, while the front window is for Delphi 3.0. In both cases there is an alias called DBDEMOS

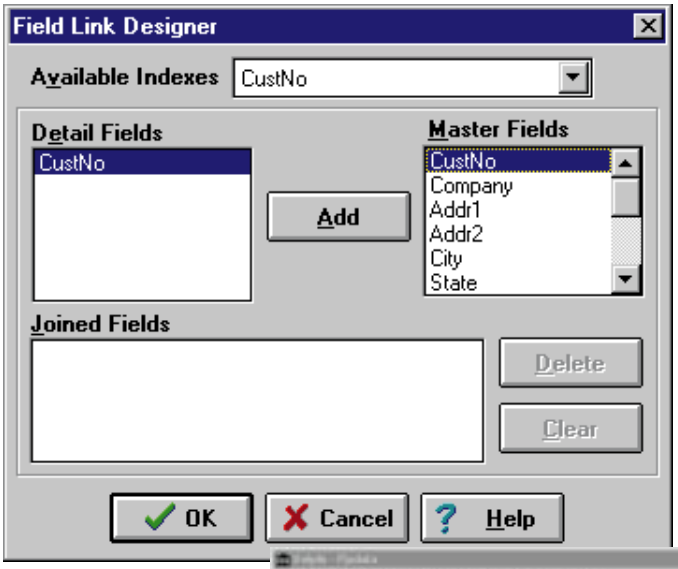
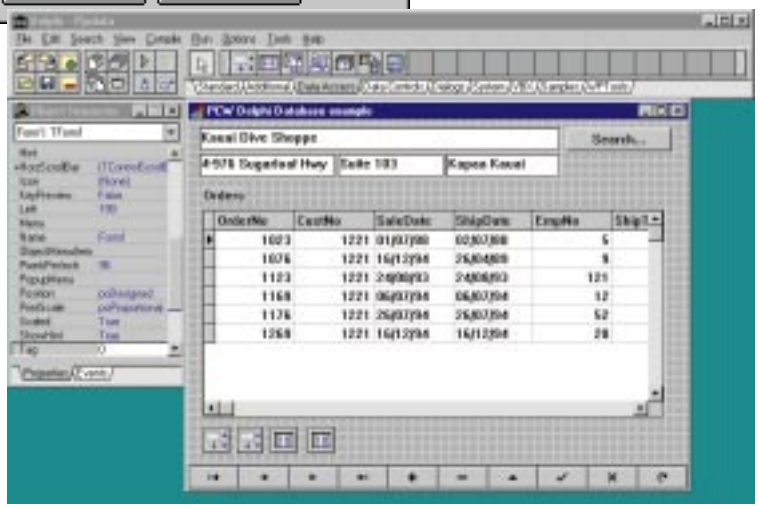


Fig 2 (left) Use the Field Link designer to set up a master-detail relationship. In this, the CustNo field links the Customer and Orders tables

Fig 3 (below) The database includes a master-detail relationship. For each customer, the associated orders are displayed in the grid



Post Edit button to save it. You can use the Delete Record button to remove a record — not bad for an instant database application.

• *Tip: Set the ShowHint property of the form to True. When you hover the mouse over the DBNavigator buttons at runtime, a hint tells you what each button does.*

7. The records are currently unsorted. To sort the records by Company, select the Table component and set its IndexName property to ByCompany. Now the records in the grid are sorted.

8. Finally, a glaring omission in the application is the lack of a search facility. Pop a button on the form, double-click to open the code editor, and enter the code for the search facility (see opposite page).

• *Tip: InputBox is a handy function when you need a quick way to get a search value from the user. Select the word InputBox in your code, press F1 to open online help, and see what the parameters do.*

This routine will only work if the table's IndexName property has been set. Delphi will look for a match in this index. Note the square brackets around the search variable — needed because FindNearest accepts an array parameter.

Dealing with master-detail data

You can easily enhance the example above to deal with a master-detail relationship (Fig 2). Here is how it works.

1. Add a second DataSource and Table component. Set the DataSet property of DataSource2 to Table2.

2. Select the Table2 component and set its DatabaseName to DBDEMOS. Then set its TableName to ORDERS.DB.

3. Table2 is the detail table. In this next step, it is linked to the master table of customers. Set the MasterSource property to DataSource1. Select the MasterFields property, and click the three dots to open the Field Link Designer. Under Available Indexes choose CustNo. Select the CustNo field in both left and right field lists, and click Add. This operation tells Delphi to look up the orders for each customer based on the value of the CustNo field in each table.

4. Currently, the DBGrid points to the Customer table. Change its DataSource property to DataSource2 so that it points to the Orders table instead. Now add several DBEdit controls above the grid and set their DataSource property to DataSource1. Set the DataField property of each DBEdit to a suitable field, such as Company, Addr1 and City. Fig 3 shows a possible design for the form. Make sure the Active property of the Table components is set to True and run the application.

Other database components

When you are starting with Delphi, the Table component used in the above example is the quickest, easiest way to display data. Surprisingly, many Delphi

Defining your own database

Demonstration databases are all very well, but you will soon want to use your own data. To define your own tables, run the Database Desktop (a separate application that gets installed with Delphi) and choose File, New Table. When asked to select a table type, choose Paradox. The Create Table dialog opens, where you enter field definitions that define how data is stored. It is essential to do a good job at this stage, otherwise your application will never work well. Here are some tips:

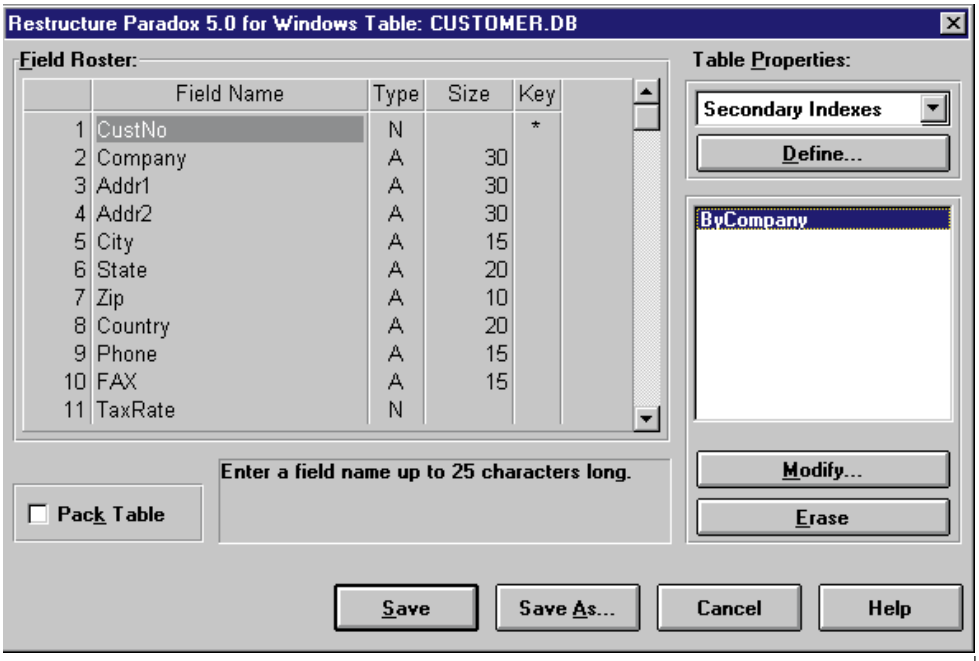
- Include a unique identifier for each record. This is essential when you want to link several tables together, as with books and authors. An obvious technique is to use an auto-increment field for this, but beware: in most versions of Paradox this is not safe, as the value may change if the table is rebuilt. Most developers generate their own unique ID value. Mark this as the Primary Key by putting it at the top of the field list and double-clicking the Key column so an asterisk appears.
- Index other fields that you are likely to use for searching or sorting. Because Paradox uses the Key column for the primary key, other indexes should be defined as secondary indexes. Ensure all indexes are marked as maintained.
- Take full advantage of the Paradox facilities: for required fields or other validity checks for example. It is safer and easier to let the BDE handle validity checks, than to do so in your code. For instance, someone may open the table using Paradox rather than your application. In this case, rules you have defined in the table structure dialog will still hold, but those defined in your application will not.

p244 ➢

How Delphi manages data

Delphi is a Borland product and Borland is the company which developed the Paradox database manager, first for DOS and later for Windows. Paradox is now marketed by Corel but the grunt-work of data access in Paradox is still handled by Borland code.

Borland has a couple of other database products: dBase and InterBase. Like Paradox, dBase is a desktop database designed for standalone use or for small networks of no more than a dozen or so users. The format of the data files is different and you will notice that Paradox files have a .DB extension, while dBase files end with .DBF. Borland decided to combine the code for dBase and Paradox, so that one low-level component could handle data in either format. This code, actually a set of dynamic link libraries, is called the BDE (Borland Database Engine). When Delphi was created, it made sense to include the same code there, as well. That means Delphi has native access to dBase and Paradox data using exactly the same code libraries as the full products. The Database Desktop, bundled with Delphi, is actually a cut-down version of Paradox.



The dialog for defining or restructuring a database table. It is part of the Database Desktop, a powerful interactive tool for managing data

Borland's InterBase is a different kind of product. It is a client-server database manager. In a dBase or Paradox system, each machine on the network runs its own data management code, accessing shared files on a central server. In a client-server system, the data management code runs on the server and the client machines run applications which despatch query requests and then receive the results. The advantage of client-server is partly that the server database can manage the data more intelligently, and partly that network traffic is reduced because only the query and its results need to be passed back and forth. The result is that InterBase and other server databases, like Oracle and SQL Server, can handle many more users and manage databases beyond the capacity of Paradox or dBase. For maximum flexibility the BDE can handle these server databases via plug-in drivers. You can also access databases through Microsoft's standard for database drivers, ODBC. Overall, the BDE can handle three kinds of database:

1. Paradox or dBase natively.
2. InterBase and other server database via native drivers.
3. Any ODBC database.

To manage all these combinations you will find a utility called Database Engine Configuration in Delphi 1, or BDE Administrator in Delphi 3. The latest BDE can also handle Microsoft Access databases, but only if they are in Access 95 format.

• ***Tip:*** *The BDE works best with dBase, Paradox, or its own native drivers. Use ODBC or Access data only as a last resort. Use Paradox rather than dBase, unless you have a reason to prefer the dBase format. Paradox data files have useful additional features.*

You do not have to use the BDE with Delphi. Although it is powerful, the database engine is large and memory hungry. For small databases, using Delphi's low-level file functions is more work to program but fast and efficient.

There are alternative third-party database libraries for handling dBase or FoxPro data, or for using ODBC or Access databases without going through the BDE. These solve compatibility problems, are smaller and often perform better than Borland's engine.

developers hardly ever use it. The reason is that you seldom want to view a whole table at once; more often, only a small subset of the database is required. Table components can perform badly on large databases.

The other factor is what happens when you need to scale your application up to a server database. In this case, you need an easy way to query and update the data using SQL (Structured Query Language), the common language of server database managers. Delphi has a Query component for exactly this purpose. You can use Query components on Paradox and dBase data as well, so there is no reason why it should not form the basis of all your database applications. It is a little harder to use, though, since you must be familiar with SQL.

Delphi also has a Database component but this is not necessary for simple apps. Internally, Delphi uses a default Database object instead. It is useful for systems where you need to log on with a user name and password, or if you are using a server database and want to use explicit sessions and transactions to ensure data integrity.

Another important database component, in Delphi 2.0 and higher, is the Data Module. This is invaluable if your application has more than one form: instead of putting data components on every form, you can add them to a single data module. Each form then links to the data module by referencing it in the Uses clause of its unit. This is both simpler and more secure than having data components dotted all over your application.

■ *The code for the database example is on our cover CD-ROM, in the file DELPHI2.ZIP.*

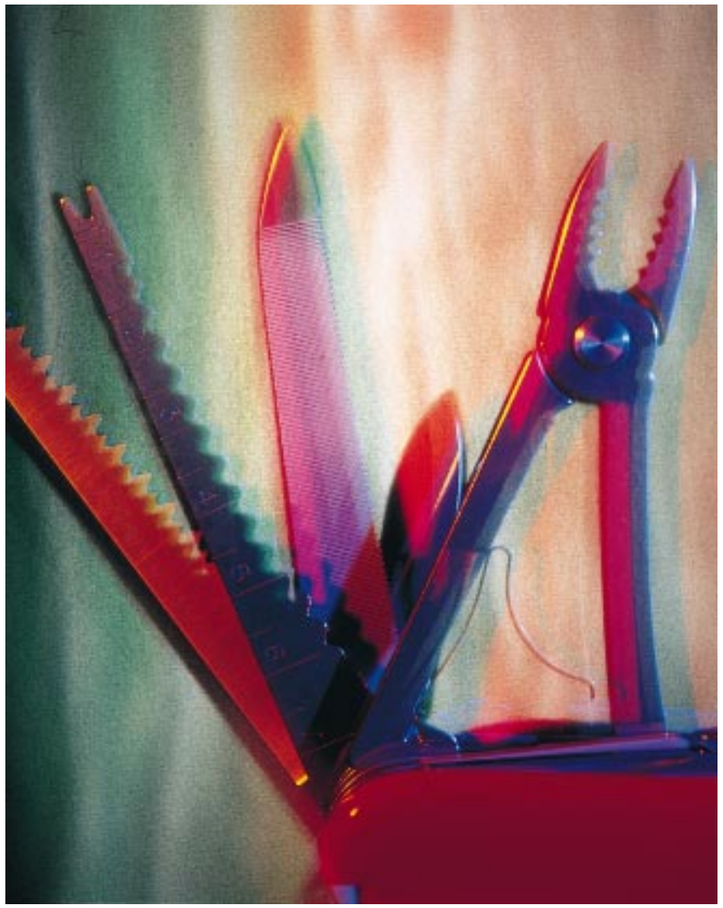
PCW Contact

Delphi queries are regularly tackled in *Hands On Visual Programming*. You can contact **Tim Anderson** with your queries and tips at the usual *PCW* address (see page 12) or at visual@pcw.vnu.co.uk

Loan star

Delphi is a good solution to creating software. In the first of a four-part series, Tim Anderson explains how to use it and shows you how to create a working loan calculator program.

If you want to create software that is fast to develop and speedy at runtime, Delphi is the solution. Delphi 1.0 is on this month's cover CD, so you have all you need to get started.



Why Delphi?

Delphi is a gem. It delivers three of the key features most sought-after by developers:

- **Rapid development:** On the surface, Delphi looks a lot like Visual Basic. Add a form, pop on a control or two, double-click a button to add a few lines of code, click Run, and away it goes.
- **Native code compilation:** Delphi compiles to true executables that potentially run as fast as apps created in C or C++.
- **Object orientation:** Delphi supports classes with constructors and destructors, inheritance, encapsulation and polymorphism; all the essential characteristics of an object-orientated language. You can also program procedurally in Delphi, so object purists may prefer languages like SmallTalk or Java. In many ways, though, Delphi is the best of both worlds: easy to learn if you are familiar with Visual Basic or xBase, but with all the benefits of objects available, as well.

Is there a catch? Not really, although Delphi is a little harder than its main rival, Visual Basic. It supports pointers (variables that contain memory addresses) and mistakes in this area can easily crash your application, or even Windows itself, particularly if you are using Windows 3.1. Another feature is strong typing, which means the type of each variable has to be specifically declared. These things are actually strengths, not weaknesses, but it means beginners have more to learn.

Topics covered in this workshop will include creating your own classes, database applications, ActiveX controls and internet programming. This first part is introductory, so anyone can get started.

Which Delphi?

There have been three releases of Delphi:

1. Release 1.0 is for 16-bit Windows only, although applications you create run fine on Windows 95 or NT.

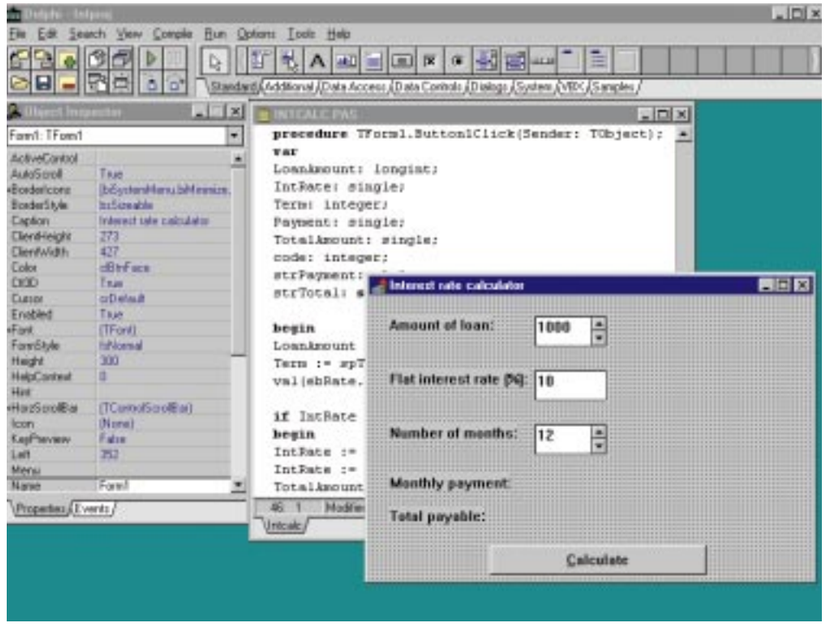


Fig 1 **Laying out the Interest Calculator in Delphi. Note the component palette, object inspector, code editor, and form designer**

2. Release 2.0 is the 32-bit version which has now been replaced by Delphi 3.0. Borland kindly bundles Delphi 1.0 with the 32-bit releases, so you only need buy one product.
3. To confuse matters, Delphi 3.0 comes in three versions:

- The entry-level Standard version is far from useless. It is not crippled and you can build excellent applications with it.
- Serious developers will want the Professional version, which adds 30 Delphi components, the source code for Delphi's component library, the ability to create ActiveX controls, and ODBC database connectivity.
- The Client-Server version includes drivers for leading SQL databases like Oracle and SQL Server, together with other tools for enterprise development.

Getting started

To get you going with Delphi, here is a simple but useful application. It is a loan payment calculator that you can use to find out what that new stereo is going to cost you per month. For simplicity it uses the flat-rate method, so it calculates the interest as if the whole amount were always outstanding. You can have fun adapting it for other methods. To create it, follow these steps. It will work in any version of Delphi.

Creating a loan payment calculator

1. Start a new project. Place on the blank form a button, several labels, two SpinEdit controls, and an edit box, as shown in Fig 1. The SpinEdit is found on the Samples tab of Delphi's component palette, while the others are on the Standard tab. To place a control, click on the palette to select the control you want and then click-and-drag on the form.
2. Next, use the object inspector to customise the properties of the controls. Select the control, then enter the required values in the object inspector's grid. Most can be left at the default. The Name property is important, and a good tip is to name the controls in a way that reminds you of their function. Therefore, the two SpinEdit controls are called **spAmount** and **spTerm**. SpinEdits have a MaxValue and MinValue property (which work as you would expect) and an Increment property which determines the amount of change when you click the up and down arrrows. The Caption property for the form, button, and label controls is used to set the text.

Why not use a SpinEdit for the interest rate as well? Simply because the SpinEdit does not support floating-point numbers. Of course, there are ways to overcome this problem, but to keep things simple an Edit box is used instead.

3. Double-click the Calculate button to open the code editor. Delphi automatically creates and opens a procedure called Button1Click (or whatever-you-named-the-buttonClick). The code to enter is shown in Fig 2 (*p236*).

If you are used to Visual Basic, the code will look familiar. The main points of difference are explained in the panel on Object Pascal (*alongside*). Note that the **Str** function is different from the Visual Basic equivalent. Look it up in online help to see how it works. If you have Delphi 3.0, you can improve the formatting of the results by

p236 ➢

Key features of Object Pascal

The language of Delphi is Object Pascal. Here are some key features, especially aimed at Visual Basic users.

- Object Pascal is not case-sensitive.
- As with C and Java, each statement (line of code) ends with a semi-colon. If the line is too long, you can break it up more or less as you want, but the compiler will read it as a single statement until it sees the semi-colon.
- Comments appear within curly brackets and are ignored by the compiler.
- Array elements use square brackets
- Pascal has the concept of compound statements. This is a series of statements that are to be treated as one block. They are enclosed by the keywords "begin" and "end". Read the box entitled "Beware if…then" (*p236*) for an example.
- Pascal uses different operators for assignment and comparison.

For example:

```
myvar := 1000;
```

sets myvar to the value 1000. To test for equality, use:

```
if myvar = 1000 then …
```

- Strings in Pascal are enclosed in single quotes.
- In Delphi 1.0, String variables can be a maximum of 255 characters. Longer strings are handled by a pointer variable called a Pchar. ■

Fig 2 Code for the click event

```
procedure TForm1.Button1Click(Sender: TObject);
var
LoanAmount: longint;
IntRate: single;
Term: integer;
Payment: single;
TotalAmount: single;
code: integer;
strPayment: string;
strTotal: string;

begin
LoanAmount := spAmount.value;
Term := spTerm.value;
val(ebRate.text,IntRate,code);

if IntRate > 0 then
begin
IntRate := IntRate/100;
IntRate := IntRate * (Term/12);
TotalAmount := (LoanAmount * (1 + IntRate));
Payment := (TotalAmount /Term);
str(TotalAmount :12 :2, strTotal);
str(Payment :12 :2,strPayment);
lbResult.Caption := 'Monthly payment: £' + strPayment;
lbTotal.Caption := 'Total payable: £' + strTotal;
end
else
begin
lbResult.Caption := ('Invalid interest rate');
lbTotal.Caption := '';
end;

end;
```

using the **TrimLeft** function to strip leading spaces.

4. Now click the Run button to test the application. The code runs in Delphi, and you will also find a file called INTPROJ.EXE in the directory where the code is saved. This is a native code executable which will run anywhere. You do not need any supporting files.

All about Delphi components

Delphi is a component-based tool. It is the component palette that provides fast and robust development. Components make more reliable applications because they are pre-built and already debugged. Another feature is that if you update a component with an improved version, those changes can easily be migrated to all the applications that use it, simply by re-opening the project in Delphi and recompiling.

Delphi can use three kinds of component. The first and best kind is native to Delphi. These are pure Pascal code, created in Delphi, and with some special features that allow them to be installed in the component palette. Native components have four significant advantages:

1. Performance is optimised because, to Delphi's compiler, it is simply another chunk of Pascal code.
2. Deployment is easy since the component is bound into the final executable. In Delphi, you also have the option to make a package which enables several applications to share the same component.
3. If you have the Pascal source, you can use Delphi's debugger to step through the component code as well as your own application code.
4. You can create your own custom components, either from scratch or by adapting an existing one.

Native negatives

While it is best to use native components most of the time, there are two problems. One is that there are not as many third-party products for Delphi as for Visual Basic. Another is that you cannot use a native component in other applications — Internet Explorer, for example. Therefore, Delphi can also use Visual Basic components.

In the 16-bit Delphi 1.0, that means VBX controls, and in Delphi 2.0 or 3.0, that means ActiveX controls. You install these using the Install Components option in Delphi 1.0, or the Import ActiveX control option in Delphi 3.0. In both cases, Delphi creates a wrapper Pascal file to be the bridge between your code and the component. This is transparent to the user, and you can use the Visual Basic control as if it were a native Delphi component.

Sometimes you may run into problems. String properties in VBX controls are Delphi strings and therefore limited to 255 characters. Visual Basic strings can be up to around 65,000 characters long. There are ways to overcome this and other snags, but extra work is involved. In 32-bit versions of Delphi this string problem does not occur, but you may sometimes have compatibility problems

p238 ➢

Beware if … then

When you start with Delphi, it's easy to write code like this:

```
if myvar = 1000 then
  dothis;
  dothat;
  dotheother;
```

This will compile fine, but you may not realise that the second two statements will always execute, whatever the value of myvar. If you want all three statements to be in the conditional code, you need to do it like this:

```
if myvar = 1000 then
  begin
  dothis;
  dothat;
  dotheother;
  end;
```

The else part can be confusing, as well. You need to make sure there is no semi-colon before the else. You can see why if you think of the simplest case:

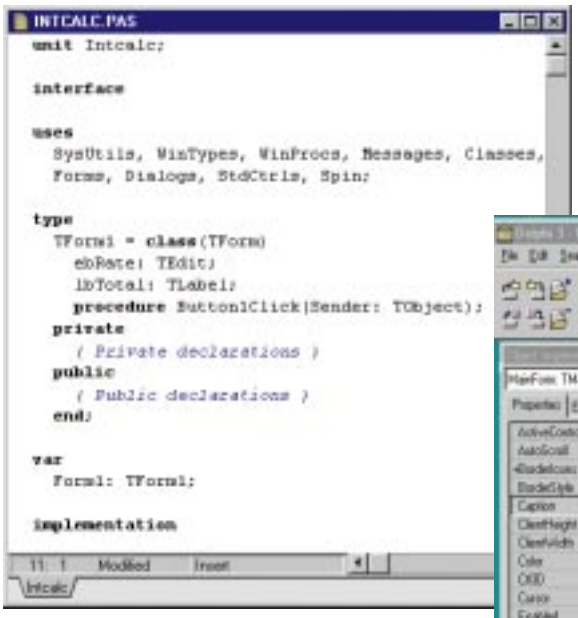
```
if myvar = 1000 then dothis else dothat;
```

It would look strange to have a semi-colon after dothis, and the same applies when you expand it with begin … end; ■

What's in a unit?

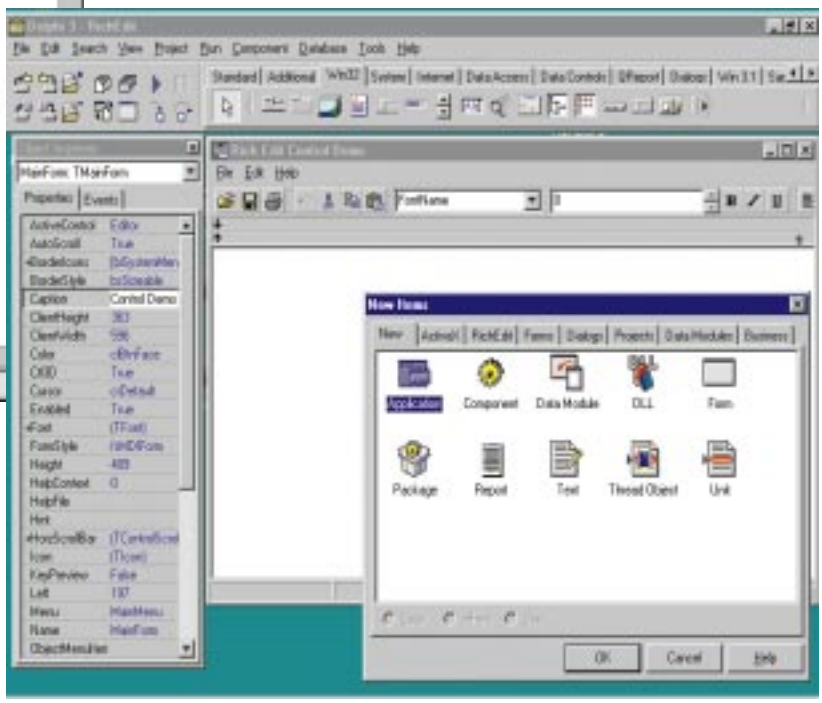
Delphi's equivalent to Visual Basic code modules is called a unit. Each form has its own unit and you can add others, too. All units have the same structure, which in essence goes like this:

```
unit myunit; {names the unit}
interface {introduces the part which is exposed to the outside world.}
uses … {names the other units which this unit refers to in its code.}
{Declaration part – declares constants, types, procedure and function headings, and public variables}
implementation {introduces the part which contains the code that lies behind the interface}
{procedures and function bodies go here}
end; {marks the end of the unit}.
```

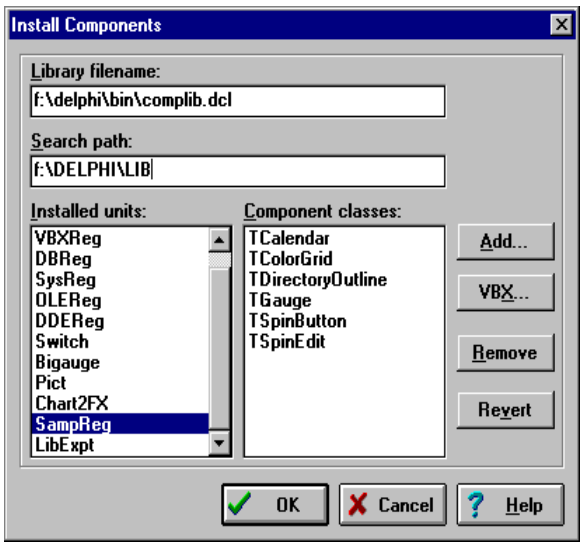


Left A programmer's dream: Delphi has speed, rich features and full object orientation. This is version 3.0

Below Part of the unit used by the Interest Calculator application (*p235*)



What you may not realise when starting out with Delphi is that the implementation part can have its own "uses" clause and declaration part. This goes after the implementation heading and before the procedure and function bodies. Anything here is private to the unit, and good programmers keep everything as private as possible. You can also include an initialisation part just before the end of the unit. This is for code that runs automatically when the application starts up. Many Delphi applications do not use this feature. ■



Use this dialog to install a component in Delphi 1.0. Click the VBX button to install a Visual Basic control

because of the underlying complexities of ActiveX.

When you install components, Delphi compiles the Pascal wrapper into a library file. In Delphi 1.0 this is a single file with a .DCL extension. If this gets corrupted for any reason Delphi will not run, so a good tip when installing components is to use a new name for your customised library. That means the original COMPLIB.DCL, which contains the default Delphi components, is preserved in case of emergency. Delphi 3.0 is more flexible, and stores components in a number of different packages.

■ *Next month: Delphi databases*

PCW Contacts

Tim Anderson welcomes your comments and queries on any aspect of this workshop or visual programming in general. Write to him c/o *PCW*, or email visual@pcw.co.uk.

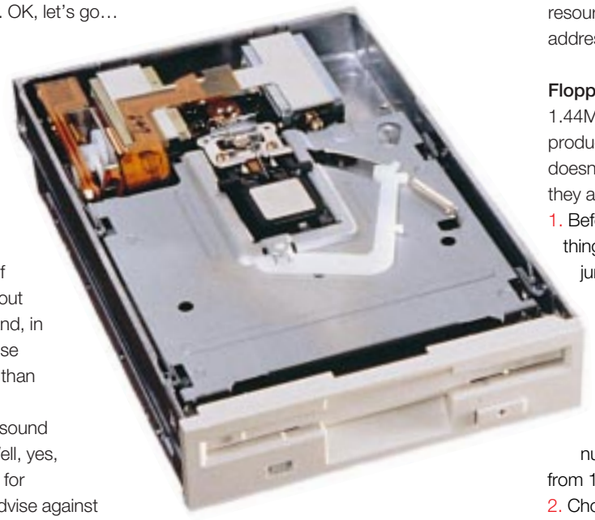
Drive on

Nearly there. It's all coming together now, and in part IV of his series on how to build your own computer, Roger Gann tells you how to install the expansion cards and fit the drives.

For those who came in during the interval, it's been a roller-coaster ride of a project! Yes, you've missed some well-crucial instalments. (You can obtain back issues of the magazine — see the panel, *Back Issues*, p214).

OK, to recap: in part one I covered the pros and cons of the build-it-yourself PC and led you through choosing the motherboard and the system unit. In the second part I looked at specifying all the components needed to build your PC; choosing a games PC for the simple reason that it represents the most powerful and well-specified PC you can currently buy. I then started to prepare the motherboard. Last month, we got around to putting in the motherboard and began installing the expansion cards. This month, I'll complete this and fit all the drives, the floppy and the hard disk. OK, let's go…

New breed of super-floppies: Iomega's Zip drive (top), with OR Technology's LS-120 (beneath)



Expansion cards

With so much I/O now integrated on the motherboard, the number of expansion cards required by a modern personal computer can often be counted on the fingers of one hand. I'm talking about the graphics card here and, in 99 out of 100 cases, these require no further action than simply plugging them in.

"OK, what about the sound card?" I hear you say. Well, yes, this is a prime candidate for installation but I would advise against fitting it at this juncture, simply because we won't be in a position at this stage to avail ourselves of the miracle that is Plug and Play, and believe me, this can make life a lot simpler. So hold off for a while until we've installed our operating system (Windows 95) and let it sort out the arbitration of the multitude of hardware resources, the IRQs, DMAs and I/O addresses needed by such devices.

Floppy disk drive

1.44Mb floppy drives are now generic products, almost unbranded, and so it doesn't really matter which one you buy as they are all much of a muchness.

1. Before fitting it in a drive bay, there's one thing to do: check that the Drive Select jumpers (or switch) is set to the first drive. A PC can support up to four floppy drives and this is a way of discriminating between them. It may be Drive 0 (DS0) or 1 (DS1) depending on the drive you've bought. Some start their drive numbering from 0, while others start from 1.
2. Choose a suitable 3.5in drive bay and unclip the blanking panel from the system case front bezel.
3. Slide the floppy drive into the bay and when it is flush with the front panel, insert the mounting bolts and tighten them up.
4. Plug in the power lead (it will be the smaller of the two types of power-lead plug).
5. Using the 34-way ribbon cable which came with the motherboard, plug one end into the connector at the back of the drive and the other into the floppy interface connector on the motherboard. Ensure that the coloured edge of the ribbon cable aligns with Pin 1 at both the drive and motherboard ends.

This cable may have an extra connector for a second floppy disk drive. It may also have a "twist" in the middle of the ribbon cable, between the two FDD connectors. This twist is a production convenience for PC manufacturers because it lets them set all their floppy disk drives to be the second option (e.g. DS1 or 2) and they let the choice of connector used (i.e. before or after the twist) determine whether it's a Drive 1 or 2.

In this case, with your drive set to Drive 1, make sure you use the connector that precedes the cable twist. Past experience has taught me that, for reasons unknown, sometimes you have no choice but to set the floppy to Drive 2 and use the twisted part of the cable in order to get the floppy drive to be recognised. So prepare for a degree of faffing around when getting the system to recognise the drive.

6. With the drive installed, enter CMOS Setup and specify that you have a 3.5in floppy installed. Most BIOSes default to this anyway so you may not have to change anything.
7. Quit Setup, saving your settings, and reboot. At this point it makes a lot of sense to test that we can boot the PC from our new floppy drive before proceeding further. So bung in a bootable floppy and hit the Reset button, crossing your fingers in the meantime.

The Quantum Bigfoot, like most IDE hard disks, requires a 5.25in drive bay

Other floppy drives

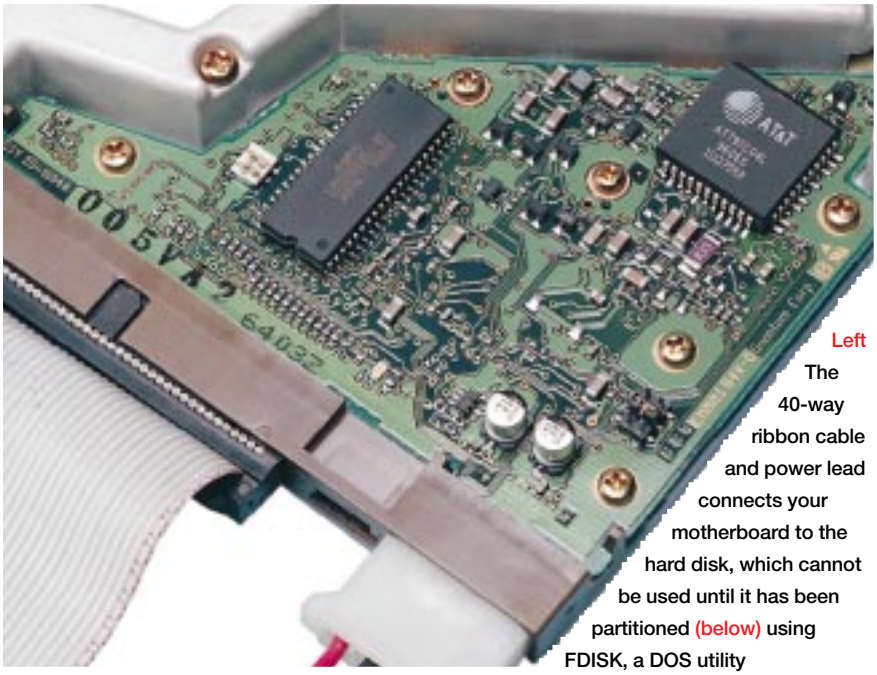
If you have a really modern motherboard you may find that it supports the new breed of super-floppies as bootable devices — I'm talking about Zip and LS-120 drives, here. At present, internal Zip drives (about £90 ex VAT) require a SCSI host adapter in order to work, which is outside the scope of this project. However, the LS-120 (about £160 ex VAT) is an ATAPI device and can be easily attached to the second IDE channel on our motherboard. It's also directly supported by the OSR2 release of Windows 95.

Externally an LS-120 looks just like a normal 1.44Mb/3.5in floppy drive but at the rear it has a 40-way rather than a 34-way connector. It also has jumpers for Master and Slave settings. If you are going to fit an LS-120 then it's probably best to attach it to the second IDE channel as a master, and slave the CD-ROM off it.

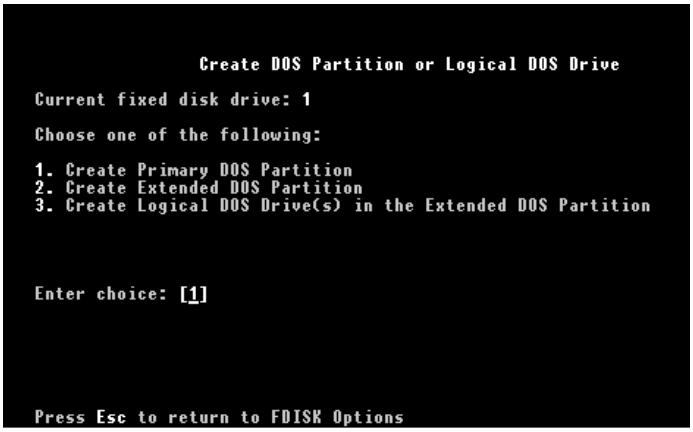
The hard disk

1. Check out the new drive's fixings and where it's going to fit in the system case. It won't need an externally accessible drive bay so look first at installing it in an internal bay. Most, but not all, IDE hard disks are 3.5in devices and so this shouldn't be a problem, but if you opt for a Quantum Bigfoot drive it will have to go in a 5.25in bay, which will probably mean losing an external drive bay.
2. Make sure you have the right mounting hardware, too; things like bolts or rails. Mount the drive rails if needed. If your new drive came with its own mounting screws, use them. Avoid using just any old bolts: if they're too long, they can damage the drive's electronics.
3. Power down the PC. Install the new drive in its bay — slide it in and tighten the mounting bolts.
4. Attach the power cable and the 40-way

p214 ➢



Left The 40-way ribbon cable and power lead connects your motherboard to the hard disk, which cannot be used until it has been partitioned (below) using FDISK, a DOS utility



ribbon cable to the new drive and to the primary IDE channel on the motherboard.

Some ribbon cable connectors have a "polarised" shroud, with a little notch which prevents the cable from being inserted incorrectly. If yours isn't, again make sure that the coloured stripe on the ribbon cable goes to Pin 1 of the connector on the motherboard and the hard disk. The power lead is always notched to ensure correct insertion.

5. Power up the PC. The new hard disk hasn't been "initialised" yet and thus is incapable of booting. So at this point press the DEL key (or whatever) to access your CMOS Setup program.

6. You now have to tell the CMOS the "geometry" of the new drive: that is, the number of heads, cylinders and sectors per track, of the new drive. You can either do this manually, keying the figures into the "User Definable" section, but it's much easier to select the option that auto-identifies the new drive. Save the new CMOS settings and quit Setup.

7. Restart the PC, this time with the system floppy in the drive. Now, even though your PC can recognise your new disk, you still can't actually use it until you perform two additional, related operations: partitioning and formatting (see also "Big drive partitioning", above).

Partitioning is carried out using the venerable DOS utility, FDISK:

- Boot from the system floppy.
- Run FDISK and select option 1 from the menu "Create DOS Partition".
- Select option 1 from the second menu "Create Primary DOS partition".

For reasons of storage efficiency avoid the temptation to make a single humungous partition occupying the entire hard disk. Make it no bigger than 511Mb: this will give you a cluster size of eight 512-byte sectors (or 4Kb), which is tolerable.

By contrast, if you create a single partition on a 2.1Gb disk you'll wind up with 32Kb clusters, which is a profligate waste of disk space.

Sadly, while the OSR2 version of Windows 95 supports FAT32, which overcomes this slack space problem, it isn't commercially available. The retail version of Windows 95 still supports the inferior FAT16 file system and so this is the regime to which we must adhere.

8. Press ESC a couple of times to back out of FDISK and reboot the PC using the system floppy.

9. You can now format the new primary partition on the hard disk. Type:

```
FORMAT C:/S <CR>
```

Not only will this format the drive, making it usable, but the /S switch copies the system tracks across. Depending on the size of the drive this could take a couple of minutes.

10. The hard disk should now be bootable so you no longer need the system floppy to boot with: test it by hitting the Reset button.

And that's enough drives for this month. In the next part of the series, I'll perform the topping-out ceremony of this building project: installing the CD-ROM drive, installing the operating system, and tweaking the system to make sure we get the most bangs for our buck.

Big drive partitioning

If you've bought a really big drive (<2Gb) you should be aware that the largest partition possible under FAT16 is 2.1Gb or 2,146,959,360 bytes. Hence, drives larger than this have to be partitioned into smaller logical drives using FDISK.

- Assuming you've created a 511Mb primary partition, load FDISK again and this time select option 2 from the second menu, Create Extended Partition.
- You can then devote the remaining space on the drive to the "Extended Partition".
- From this you can create "logical" drives.

For example: say you had a 2Gb hard disk with a 511Mb primary partition; you could create a 1.5Mb extended partition and from this carve out three 511Mb logical drives.

Back Issues

See page 12, or this month's *PCW* CD-ROM, for how to obtain Parts I, II and III of the Workshop, which appeared in our June, July and August '97 issues. The final part (V) appears next month (October issue).

PCW Contact

Roger Gann can be contacted by post c/o *PCW* at the usual address or via email at hardware@pcw.co.uk.

Cable junction

In the third part of his Workshop series on building your own PC, Roger Gann explains how to install the motherboard, how to connect the cables, and how to check that all is well so far.

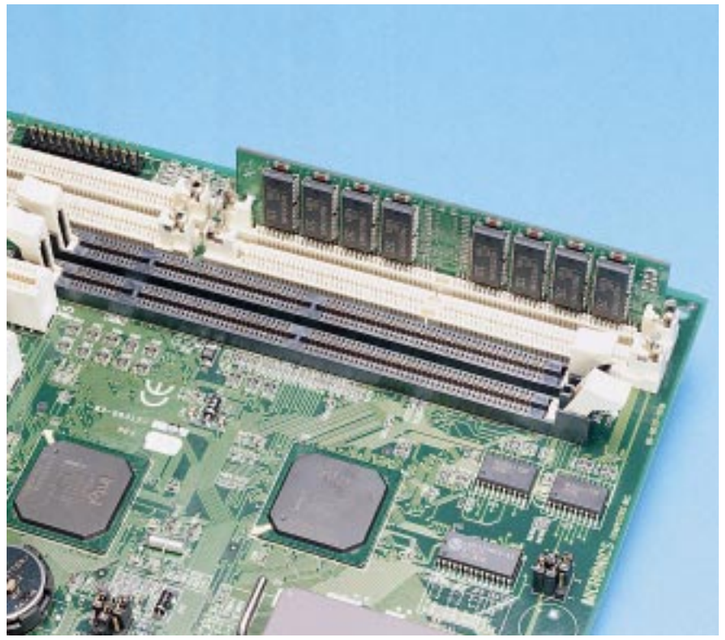
Last month I looked at the pros and cons of the build-it-yourself PC and led you through the process of choosing the motherboard and the system unit, and specifying all the components needed to build your PC. I chose a games PC for the reason that it represents the most powerful PC available. I then began preparing the motherboard.

This time I'm going to describe the final installation of the motherboard, and begin installing the expansion cards.

Install the new motherboard

1. Give the new motherboard a final once-over to make sure you've correctly set all the jumpers. Check that the SIMMs are correctly mounted in their slots and that the CPU has been orientated properly.
2. If you have a "cache-on-a-stick" module, make sure it's plugged into its socket. If you have a mini-tower system case, lay it on its side so the area where the motherboard will go is at the bottom.
3. Slide the new motherboard into position in the base of the case so the stand-offs engage in the corresponding locating holes in the floor: the position of the keyboard socket (and the hole in the casing for it) will help you locate the motherboard correctly.
4. Slide the motherboard as far to the right as possible (this may entail a little wiggling).
5. Make sure all the stand-offs have engaged in the base. Your system case may have several threaded nuts on the base — align the remaining holes in the motherboard with these and fit the supplied bolts to secure the motherboard in place.

Sometimes the system case may lack mounting holes that correspond with holes in the motherboard. This can be a problem, especially at the front on the motherboard which may be unsupported as a result and may flex alarmingly when inserting cards or processors. The solution is to install stand-offs in the appropriate holes but to cut off the lugs that fit the holes in the system case, so the stand-offs merely act as legs.



Make sure you check all your motherboard components thoroughly before mounting it in its case

Cabling

Having fixed the motherboard in place in the system case, our next task is to connect the myriad cables. These are the power cables, the cables which drive the LED displays, the loudspeaker and the reset button.

■ First in line are the two power cables, the two chunky female plugs with multi-colour cables attached. They will be labelled P8 and P9. By the rear right-hand corner of the motherboard you'll find the male socket for these plugs — it's probably white or cream and about 50mm long.

The plugs are identical, and while it's impossible to fit them facing the wrong way, it's certainly possible to put the one that belongs on the left, on the right, and vice-versa. Luckily, there's an easy way to orientate them correctly; fit them so that the black cables on each plug go together.

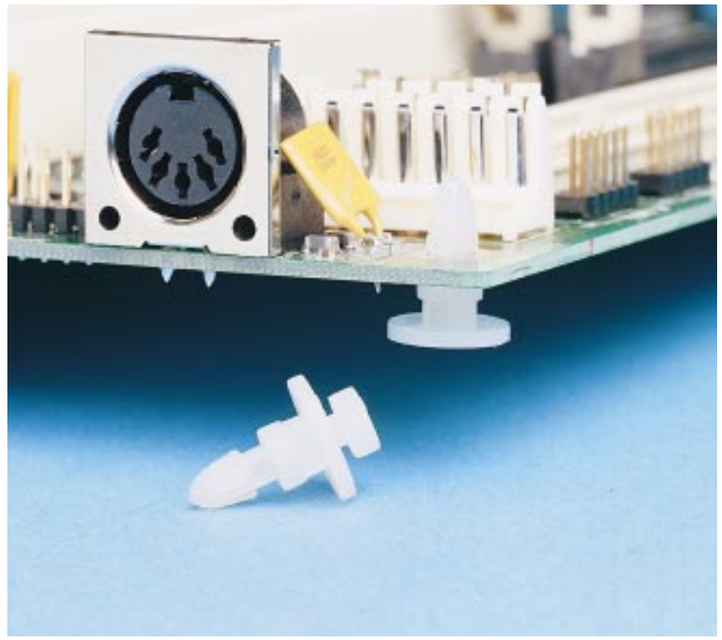
■ If you have an ATX motherboard, its power connector is slightly different: it's a single 20-pin socket and you'll need an ATX-style case that has a PSU (Power Supply Unit) with the appropriate connector.

■ The most fiddly task of all is fitting the little multicoloured cables. Typically, these are positioned along the front edge of the motherboard. Your case will have quite a few of these and will include connectors for these cables: keylock, reset, power, turbo LED, and switch and speaker.

There's no earthly reason why the number of leads and plugs attached to the case should match the number of connectors on the motherboard, so occasionally you'll have to use your noddle when ne'er the twain shall meet. For example, the leads can be grouped together to form just two or three plugs. If this is a problem because the motherboard connectors are not so arranged, it's OK to split the plastic plug into two separate plugs with a sharp knife. Pay close attention to the schematics provided in the motherboard manual and make sure you get the orientation of the plugs right. Often the connectors are tightly crammed together and badly labelled.

■ If you're fitting a 486 processor, you should now fit the heatsink which just clips on over the processor. If you are fitting a DX4 or a Pentium, you will need a combination cooling fan and heatsink — and this will need to pick up power from somewhere. Modern motherboards often have a power connector specifically for the fan, located adjacent to the ZIF socket, while others may have a fly lead which will take its power from one of the peripheral power leads.

Your PSU may make no special provision for the cooling fan, so you may have to buy a special lead and plug it in to one of the normal power leads. Buy a "Y" power lead adapter to avoid sacrificing a precious power lead.



If the system case lacks mounting holes for your motherboard, slice the standoffs in half so that they sit flat on the base

I/O ports

All modern motherboards now feature integrated I/O (input/output). Rather than use dedicated I/O cards which hog expansion slots like a serial/parallel/floppy/IDE card, these ports are now built into the motherboard. So the next step is to

p242 ➢

Check that the CPU is correctly mounted in its ZIF socket

commission your new PC's basic I/O.

■ Because expansion cards are no longer needed, the various sockets for these I/O ports have to be separately mounted on the chassis. You should find a set of these ports (a parallel and a pair of serial ports complete with grey ribbon cable and sockets) included with the motherboard.

■ Sometimes these are mounted on an expansion slot blanking plate, which makes installation a cinch — you just substitute a spare blanking plate for this one. Try to use the blanking plate closest to the power supply to avoid losing the use of a precious PCI. Make sure the lower socket doesn't foul up anything on the motherboard (they tend to be located low down on the blanking plate).

■ You may have to use cut-outs in the rear panel of the system case to mount your ports. These are held in place by a bit of solder and a light tap with an old screwdriver will dislodge them. Undo the bolts on the sockets, offer up the socket from inside the case and tighten up the bolts. They can be a fiddle to tighten because they're mounted so close to the socket itself. Install the parallel two serial ports in this way.

■ Complete the job by plugging the ribbon cable attached to each port into the appropriate header connector on the motherboard. Pin 1 should be marked on the motherboard. Align the coloured edge of the ribbon cable with Pin 1. Better motherboards use shrouded connectors with a polarising notch which prevents you from inserting the motherboard plug incorrectly. (Note that some motherboards don't have a joystick port, relying on the fact that most sound cards come with one.)

■ Another connector you might come across is the PS/2 mouse connector. There should be a motherboard connector close to the keyboard socket and the mouse port is almost certainly supplied on a blanking plate. Check the Pin 1 alignment as above when plugging in the data cable.

Fit the peripherals

That completes the cabling phase and we can start fitting the peripherals at long last. But we won't get carried away; we'll start with the graphics card. The reason for this is that I want to do a little preliminary system checking and make sure that what I've done so far is OK before proceeding any further. So, unpack your graphics card, again remembering to discharge any static electricity before touching any electronic components on it.

1. The card will probably be a PCI card, so fit it in a free PCI slot. Undo the bolt securing the blanking plate at the end of the slot and remove the plate.
2. Hold the card firmly by its top edge and press its edge connector firmly into the expansion slot (it may be a tight fit and you may have to use a modicum of steady force). Tighten up the bolt to stop the card from flapping around.
3. Now, without putting the cover back on (but taking the usual safety precautions), plug in the keyboard and monitor to their respective ports at the back of the system unit. Plug in the mains lead to the system unit and hit the on/off button.

Is it happening?

With luck, things should start to happen: the power LED should light, the CPU cooling fan should spin and the floppy disk drive should make a little grinding noise ("perform a seek" in technical jargon). The motherboard's BIOS should now perform its POST (Power On Self Test). It will count through the installed memory as it checks it, so watch this.

There's no boot device so your new PC cannot boot yet, but you can enter CMOS setup (typically, by pressing the DEL key at boot time) and set useful things like the time and date. This also confirms that the basic PC is a working machine.

If nothing happens, check the mains lead, the two mains plugs and the miniature multicoloured cables. Also check that the graphics card is plugged in and that the monitor is connected to it. Check the CPU in its ZIF socket. If the LEDs light up, but nothing happens on-screen and instead you hear a series of beeps, you have a fundamental error condition: listen to the pattern of the beeps and then consult the motherboard manual to see what error condition they indicate. Often it is a problem with main memory or with the graphics card, so try re-seating them in case it is a poor contact.

Next month, we will fit the remaining peripherals.

Back Issues

See p12, or this month's *PCW* CD-ROM, for how to obtain Parts I and II of the Workshop, which appeared in the June and July '97 issues.

PCW Contact

Roger Gann can be contacted by post c/o *PCW* at the usual address or via email at hardware@pcw.co.uk.

The games that PCs play

Why not build your own PC? It's easier than you imagine. Here, in Part II, Roger Gann considers a high-power spec, prepares the motherboard, and shows how to fit the SIMM and stand-offs.

Last month we looked at the pros and cons of building your own PC and how to go about selecting those most anonymous of components, the system case and the motherboard. This time around I'll deal with specifying the ultimate games PC, from the ground up.

When I tell you that a top games PC is probably the most powerful, this side of a dedicated graphics workstation, it gives you some idea of what is possible when you have the luxury of being able to build your own PC. But if you don't actually need that much power, you'll be able to scale down the specification accordingly. I'll also be covering the first stage of assembly — the preparation of the motherboard.

You might think that the most powerful PCs are invariably to be found on business desktops? Wrong. The most powerful PCs are those dedicated to playing games, and these are found in the home. If you're serious about games-playing on a PC, then you'll need a serious PC on which to play them. And I'm talking about *really* serious, modern games, which incorporate all manner of multimedia effects: sound, animation, video and 3D rendering; these test the capabilities of a PC like few other programs can. It may sound preposterous, but Intel is aiming the latest and fastest 200MHz MMX Pentiums at the home market simply because games run better on the fastest processor you can buy.

The processor

So, the good news is that as far as games are concerned, you now have two choices. Your first port of call is the 200MHz Pentium MMX. The Pentium Pro is undoubtedly a more powerful processor, but it's not so hot when it comes to running 16-bit apps and doesn't have MMX support (although the Pentium II will), so we'll shortlist the 200MHz Pentium MMX. Although not many games support MMX at present, the list is growing, and it's a must-have for the serious games and multimedia user.

The other hot candidate for fast game play is the recently announced AMD K6 processor. This is faster than the Intel Pentium and in some cases is faster than the Pentium Pro. It's also about 25 percent cheaper than the equivalent Pentium and available in a 233MHz version too, making it both cheap *and* fast. Add to this the fact that there's no performance penalty when running 16-bit apps, that it supports MMX and that it is Socket 7 compatible, and the K6 begins to look very interesting.

Because the K6 is so new, make sure the motherboard you buy supports this advanced CPU. (*See last month's column, which dealt with selecting a motherboard*.)

Storage: hard disks & CD-ROM drives

Although the most powerful games will run almost exclusively from CD-ROM, the hard disk still has an important role to play, so you should ensure that you've got the fastest-possible hard disk subsystem. The vast majority of new PCs these days come equipped with Enhanced IDE hard disks and, thanks to the modern miracle of Mode 4 PIO, can belt out data at a cracking pace of up to 16Mb/sec. Very soon, though, motherboards will appear which support Ultra DMA EIDE, delivering 33Mb/sec throughput.

OK, so the EIDE transfer rate is good; but there's a price to pay, and that is extremely high CPU utilisation rates. But thanks to PIO, the CPU alone has to supervise every chunk of data that is hoovered up off the disk and this takes up plenty of its time; time which could be better spent actually running your game.

Games purists should avoid Enhanced IDE hard disks altogether, good though they are, and instead plump for SCSI because it is clever and handles all data transfers itself, thus not wasting processor power.

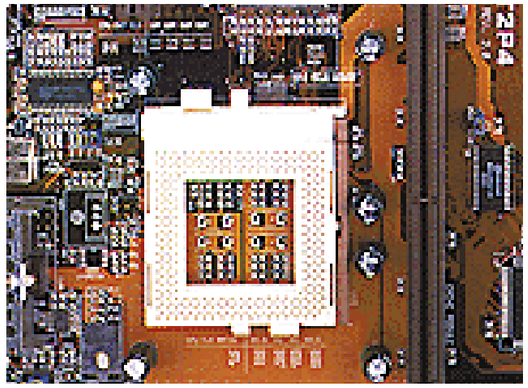
Here, I'd recommend something like the Adaptec AHA-2920 or 2940 host adaptor cards coupled to a Seagate Hawk Fast SCSI-2 drive. Or, if you're very keen, go for Ultra Wide SCSI such as the AHA-2940UW plus an Ultra Wide SCSI drive. This combination can deliver phenomenal throughput in the region of 20Mb/sec to 40Mb/sec.

For the same reasons, you'd have to choose a SCSI CD-ROM as well. Even though most modern CD-based games are mastered on four-speed drives, it is a good idea to get a faster drive. Right now, the fastest CD-ROM drives are 12- and 16-speed models. Unfortunately, these are mostly IDE/ATAPI drives which make severe demands on the CPU to deliver this kind of performance: SCSI CD-ROM drives don't, so I'd recommend these above their ATAPI counterparts any day. Sadly, they're also considerably more expensive, but I would bite the bullet and opt for something like the new Plextor 12/20Plex drive which can deliver data at 3,000Kb/sec.

Graphics cards

Without doubt, if you want the most realistic-looking games you'll need a graphics accelerator which supports 3D graphics. Unfortunately, the development

Step by Step — Motherboard, SIMM and stand-offs



Preparing the motherboard

1. First, get your toolkit together. You'll need:
- a Phillips screwdriver;
- an electrician's screwdriver; and
- a pair of fine needle-nose pliers.

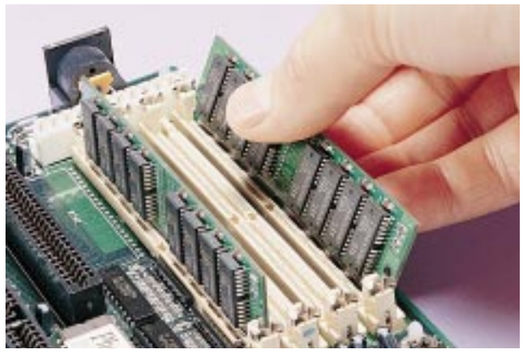
2. Take a moment to examine your new motherboard and read through its (no doubt sparse) manual. Check whether there's anything important of which you should be aware.

3. Most motherboard manuals are invariably terse and techie but you should try to identify the positions of important components and jumpers.

4. If the processor is supplied loose, fit it into the ZIF socket on the motherboard:
- lift the socket lever clamp;
- orientate the CPU so that the bevelled corner on the processor (a.k.a. Pin 1) aligns with Pin 1 on the ZIF socket; and
- lower the CPU in, then lower the lever to clamp the chip into its socket.

Most modern motherboards support a variety of processors from Intel, AMD and Cyrix, and you normally have to move a fair number of jumpers to configure the motherboard for the particular type of processor you're using.

You'll also have to configure the motherboard for the clock speed of the CPU. Many Pentium motherboards are festooned with these tiny black jumpers which are often poorly laid out (from a point of view of ease of use). Mercifully, though, jumpers are on the way out and the latest motherboards are entirely software-configurable from the CMOS Setup menu.



Fitting a SIMM

While access to the motherboard is so easy, take the opportunity to fit the SIMM memory. You will probably have planned to buy EDO SIMMs but if both your motherboard chipset and your budget support it, consider buying Synchronous DRAM (SDRAM) instead — it's a tad faster than EDO.

1. SIMMs are notched at one end to prevent them being inserted incorrectly in their sockets. Find the notched end and locate the

p256 ➢

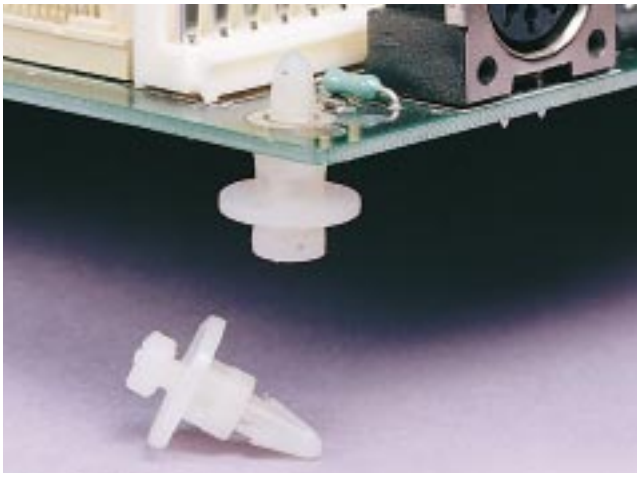
corresponding key in the SIMM socket.
2. Insert the SIMM module, at a shallow angle, into the first SIMM socket (they'll be numbered), then gradually rotate it until it's vertical and the side clips have snapped into place.
3. Do the same for the second SIMM (they have to be fitted in pairs).

Fit the stand-offs

Our final task for this month is to fit the plastic stand-offs to the motherboard. These plastic legs both secure the motherboard to the system unit and insulate it.

• The motherboard will have a number of holes through which the plastic stand-offs are pushed. These stand-offs then locate in tapered "key-holes" in the floor of the system unit. The problem here is that the two sets of holes seldom match up exactly and there will probably be more holes in the system unit base than there are on the motherboard itself.

• At this point you have to carefully work out which holes in the motherboard match up with the corresponding holes in the base of the system unit and *only* fit stand-offs in these holes.



■ **Next month:** We'll actually fit the motherboard and the rest of the peripherals.

Motherboard check list

Your minimum motherboard specification should look something like this:

- At least a 166MHz Pentium MMX (or equivalent) CPU.
- At least four SIMM slots. Maybe one DIMM slot.
- At least 256Kb of pipeline burst-mode secondary cache memory.
- A Triton 430HX or 430VX chipset.
- At least three PCI and four ISA slots.
- On-board I/O (e.g. EIDE, floppy, serial and parallel ports).
- A PnP BIOS of reputable brand such as AMI, Award, MR or Phoenix.

You should be able to get a motherboard to this specification for about £275 (ex VAT). Don't forget the memory: at the time of writing, 16Mb SIMMs cost about £55 and you'll need a pair of them, making £110 (ex VAT).

and take-up of 3D games has been hindered by Microsoft's dilly-dallying over Windows 95 graphics standards. Ordinary DOS games, which are always looking for maximum performance, access the graphics hardware directly; something that was *verboten* under Windows 3.1x. As a result, most DOS games couldn't run under Windows, or if they did, ran so slowly as to make them unplayable.

That settles it — it's DirectX

After much to-ing and fro-ing, Microsoft finally settled on DirectX, a video standard which permitted games to run under Windows 95. This standard embraces DirectDraw, DirectVideo, DirectSound and Direct3D, among others. All are supposed to simplify and speed operating-system access to hardware devices by providing direct access with as little driver overhead as possible.

Most 3D (and 2D) cards now ship with DirectX drivers but it's an emerging technology and you should check Microsoft's web site, which can be found at www.microsoft.com, to see whether updates are available.

DirectX and especially Direct3D are important because until recently games were, in the absence of a common 3D standard, specific to a particular graphics accelerator. Once Direct3D becomes ubiquitous, you'll be able to play any Direct3D game on any 3D graphics card.

Games developers will no longer have to account for what 3D acceleration hardware you might possess, and 3D acceleration hardware vendors will not have to worry about what games you have. So you should check for Direct 3D support when making your choice of graphics accelerator.

3D hardware

So what 3D hardware is available? The most popular 3D accelerators are based around S3's ViRGE and ViRGE/VX chipsets; cards like the entry-level Diamond Stealth 3D. As well as offering 3D rendering, they can all be hooked up to S3's Scenic/MX2 hardware MPEG decoder and, potentially, to other multimedia components via S3's Scenic Highway local-peripheral bus.

Because 3D cards deal with a third dimension, they require far more memory than a 2D card and you should be thinking about 4Mb or maybe even 8Mb of display memory for these.

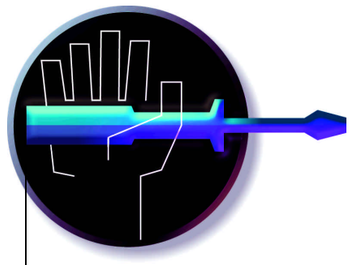
Then there are the dedicated 3D processors, like the VideoLogic Apocalypse 3D or the Diamond Multimedia Monster cards. These work in conjunction with existing 2D cards to deliver high-quality 3D graphics. They add realism to 3D objects by using transparency effects, lifelike shadows, shading, fogging and search-lighting. These cards are fast, too, because they perform the complex 3D calculations on-card and don't bog-down the CPU. Priced at less than £150, these 3D add-ons are worthy of an appearance on your shortlist.

Back Issues

See page 12 or this month's *PCW* CD-ROM for details of how to obtain Part I of this Workshop, which appeared in the June 1997 issue.

PCW Contact

Roger Gann can be contacted by post c/o *PCW* at the usual address or via email at rgann@mcgilivray.win-uk.net.



The home-brewed PC

Why not try building your own PC? It's fun, and you can learn a lot about what makes computers tick while you're tinkering. Roger Gann starts off with a list of the bits and pieces you'll need.

In my Hands On Hardware column over the past year or so, I've looked at the many ways you can upgrade your PC's hardware, from adding a SCSI host adaptor to replacing your motherboard. Over the next four months I'll be taking you through a much larger upgrade project: building your own PC from scratch. I'll take you through the A to Z of building a decent quality entry-level Pentium PC, complete with an Enhanced IDE hard disk and a CD-ROM drive.

Let's face it, assembling your own PC isn't a popular pastime, and doesn't begin to rival gardening or fishing as a hobby. But building a DIY PC can be a lot of fun and it's very instructive: at the end of it you'll have a much clearer understanding of PC internals and how they work together. You'll also be better placed to troubleshoot hardware problems in future. By building it yourself, you can opt for the piecemeal approach, spreading the purchase of the components over a period of time. You'll have the benefit of a PC built to your exact specification and to your standards of workmanship.

That's the good news. There's bad news, too. For a start, most PC assemblers already build PCs to your precise specification. More importantly, you're unlikely to beat them on price by opting for the DIY PC. You'll be buying components individually, while PC manufacturers will buy *en masse* so their unit costs will be much lower. Factor in the time spent building it, add up the cost of bundled software so often included with PCs sold today, and you'll see that, overall, it will be cheaper to buy a complete PC. So there are no savings to be had from building it yourself: but we're not doing it for the money, are we? Then there's the warranty. Escom owners may have a view on the value of service warranties, but any warranty is better than no warranty; and when you brew your own PC, you're on your own.

Degrees of difficulty

To be honest, building a PC isn't for everyone, and if you're the technically timid sort that finds fitting a graphics card an ordeal, you should stop reading here and quickly turn to the next feature, as building a PC from scratch would be a big mistake.

However, if you've attempted any of the more adventurous upgrades I've covered over the months, such as swapping a motherboard or fitting a hard disk, building a PC really isn't substantially any more complex than this. Rocket science isn't involved: there are just more bits to fit and a tad more preparation and planning. Once you've started, will it take ages to complete? No, is the short answer. With all the bits in front of you, a simple PC can be assembled in less than an hour.

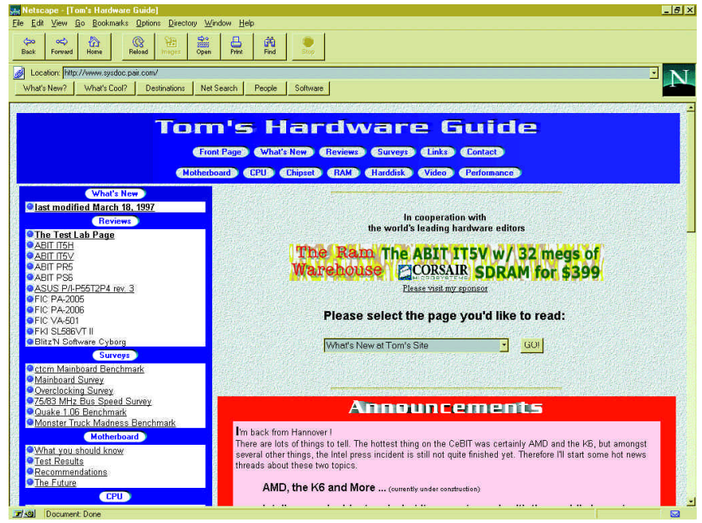
Sadly, there seems to be little in the way of books on the subject. Incredible as it may seem, until recently there were no books at all in the Computer Manuals catalogue on this subject. Now there's one, called, funnily enough, *Build Your Own PC*. But no fear: over the next four months I'll be giving you the low-down on the whole process, from start to finish.

Choosing components

Choosing some of the most important components of your home-brewed PC will be particularly tough. OK, you'll have the reviews at the front of *PCW* to guide you when you come to choose hard disks, graphics accelerators, monitors and the like. But you'll be on your own when it comes to such things as motherboards and cases, as these are invariably ignored when it comes to product reviews in any computer magazine. I guess cases are just too dull and motherboards too anonymous and unbranded to bother with.

Assuming you've got no spare hardware lying around, you'll need the following:

- 3.5in floppy drive (£15)
- 2Gb hard disk (EIDE) (£165)
- Eight-speed CD-ROM drive (£65)
- 72-pin EDO or SDRAM SIMM memory (16Mb) (£50)
- PCI graphics accelerator, e.g. Matrox Mystique or VideoLogic GrafixStar 600 (£100/£80)
- 15in 0.28mm dot-pitch SVGA display (£250)
- 102-key AT Windows 95 keyboard (£25)
- Mouse (£20)
- System case with power supply (£50)
- Motherboard and P166 CPU (£300)
- Any software, i.e. Windows 95



Check out Tom's Hardware Guide for some seriously detailed hardware info

Choosing the system case

Between them, the case and the motherboard amount to the foundations of your PC, so it pays to thoroughly check out what's available. These parts may look as alike as peas in a pod in the ads, but believe me, they aren't. With system cases, it's important to actually see the case and open it up. This way, you can judge for yourself just how easy it is to use and whether it meets your needs. If possible, don't buy "blind" (off-the-page); buy in person. Case ergonomics should play a big part in your choice but you'll probably only discover its shortcomings after you've bought it. An example of this is my PC's tower case. I had to remove the entire motherboard just to be able to undo a pair of bolts in order to swap

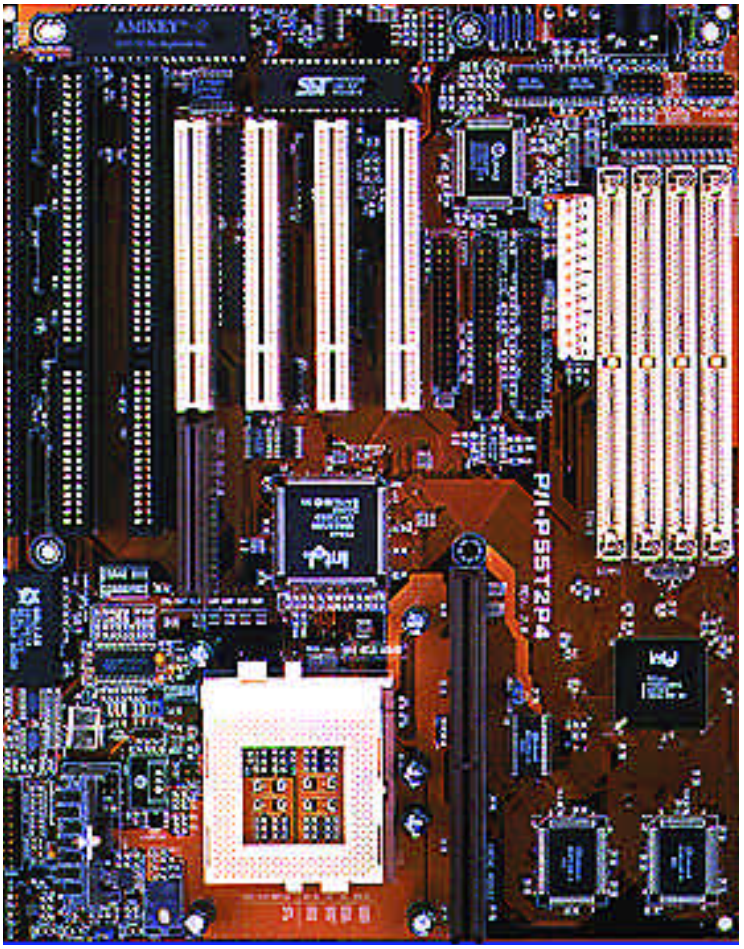
p240 ➢

a sickly hard-disk drive.

Flicking through the ads reveals that a good selection of system cases are available. However, they're all much of a muchness and fall into four broad types: normal desktop, slimline desktop, compact desktop/mini tower, and full tower. Prices start at about £35 and top out at about £85 for a full tower, although if you want a case that complies with the Euro CE safety standard you can add roughly a tenner to these prices. Wherever possible, try and get the biggest system unit possible. Not only will this give you maximum expansion potential, but it also makes access to the internal components easier. Unless you specifically want a slimline case, go for the larger case. In the end, the final decision boils down to expansion potential: if you want to fit a lot of drives, buy a tower case. If not, buy a desktop or mini-tower case. Don't forget, you get what you pay for: pay peanuts for a case, and you'll get something of flimsy construction and awkward to use.

All "Baby AT" motherboards will fit standard cases, but watch that the slimline case doesn't require an expansion card riser or "tree" so that cards can be fitted horizontally. If it does, be sure to select a motherboard that has these features and, most importantly, fits the case. And if you're fitting an ATX-style motherboard, make sure you buy a case designed to take ATX form-factor boards.

Most cases come with a 150W or 200W power-supply unit (PSU) as standard, but this might be a bit light for a well-stocked full tower. Ask how many power connectors the PSU has (the more the merrier), and what sort they are. It should have two types: the standard Molex, and the mini power connector. Most PSUs have power leads for only four peripherals, but try and find one with six. Ask whether it comes complete with all the fixings and accessories, things like printed circuit-board (PCB) supports, mounting bolts and drive rails. Consider at this point whether you want to fit a removable hard-disk tray.



The Asus P/I-P55T2P4 motherboard recommended in Tom's Hardware Guide pushes the new Intel 430HX chipset to its full potential

Choosing the right motherboard

If choosing a simple thing like a system case isn't straightforward, choosing a motherboard to go inside it is tougher still. The motherboard is, of course, the heart of your PC and, thus, is a fairly technical piece of kit. They are mainly sold as virtually unbranded, generic devices, each one near enough identical to its neighbour. There are such things as motherboard "best buys", but in the absence of proper product reviews who's to know? Sadly, there's no comfort to be derived from relying upon brand names to guide you. With the exception of Intel, you probably won't have heard of the major motherboard players: people like Asus, Abit, ECS/EliteGroup, Gigabyte, Micronics and SuperMicro.

So what should you be looking for in the ideal Pentium motherboard? Well, there's a veritable laundry list of desirable features that should appear on your checklist. There are the obvious ones like the form factor (Baby AT or ATX), the number of PCI and ISA slots, and the nature of the on-board I/O it has (EIDE, fast serial and enhanced parallel ports). There are other less obvious but just as important features. These include having a Flash BIOS (which permits software upgrading), the number of SIMM slots (usually four but sometimes eight), and is there a DIMM slot? Does it have an IR port or support for Universal Serial Bus?

On the techie side, you should check the board supports a wide variety of processors, including Cyrix and AMD CPUs. It should have an adjustable CPU voltage regulator (Standard/VRE/MMX), support EDO and SDRAM (particularly the latter) and should have a modern, up-to-date chipset. If it's an Intel chipset, it ought to be a Triton 430VX, HX, or the just-released TX.

So which motherboard is best? Luckily, there is an excellent web site that conducts benchmarking tests on motherboards which you can refer to. Tom's Hardware Guide (www.sysdoc.pair.com) contains an absolute goldmine of technical info plus hints and tips about PC hardware, and is well worth a visit. There you'll find various motherboard "Top Tens". For example, for 430HX boards, Tom Pabst recommends the Asus P/I-P55T2P4 and Abit IT5H boards, and for Triton 430VX boards, the Abit IT5V. Boards like these not only offer jumperless "Soft Menu" configuration but can also run the bus at 75MHz or even 83MHz (as opposed to 66MHz) for the latest generation of fast CPUs. It's worth searching Yahoo on the keyword "motherboards": you'll find all the motherboard manufacturers with an internet presence.

And that's all for this month. In part two, I'll be looking at what you'll need if you want to build the ultimate games platform, plus the first step, installing the motherboard. ■

Getting automated

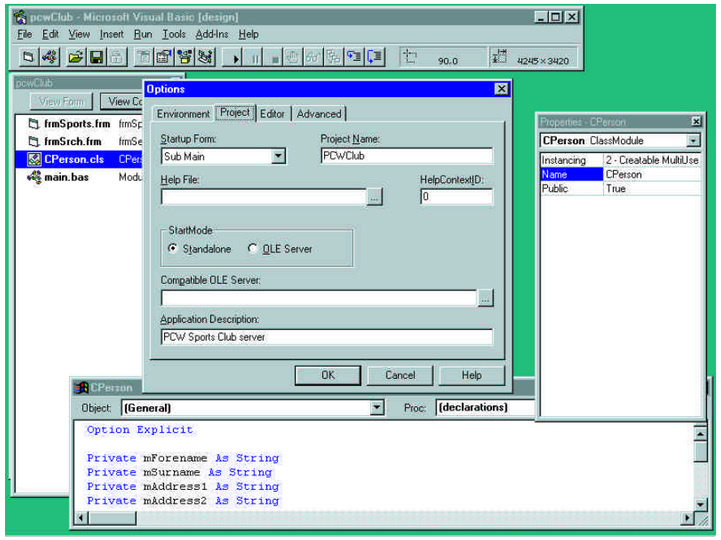
Tim Anderson does clever stuff with automation servers in the final part of the workshop. Plus, polishing the Sports Club database and adding some essential new functions.

Last month's workshop demonstrated how a Visual Basic class can be plucked out of the standalone version of VB and, with care, planted into Microsoft Office as a Visual Basic for Applications class. That is one way of re-using code, but an even better approach is to create objects that can communicate with any number of different applications without the need to recompile. With the rise of PC networks, and now the internet, this kind of software is the way of the future.

Under Windows, the way to achieve this is by using Microsoft's Component Object Model, or COM. This is the technology beneath OLE and ActiveX, and Visual Basic programmers can use it without knowing the detail of how it all works. For example, what if the PCW Sports Club wants to get at membership details not only via Word, but also from other programs like accounts and desktop publishing packages?

A key part of the Sports Club application is the CPerson class module which describes a club member. By making a few changes, that class module can become an automation object which exposes its properties and methods to other applications. The steps are as follows:

1. Open the CPerson module and press F4 to reveal the class properties. Set the Instancing property to CreateTable MultiUse, and the Public property to True.
2. Add a module to the project using Insert - Module. In the module, create a Sub Main.
3. On the Tools menu, in Options - Project, change the project name to PCWClub, and put a few words of description in the Application Description field. Finally, change the Startup Form to Sub Main.



Two key steps in creating an automation server are: first, the properties for the class to be exposed; and second, the project options

The project name must be unique to your automation server. The full class name of objects in your server will be of the form PCWClub.MyClass.

By taking these simple steps, you have created an automation server that lets other applications create and control objects of the CPerson class. All that remains is to register the class in the system registry. If you run the application in VB's development environment, it will be registered temporarily. If you build an .EXE or OLE DLL, it will be registered permanently. Then you can write code like this in Excel:

```
Dim myobj As Object
Set myobj = CreateObject
("PCWClub.CPerson")

  If myobj.FindPerson("Bloggs",
"Fred") Then
MsgBox "Fred lives at: " &
myobj.Address1
End If
```

Although applications such as Excel function both as automation servers and standalone, most VB applications will be one or the other. Often, VB automation servers have no user interface, since this is provided by the client application. The workshop example, though, is designed to work in both guises. The trick is to use Sub Main to detect whether the application is running standalone or as an automation server. Here is the code:

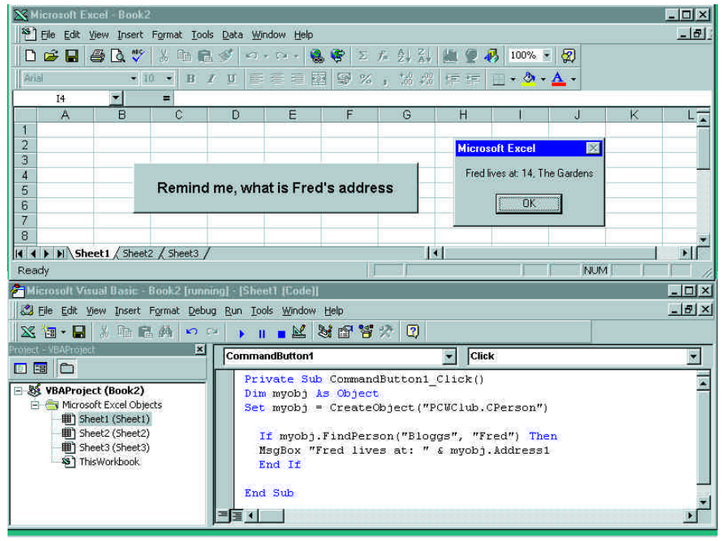
```
Sub Main()

If App.StartMode <>
vbSModeAutomation Then
frmSearch.Show
End If

End Sub
```

You may wonder what Sub Main does when running as a server? The answer is, nothing at all. The only thing the server application does is to expose its classes so other applications can create and control objects. For testing, you can simulate this mode by setting the startmode in Project Options to OLE Server. Run the application and then minimise VB. Next, run another instance of VB, open the References dialog and check the PCW Sports Club server. Now you can test the server by creating objects of the CPerson class.

If you have run a compiled VB automation server such as PCWCLUB.EXE, it will be entered permanently in the system registry. Once you have finished testing, it is good practice to remove it. You can do so by running it from a command line with the /UNREGSERVER parameter. DLLs are unregistered using REGSVR32.EXE which is found in the System directory. Run it without parameters to see the switches.



Excel is able to get at Sports Club details by using a VB automation server

Automation servers are powerful but do present some new programming challenges. The section of the Visual Basic manual called "Creating OLE Servers" is essential reading.

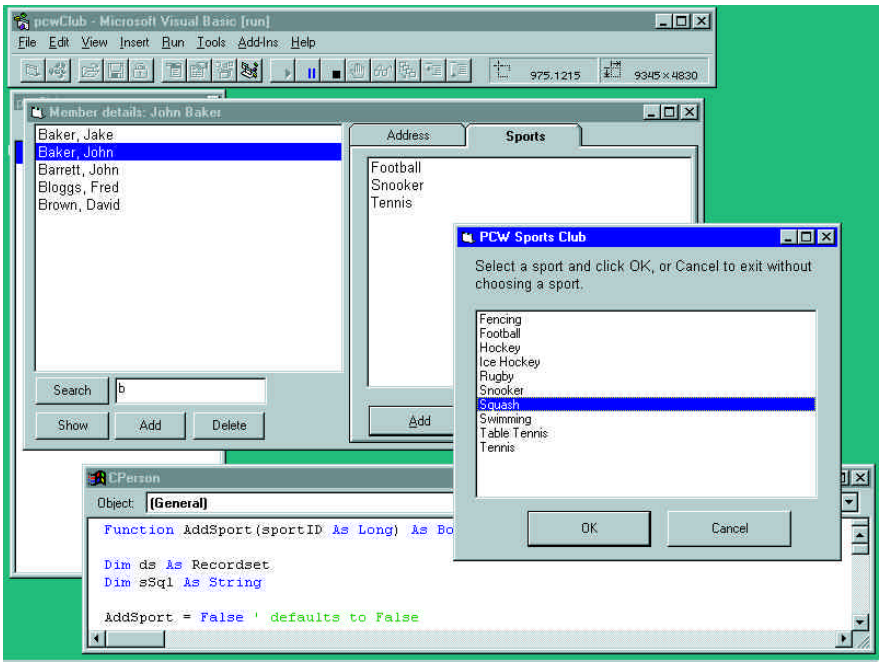
Adding the essentials

The Sports Club database is also used as a standalone application, and the version in last month's workshop is lacking some essential features. First, there is no way to add or delete members; and second, you cannot add or remove sports from the list which applies to each member.

The thinking behind the design of this simple application is that interface code belongs in the main form, while database code belongs in the CPerson class so that it can be used in other applications or as an automation server. The natural approach is to create new public methods for CPerson that give this new functionality. For example, here is code to add a new member:

```
Function CreateNew(sSurname As
String) As Long
' creates a new person in the
database
Dim lId As Long
myRecordSet.AddNew
myRecordSet!surname = sSurname
lId = myRecordSet!ID
myRecordSet.Update
Me.load (lId)
End Function
```

p260



Adding a sport to a member's list of interests is achieved via a simple dialog

The ID field is a counter, which means that the JET database handles the business of ensuring that the new member has a unique number. There is an issue, though, about how to cope with users who change their mind.

What if someone starts to create a new member, and then wants to back out and leave things as they were? One possibility is to call the AddNew method, but not to call Update until the user confirms the action. Unfortunately, bullet-proofing the application so that Update is only called after AddNew or Edit is prone to error. The easier approach is to minimise the time when JET has unsaved changes in its copy buffer. In this application, clicking the Add button creates a new member with the surname "Unnamed". If the user wants to cancel the addition, it is just a matter of clicking Delete.

The DeletePerson method is a little more involved. The problem is that there may be other records, in the SPORTLINK table, which refer to the member being deleted. To maintain data integrity, these records also need to be removed. Database objects have an Execute method which is an ideal solution. Execute takes an SQL command and applies it to the database. For example:

```
sSql = "DELETE * from SPORTLINK
where SPORTLINK.MEMBERID = " &
Str$(lId)
myDB.Execute sSql, dbFailOnError
```

The code at form level also has some work to do. When a member is deleted, the name must be removed from the list currently displayed, and the other fields on the form updated as necessary.

To make sense of adding sports to a member's list of interests, you need to throw a dialog listing the available sports. The dialog has a SportID property. To add a sport, the application takes these steps:

1. Show the Sports dialog modally, which means the user must either choose a sport, or cancel, before continuing.
2. When the OK button is clicked, the Sports dialog sets the SportID property to the currently selected sport.
3. Next, the program calls CPerson's AddSport method, passing the SportID as a parameter. AddSport creates a dynaset-type recordset which looks for records in the SPORTLINK table that match this member with the chosen sport. If the dynaset is empty, AddSport adds the required record. If it is not empty, AddSport reports that the member is already linked to that sport.
4. Finally, the program updates the form with the new list of sports.

Finishing touches

There is plenty more work to do in improving the Sports Club application. One professional touch is to enable and disable buttons according to whether or not they are applicable. For example, when no sports are listed, the Remove sport button should be disabled. Next, you can add keyboard shortcuts for mouse-free typing.

Another important area is error-handling, to prevent the program from crashing and to show the user informative messages when things go wrong. For instance, the database could become corrupted.

Finally, there is the issue of multiple users and what happens when two people try to update a record at the same time.

■ *All the code for this month's workshop is on the cover CD. And see Hands On Visual Programming (p301) for answers to queries concerning this workshop and other Visual Basic problems.*

Beating the OLE jargon

OLE has lots of strange jargon. and here are two examples that can cause confusion. Mastering these issues is important to make good use of the technology.

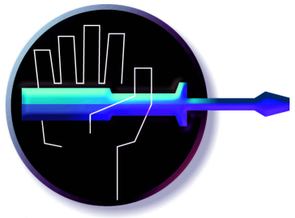
First, you will see references to in-process and out-of-process servers. In-process servers are DLLs which run in the same address space as the calling application, whereas out-of-process servers run in their own address space. This is a decision you take when building a VB executable. In-process servers have substantially better performance but introduce more programming restrictions.

Second, there is the matter of early or late binding. Binding is the process of locating the properties and methods which the client application calls. If you use variables declared as Object in the client application, then these identifiers are not resolved until runtime. This is called late binding. On the other hand, if you use an OLE-type library to resolve these identifiers at compile time, the code will execute faster. This is early binding. To use early binding in Visual Basic, open the Tools - References dialog and check the type library required. Then, declare variables of the specific class required, rather than the generic Object. This is much faster and also enables you to detect errors in parameters, properties or method names when the application is compiled. Another bonus is that you can use an object browser to inspect the interface of available classes.

Naturally, the best performance combines both techniques — that is, in-process servers called with early binding.

PCW Contacts

Tim Anderson welcomes your comments and queries. Write to the usual *PCW* address, or email freer@cix.co.uk.



First class letters

With VB you can use Word to create letters that virtually write themselves. Tim Anderson shows you how. Plus, how to delegate in Visual Basic to achieve the benefits of inheritance.

The "*PCW* Sports Club" is continually sending letters, reminders of coming events, subscription invoices, sympathy for broken bones and the like. The secretary has been running the Visual Basic application to look up the address and then using **Alt-Tab** to switch back and forth from Word while she copies it across. It is time to make her life a bit easier.

The first thought was to use VB's **Clipboard** object to copy addresses. This is easy: just add a **CopyAddress** method to the **CPerson** class, as in Listing 1.

But Windows can do better than that. It is possible to automate far more of the process of getting addresses into Word. Word has a mail-merge wizard that works fine for bulk mailings, but for *ad hoc* letters a custom solution is needed.

Here, I will show you how to create a Word letter wizard for the sports club (see screenshots, Figs1-4). The wizard is for Word 97, since earlier versions do not support Visual Basic. (As an aside, it is possible to do something similar in earlier versions, using the WODBC.WLL Word add-in and getting at data through ODBC. Another possibility is to automate the WordBasic object from a VB application. But Word 97 makes it easier.)

The plan is to create a Word macro using Visual Basic for Applications, accessing the same SPORTS.MDB database.

Because this tutorial is based on VB 4.0, you cannot import the form in Word. The good news, though, is that the **CPerson** class module can be reused, as is. The procedure is as follows:

1. In Word, open the Visual Basic editor. Choose **Tools, References**, and check the Microsoft DAO 3.0 (or higher) object library.

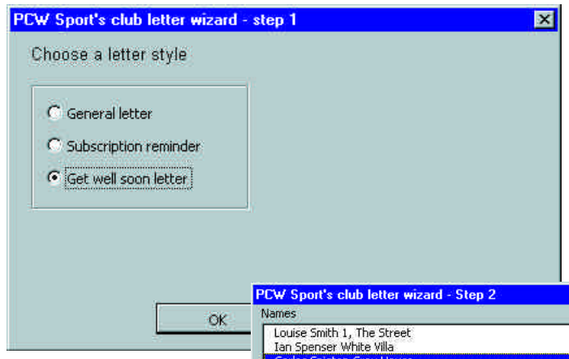


Fig 1 (left) The letter wizard begins by asking which template is needed. You can easily add further options

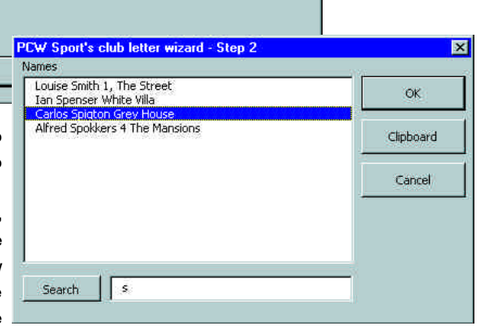
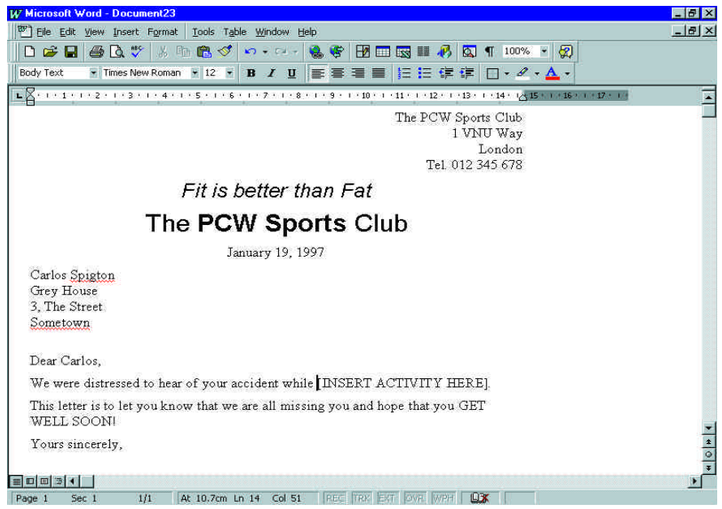


Fig 2 (right) The next stage is to choose a name to write to

Fig 3 (below) In the third step, the wizard has placed the address details into a new document based on the chosen template



Listing 1

```
Sub CopyAddress( )
‘ copies address to Windows clipboard
Dim sAddress As String
Dim cr As String * 2 ‘ fixed-length

cr = Chr$(13) & Chr$(10)

If mForename <> “” Then
sAddress = mForename & “ “ & mSurname & cr
Else
sAddress = mSurname & cr
End If

If mAddress1 <> “” Then
sAddress = sAddress & mAddress1 & cr
End If

...

Clipboard.SetText sAddress, vbCFText
End Sub
```

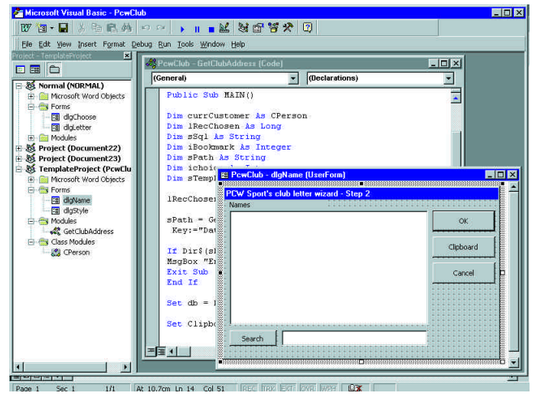


Fig 4 Editing the VBA macro from Word is very like working with standalone VB

This enables Word to use the same data access objects as VB 4.0.

2. Insert a new module into the Normal project. This means the macro will be stored in NORMAL.DOT. Call the macro **GetClubAddress** and give it a **Sub Main**.

3. Insert two new userforms. These will be steps one and two of the letter wizard.

4. Name the first userform **dlgStyle**, and put two or more option buttons on it, along with **OK** and **Cancel** buttons. Give the form a **Choice** property using a memory variable and property procedures. This is for choosing a letter template.

5. Name the second userform **dlgName**, and put on it a listbox, an editbox and three command buttons. This is for choosing a name for the letter.

6. In the Declarations section of **GetClubAddress**, declare a public database object. For the example code, I have also declared some convenient constants. Then in **Sub Main**, open the SPORTS.MDB database using code like:

```
Set db = DAO.OpenDatabase(sPath &
 “\SPORTS.MDB”)
```

sPath is a variable to store the path to the database file. (See below for how to get this path from the system registry.) **Sub Main** also creates a new **CPerson** object.

The **Main** procedure continues by opening **dlgStyle** to obtain a choice of template, and then **dlgName** to get the ID of name in the **Members** database. At each point, the user has an option to cancel. The code for the **dlgName** dialog is almost the same as that used in the main VB 4.0 application, the main difference being that VBA has no data control so you have to create a recordset in code. When a member ID has been retrieved, the record is loaded into the **CPerson** object.

7. The final step is to start a new document based on the chosen template. The templates must be pre-designed with bookmarks where the name and address information is needed. The wizard finishes by inserting the fields in the bookmark positions and then exits. Controlling Word from VB in this way is not difficult using Word's new object model. For example: p258 ➢

Delegating your inheritance

■ A common criticism of Visual Basic is that it doesn't support inheritance. If all your programming has been done in Visual Basic, which is probably true of the majority of VB programmers, this may not mean much to you. Fortunately, it's easy to explain. A class, both in VB and other object-orientated languages, defines an object. In VB, every class starts from scratch without any properties or methods. By contrast, C++, as an example, lets you begin a class definition like this:

```
class monkey : public animal
```

The result is that the monkey class inherits the properties and methods of the animal class. The monkey class just needs to add specialised code that describes monkeys; the generic animal code comes for free.

Although VB does not support inheritance, there are other ways of achieving some of the benefits. It is possible to contain one class within another. Then you can implement properties and methods of the parent class by calling the properties and methods of the contained class. This is called delegation, and the properties and methods of a class are called its interface. For example, the tutorial application has a CPerson class. Imagine you wanted to create a CEmployee class which used the properties and methods of CPerson. Here is how you can do it:

1. Insert a new class module and set its name property to CEmployee.
2. In the declarations section, put:

```
Private m_person As CPerson
Private m_wage As Currency
```

3. In the initialise section put:

```
Set m_person = New CPerson
```

4. Create a CEmployee interface that calls the CPerson interface. For example:

```
Public Property Get surname( ) As
 String
surname = m_person.surname
End Property
```

5. Add new properties and methods specific to CEmployee. For instance, you must expose the wage property.

The fourth step (*above*) is tedious, but beats re-coding all the functionality of CEmployee in CPerson. It could be automated by a VB Wizard. In Visual Basic 5.0 this approach to object-orientation is built into the language, with a new Implements keyword which guarantees that all the methods of the contained class are implemented by the outer class. You can implement the interface of any ActiveX automation server. Finally, there is nothing to stop you implementing several interfaces in a single class.

Delegation works, but it is neither as intuitive nor as elegant as traditional inheritance. For the moment, though, this is the VB way. It ties in with ActiveX, the component model which is becoming more powerful and pervasive as Windows evolves. VB may not be the fastest or most thoroughly object-orientated language out there, but Microsoft does ensure that it stays up to date with the latest ActiveX developments.

```
Documents.Add (sTempLate) ‘ starts
a new document based on the given
template

ActiveDocument.Bookmarks(“name”).
Select ‘ sets cursor to the “name”
bookmark in the new document

Selection.InsertAfter Trim
(currCustomer.forename & “ “ _
& currCustomer.surname) ‘ inserts
text at the cursor position
```

Problem-solving

There are a few things to notice about this joint Visual Basic and Word project. Although Word VBA is downward compatible with VB 4.0, there are some objects which are available in VB but not VBA. One example is the global App object which, in Word, is the Application object.

The original **CPerson** class used **App.Path** to discover the location of SPORTS.MDB. This strategy fails in any case, when the code runs in other applications. A better idea is to use a registry entry, using VB's **GetSetting** command. The registry entry is created by the main VB 4.0 application when it first runs. This way, the data can easily be found by any Windows application.

Another catch is that VBA has no Clipboard object, so CPerson's CopyAddress method does not compile in Word. The workaround is to declare a public Clipboard variable as a **DataObject**: VBA's private version of the clipboard. To demonstrate, there is a Clipboard button on the **dlgNames** form which uses the DataObject's **PutInClipboard** method to transfer text to the read clipboard.

Enhancing the wizard

There are plenty of ways you can improve on the Letter Wizard. For instance, you can add database fields for things like job title and salutation. You could increase the range of templates on offer. For the subscription template, you could write code to check a person's outstanding balance and insert the amount into the letter. By adding the bulk of the code to a shared class module like CPerson, you can easily reuse it in VB 4.0 or in other VBA applications such as Excel.

Installing the example code from the *PCW* CD

When you unpack the tutorial code from our cover-mounted CD, you will find a VB 4.0 project and a Word 97 template. To install the example code, copy PCWCLUB.DOT into your Word templates directory. Then start a new document based on this template. If you then choose Tools, Macro, Visual Basic Editor, you will find the example macros. Choose Tools, Macro, Run, to run the macro. You can also copy the macros into NORMAL.DOT if you want, by using Tools, Templates and Add-ins, Organizer. Finally, the macro will not run without a registry setting for the data path. To create this setting, run PCWCLUB.EXE.

■ *Next month: Back to native Visual Basic for the final stage in the PCW Sports Club application.*

PCW Contact

Tim Anderson welcomes your comments and queries. Write to the usual *PCW* address, or email pcw@vnu.co.uk.