Task masters

Illustration by PAUL SHORROCK

Choosing the right tool for the job in hand is vital for successful programming. Here, we have taken six leading products and analysed their suitability for three different tasks. First we have looked at creating a simple text editor. Next up is developing an all-action game, and finally comes a web-enabled address book application. The chosen products include three Windows development tools and three for Java. First for Windows is **Microsoft Visual Basic 6.0**, the latest version of the pioneering visual development tool and the most widely used programming product. Second is **Microsoft Visual C++ 6.0**, demanding to learn but powerful. Third comes **Inprise Delphi 4.0**, the package that, when first launched, stunned developers by combining VB's ease of use with the speed of natively compiled code. The Java lineup begins with **Inprise JBuilder 2.0**, which aims to bring Delphi's productivity to Java. Next comes **IBM's VisualAge for Java 2.0**, which does true visual programming and has a built-in code repository. Finally, **Visual Café 3.0** is the market-leading Java development tool from Symantec, one of the first to release a full-featured integrated development environment for Java.

TIM ANDERSON

What is Visual Programming?

Most programming languages are text-based. You write the code in English, and then run some sort of compiler to convert your code into a form the computer can actually execute. To save time and reduce errors, modern development tools do some of this work for you. For example, it would be tedious to position buttons on a form by counting pixels, so instead you can use a graphical form designer to place and size them with the mouse.

Most packages take this a stage further, letting you link text boxes to database fields, or create an event handler to define what happens when you click a button, without having to write the code yourself. A key concept is the property sheet, which lists the properties of a visual or non-visual object and lets you select or type-in values.

Development tools that offer these features are called visual programming tools, although purists use the term in a slightly different way. True visual programming lets you draw the logic as well as the appearance of an application. SmallTalk is often programmed this way, but it has never caught on in the mainstream. The closest example in this round-up is IBM's VisualAge. The other products are hybrids, environments that offer visual assistance but which still leave you looking at a flashing text cursor to create the code that drives the interface. Even then, some are more visual than others. The least visual of the products featured here, despite its name, is Microsoft Visual C++, replete with wizards but firmly text based.

Although visual programming does boost productivity, there are disadvantages. In Java's case, the visual tools have struggled to keep up with the pace of Sun's changes to the JDK, the most recent example being compatibility problems with JDK 2 (formerly 1.2). Some developers get round this by using the straight JDK along with their favourite programmer's editor.

Another problem with visual programming is that the environment and class library often adds unnecessary code, to allow for features that you may not want to use. If you want to build the smallest possible Windows application, for example, forget MFC or Delphi's VCL and do it the old way with plain C. In the real world though, productivity usually counts for more than raw performance.

In the real world, productivity counts for more than raw performance

What is Java?

Java began in 1991 as a Sun Microsystems project for embedded systems. It is based on two concepts. The language has a family resemblance to C++, but is both simplified and enhanced to make it safer and more object orientated. Second, it compiles not to native code but to an intermediate form that runs on an interpreter called the JVM (Java Virtual Machine). Sun realised that Java was ideal for apps that run in web browsers, and in 1996, when Netscape Navigator 2.0 included the ability to run Java applets, interest in it soared.

There are several factors behind Java's popularity. First, it is a well-designed language that is more productive than older rivals like C++. Second, Java's cross-platform talents make it a rallying point for anyone in the industry who would like to dislodge Windows from desktop dominance. Third, Java is a web-aware platform with built-in security and comms features. It is also a good fit for CORBA, an architecture for distributed objects, giving both technologies a mutual boost. Fourth, things are turning full circle as interest in embedded systems is once again intense.

Project 1 **Text editor**

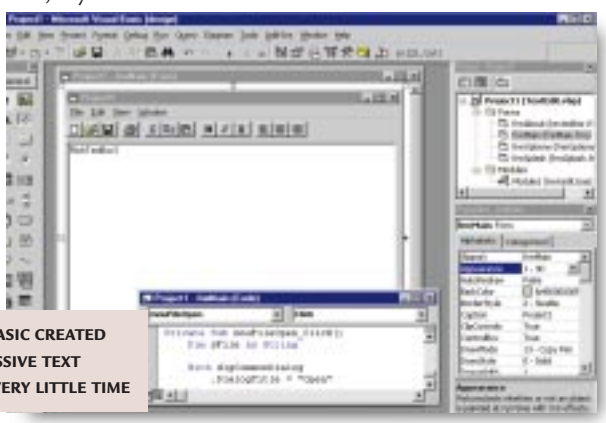
A new text editor is a revealing project for a test. It's a good starting point for any application that handles documents, and includes features like menus, toolbars, and printing functions. Following the latest trend, we decided to create SDI (single document interface) editors, with at least the ability to open, edit, save and print.

Visual Basic gets off to a cracking start with its Application Wizard. You can pick an application style, check the menu and toolbar options you want, and optionally embed a web browser into the application. The wizard application looks good, but selecting an option simply displays a 'to-do' dialogue.

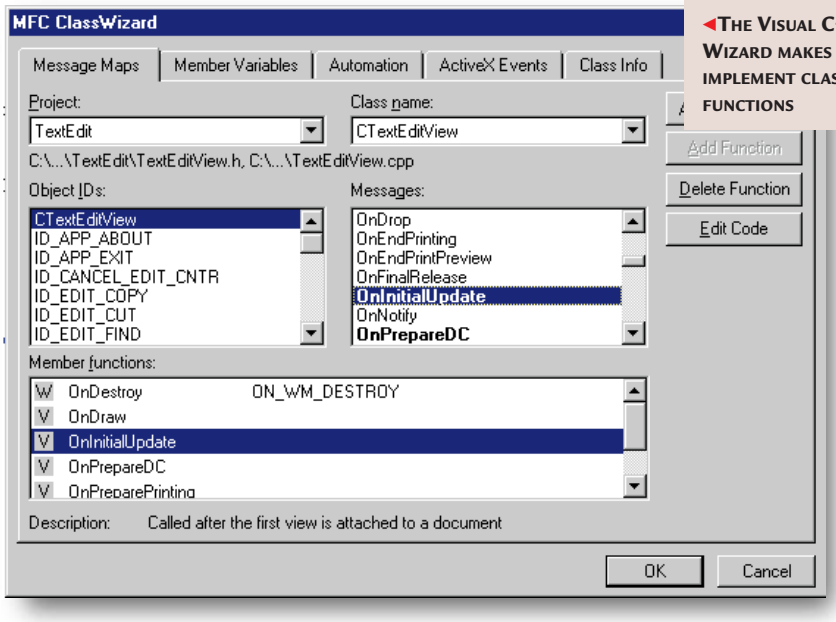
It is not hard to finish the job. Rather than stick with simple text, right-clicking the toolbox lets you add a rich-text ActiveX control. To get this to fill, the client area required a few lines of code in the form's resize event handler. The File - Open menu already displays an open dialogue, and one additional line of code loads the chosen file into the editor.

Save is nearly as easy, as is hooking up the bold, italic and underline options. The reason is that all these functions are built in to the rich-text control. The only problem with this kind of black-box development is when you want to modify the behaviour, say to print a page at a time based on the selected printer. You can do this, but it means diving into the Windows API and writing a substantial amount of code, breaking the illusion of simplicity. Even so, VB is hard to fault for this kind of project.

Visual C++ has an AppWizard that looks on the surface like the one in Visual Basic but in reality is more sophisticated. I selected an MFC application with an SDI interface again, and with support for the Document-View architecture. This is ideal for an application like a text editor or word processor, as it separates the code which handles the document from that which



▶ VISUAL BASIC CREATED THIS IMPRESSIVE TEXT EDITOR IN VERY LITTLE TIME



THE VISUAL C++ CLASS WIZARD MAKES IT EASY TO IMPLEMENT CLASS MEMBER FUNCTIONS

displays it on the screen. Unfortunately it is also fairly complex.

A key step in the AppWizard is to change the Base class to the view closest to what is required. I chose the CRichEditView. AppWizard then generated dozens of files complete with an explanatory readme, to make an instant application that had some of the required functionality. For example, you could type in the window, and save and load files.

The Visual C++ interface is very slick, but that does not make it easy to get started. For example, a natural next step is to change the application's default font from the ugly System font. There is no friendly property sheet for this, so you have to add code, and it is not obvious where to add it. The font is a function of the display, so you might try the OnInitialUpdate method of your View class. Visual C++ makes it easy to find the code, either in the Class View or by right-clicking the code window and choosing ClassWizard.

Next, declare a CFont object, call one of its CreateFont methods, and pass it to the SetFont method of the View. Visual C++ has excellent pop-up help for the fourteen parameters of CreateFont, although you still need to look up constants like ANSI_CHARSET.

Skilled users will find plenty of time-saving features in Visual C++, but this will never be RAD. On the other hand, performance is excellent, and anything you can do in Windows, you can do with Visual C++.

Delphi 4 has an SDI application wizard which, unlike those in VB and Visual C++, has no options at all but nevertheless builds a sensible frame for an editor, with a menu and toolbar complete with standard functions like open, save, cut, copy and paste. The default functionality is very limited, but adding to it is as simple as in Visual Basic. I dragged a RichEdit control onto the form, and set its alignment to automatically fill the client area, a neat timesaver and one point scored over VB. Next, I hooked up the Open function by adding a call to the LoadFromFile method of the RichEdit control's Lines property.

It helps to know beforehand that the Lines property has this key method. If you look up TRichEdit class in online help, you will not find any method for loading from a file. If you inspect the Text property, you find it is simply a string. How do you find out that the

VISUAL PROGRAMMING & JAVA JARGON

ADO ActiveX Data Objects, Microsoft's latest data access API based on COM.

API Application Programming Interface, a set of functions that allow programmatic access to the specified features.

BDE Borland Database Engine, used in Delphi and in the DataGateway middleware product.

Class library Prewritten code that defines a set of general-purpose objects for use in building applications.

COM Component Object Model, Microsoft's specification for local and distributed objects.

CORBA Common Object Request Broker Architecture, an industry standard for communication between distributed objects.

DirectX Microsoft's API for fast graphics and multimedia programming.

Embedded systems Applications embedded into devices other than computers.

IDE Integrated Development Environment, the combination of a code editor, design tools, and control over compilation.

ISAPI, NSAPI Internet/Netscape Server API, a specification for running code on web servers.

JavaBean A Java class designed for visual manipulation in a form designer.

JDK Java Development Kit, now sometimes known as the Java Platform, the JVM and core classes for developing and running Java software.

JET The database engine used in Access, Visual Basic and MS Office.

MFC Microsoft Foundation Classes, the standard C++ class library for Windows.

ODBC Open Database Connectivity, a specification for data access mainly used from Windows.

OpenGL 3D graphics library developed by Silicon Graphics and available on various computer platforms.

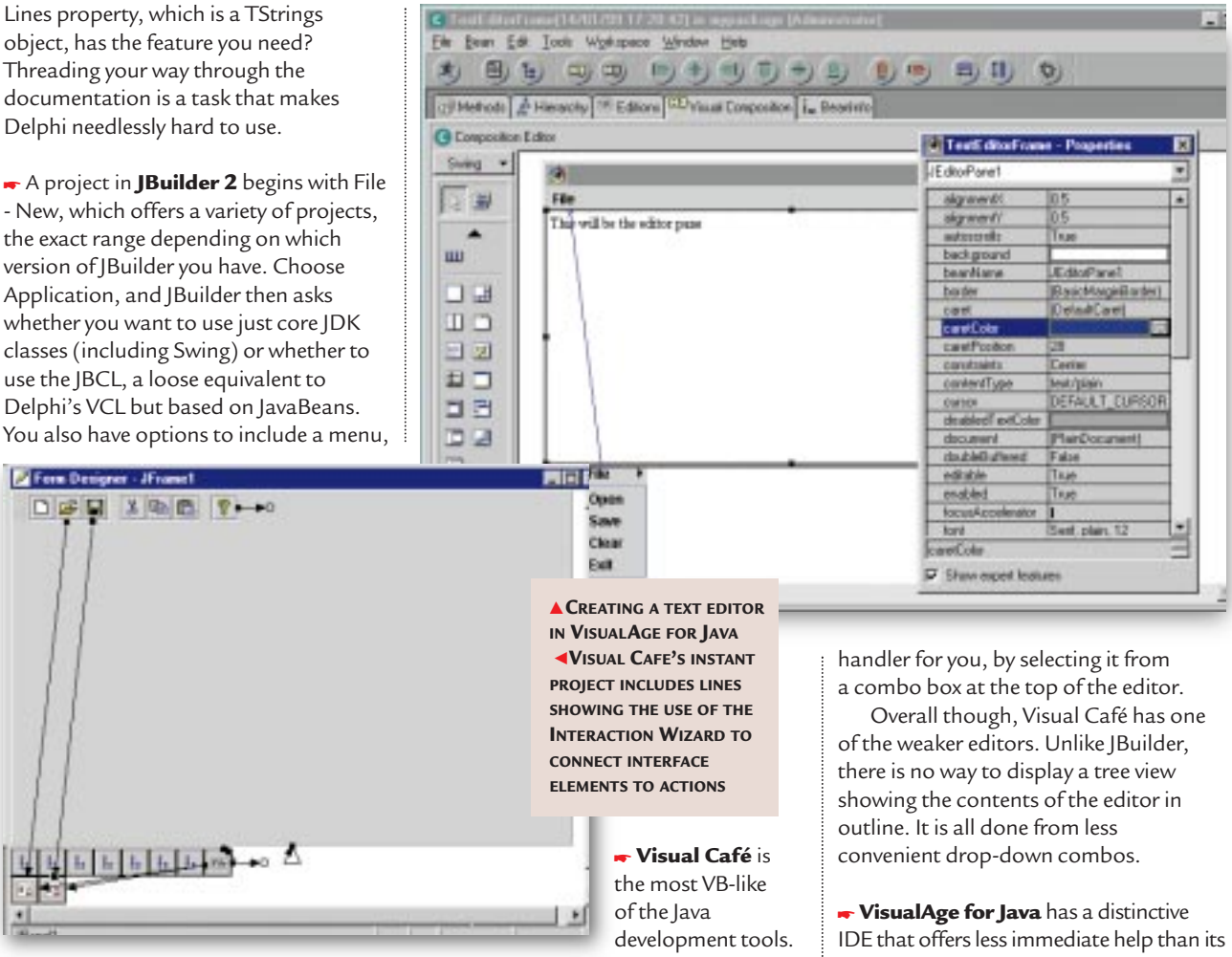
RAD Rapid Application Development, using code generation, components and/or a class library to snap applications together quickly.

SQL Structured Query Language, an industry-standard way to define database queries and commands.

VCL Visual Component Library, Delphi's class library written in Object Pascal.

Lines property, which is a TStrings object, has the feature you need? Threading your way through the documentation is a task that makes Delphi needlessly hard to use.

A project in **JBuilder 2** begins with File - New, which offers a variety of projects, the exact range depending on which version of JBuilder you have. Choose Application, and JBuilder then asks whether you want to use just core JDK classes (including Swing) or whether to use the JBCL, a loose equivalent to Delphi's VCL but based on JavaBeans. You also have options to include a menu, toolbar and about box. JBuilder then built the application, which was a poor thing compared to the slick Windows offerings. A simple frame with a menu and toolbar of just three non-functional options isn't much, but it's a start.



▲ CREATING A TEXT EDITOR IN VISUALAGE FOR JAVA
◀ VISUAL CAFE'S INSTANT PROJECT INCLUDES LINES SHOWING THE USE OF THE INTERACTION WIZARD TO CONNECT INTERFACE ELEMENTS TO ACTIONS

The first task is to add a text area to the frame. I changed its layout manager to BorderLayout and then dropped a JBCL TextArea component onto it, using the Design tab of JBuilder's project manager. It automatically filled the panel.

To load documents, I added a JBCL filer component and an Open menu item. JBuilder creates skeleton event handlers for you, and also pops up parameter help as you type, speeding the work significantly. JBuilder's online help provided a code snippet that reads text from a file, and adding this plus an import for java.io.* produced a working Open function.

JBuilder is a RAD environment a little harder than VB or Delphi, but easier than Visual C++. It's also worth noting that the Swing JEditor component has interesting features like the EditorKit class, letting you use provided file formats like RTF or HTML, or define your own.

Visual Café is the most VB-like of the Java development tools. It performs better because it is itself a Windows app, unlike most of JBuilder and VisualAge. Help is Windows help, and easier to use than JBuilder's sluggish help viewer. I started with the New Project dialogue and chose a JFC (Swing) application. This was more feature-rich than the JBuilder equivalent, complete with open and save dialogues, and a toolbar with cut, copy and paste options. Lines on the project show the use of the Interaction Wizard, which lets you connect user interface objects to actions using a dialogue. A right-click option lets you edit the Java code manually. For this editor I used a JEditorPane component and set its Placement property to fill the client area. The open function was hooked up by creating an openFile method to read the file returned by the openFile Dialog component, calling setText from JEditorPane to load the text.

Like the other products in this test, Visual Café has a Code Helper that shows valid methods in a drop-down list as you type. It can also generate an event handler for you, by selecting it from a combo box at the top of the editor.

Overall though, Visual Café has one of the weaker editors. Unlike JBuilder, there is no way to display a tree view showing the contents of the editor in outline. It is all done from less convenient drop-down combos.

Visual Café is the most Visual Basic-like of the Java development tools

VisualAge for Java has a distinctive IDE that offers less immediate help than its rivals. However, it is more helpful than it first appears. If you just add a JFrame class to a package, and then run it, VisualAge will generate the required code for the main method. Next, I opened the frame window in the Visual Composition editor and added a JMenu and a JEditorPane. I set the containing frame to use the border layout, the menu to appear North, and the EditorPane Centre. Adding items to the menu was a matter of dragging JMenuItem beans from the Swing palette to the cascading JMenu. Then a FileDialog was added. VisualAge lets you connect menu items to code through an innovative Connect feature. Choose an action such as actionPerformed, then a target action such as calling the show() method of the FileDialog, and VisualAge will generate the necessary code.

VisualAge is a strong RAD environment despite its lack of wizards. The built-in repository is great, except when you want to use any external tools, when it is a matter of exporting and re-importing the code.

Project 2 **Developing games**

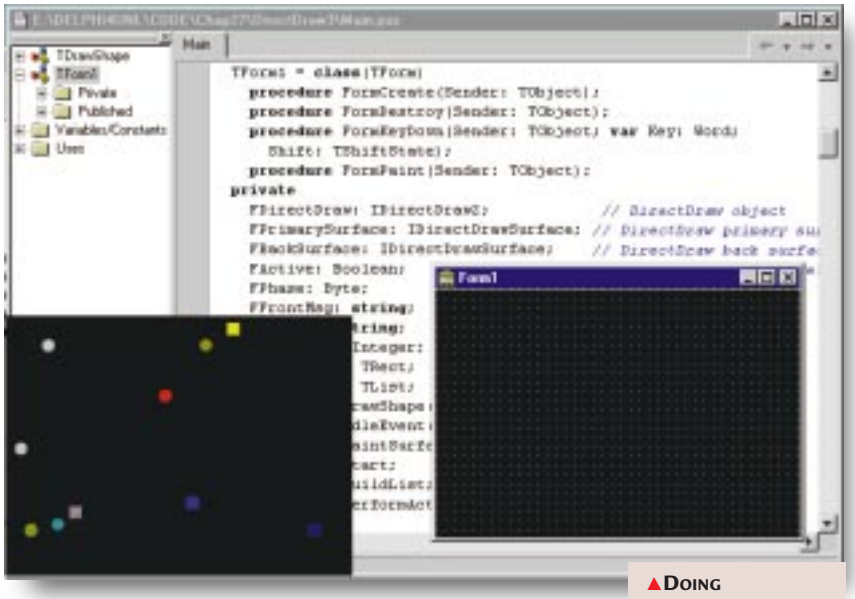
Games are among the most demanding applications. They are also poorly served by many development tools, which are aimed primarily at business users. Strategy games aren't so bad, but fast-action projects are another matter. Games development does not actually lend itself to visual programming, because conventional graphical widgets like buttons and list boxes are not much used.

On the face of it, Visual Basic has little to offer the games programmer. The reason is that most Windows games development is done using DirectX, which aims to provide a device-independent API for multimedia programming while retaining the performance of direct hardware access. Most DirectX development is done in C or C++. Since most of DirectX is a COM API, it is theoretically possible to use most of it from VB and third-party wrapper ActiveX controls have been produced. There is little advantage though, and the VB runtime is an overhead most games can do without.

Delphi is more promising although again Inprise has done little to make things easy. The DirectX API is not wrapped by the Delphi VCL, but third parties have stepped into the breach to create DirectX.pas. Delphi does include declarations for the Windows OpenGL API. The declarations are all you get, though. No information on using the API is supplied with Delphi.

What this boils down to is that Delphi is well able to support both DirectX and OpenGL, but developers have to suffer the fact that all the documentation assumes use of C/C++. For someone with Delphi skills, it may still be worth using.

Visual C++ is the natural home for games developers. Note though that neither DirectX nor OpenGL are part of MFC; many of the features of Visual C++ are therefore not relevant. Visual C++ does have an excellent code editor and browser which, along with the fact that the relevant SDKs are C/C++ based, makes this the best choice for multimedia coding.

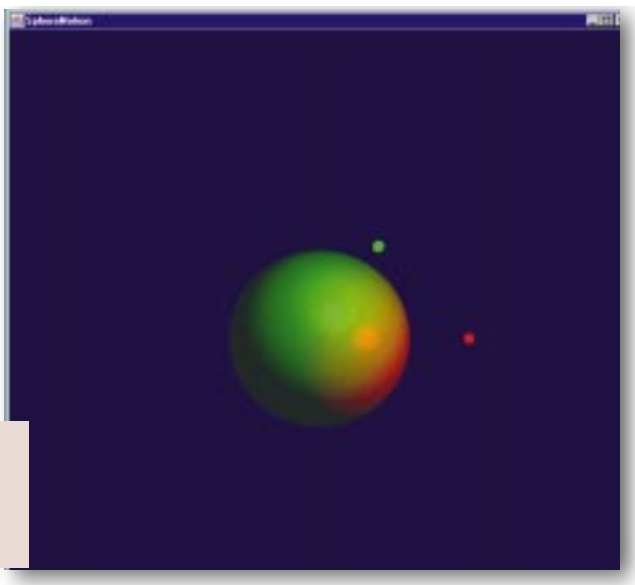


▲ DOING DIRECTDRAW IN DELPHI — NOTE THE UNEXCITING FORM DESIGN. THE IMAGE AT BOTTOM LEFT SHOWS THE RUNNING APPLICATION

Despite its performance problems, Java has multimedia potential. Relevant APIs include Java 2D, for 2D imaging, Java Media Framework, Java Collaboration for real-time multi-player games, Java Animation and Java 3D. This last is a collaborative effort from Intel, Silicon Graphics, Apple and Sun. Java is a high-level API that is intended to scale smoothly as the performance of the underlying hardware improves. Part of the attraction of Java 3D is that it will work in web browsers without needing a plug-in or ActiveX control. The current Windows implementation requires OpenGL support, with a DirectX version in preparation.

Visual C++ is the natural home for games developers ... it has an excellent code editor and browser

Which Java product is best for working with the Java multimedia APIs? Here, there is actually a good case for working directly with the Sun JDK and your favourite code editor. Having said that, the pick of the three products covered here would be JBuilder, thanks to its superior editor. Visual Café comes next, and benefits from a native code compiler, while Visual Age is a poor choice for several reasons. Its editor is nothing special, and the inability to switch to different versions of the JDK counts against it.



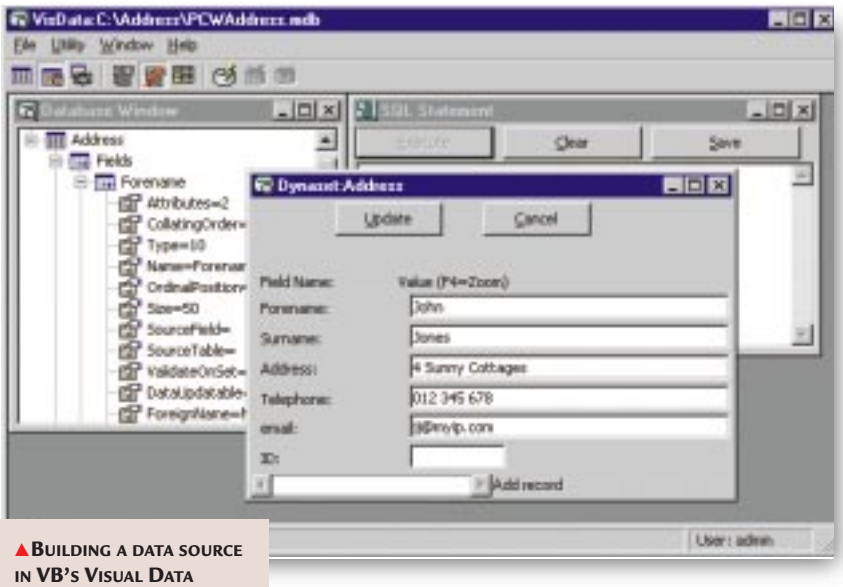
▶ 3D GRAPHICS IN JAVA. THIS IS THE SPHEREMOTION EXAMPLE

Project 3 **Web database development**

Visual Basic is stuffed with database features. It includes the same database engine as that used in Microsoft Access. Alternatively you can get at data through ODBC, or through ADO, the newest Microsoft data access API.

The starting point is to define the data source itself, which you can do through VB's Visual Data Manager. Ideally, a web data source should use a client-server database such as SQL Server or Oracle, but for small systems Microsoft Access will do. Using Visual Data Manager you can create and define fields and enter data.

One way to present the data on a web page is through a new feature called WebClasses. In the New Project wizard, choosing IIS Application starts a skeleton project that will run on Internet Information Server. The project opens with little indication of what you should do next, which is to create an HTML template with replaceable tags, and write code to connect to the data and generate HTML in response to browser requests. VB WebClasses get good marks for functionality, but ease of use needs attention.



▲ **BUILDING A DATA SOURCE IN VB'S VISUAL DATA MANAGER**

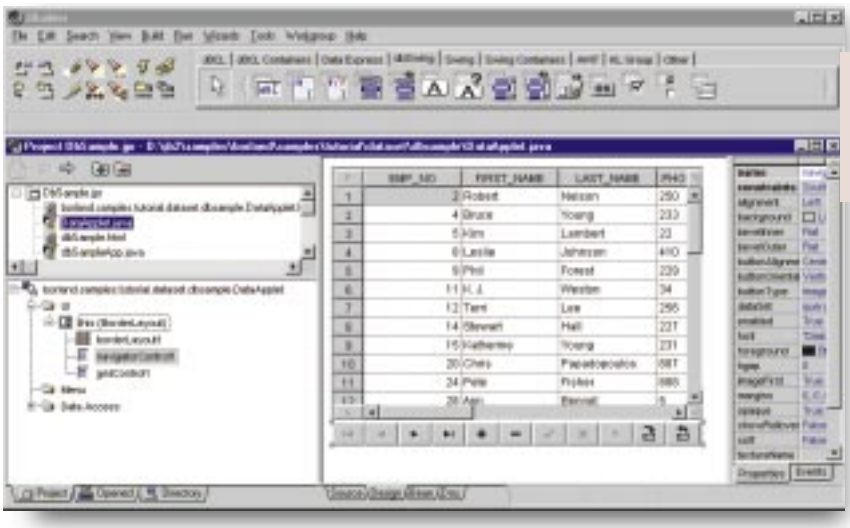
Delphi 4 is equally replete with database tools, particularly in its high-end version. A database desktop utility, essentially a cut-down version of Paradox, lets you create and edit data sources. Delphi's built-in database engine has native drivers for Paradox and dBase as well as SQL links including Oracle, InterBase and SQL Server. Building a conventional Windows application with data access is straightforward, but in this case a web application is needed. Fortunately, Delphi has a great feature for web database development, which is the capability of creating NSAPI or ISAPI DLLs, although you need at least the client-server version. These are code libraries that run on a web server, and they are fast and efficient. For the most scaleable applications you need to consider a fully distributed solution using DCOM or CORBA, but a simple ISAPI DLL will be fine for many organisations.

Delphi has a Database Web Application wizard to get you started. Like VB's WebClasses, it is not really drag-and-drop programming, but it is well designed. TWebRequest and TWebResponse objects encapsulate HTTP requests and responses respectively. These are passed as parameters to a TWebDispatcher, which processes the request. This can be integrated with a data code module for smooth data access. Deployment is a matter of installing the compiled DLL on the web server, and calling it from web pages with any required parameters added to the URL.

Visual C++ also has an ISAPI extension wizard, although NSAPI is not supported. The wizard uses MFC classes, including CHttpServer, CHttpServercontext and CHtmlStream. Little help is offered, though, over how to use the classes or how to combine them with data access. Having said that, Visual C++ comes bundled with MSDN, which has a section on developing ISAPI extensions. Visual C++ is for those who want extra performance at the expense of rapid development.

JBuilder is a natural choice for building a web database application. With an applet or application project open, you can add a database component which will prompt you for a URL that defines a JDBC database connection. Choices include the Sun JDBC-ODBC bridge, or the DataGateway, middleware that lets you connect to the Borland Database Engine via JDBC or a native JDBC driver.

Next, you can add a queryDataSet component and set its query property to link to the database component, defining an SQL query to determine what data to extract. Finally, adding

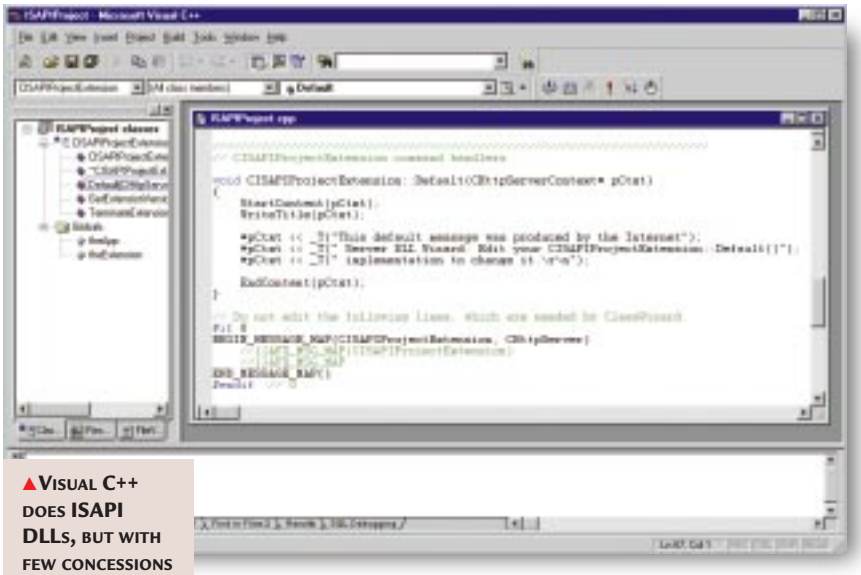


◀ **CREATING AN INSTANT DATABASE PROJECT IN JBUILDER**

data-aware controls such as a grid and data navigator lets you build an instant database application.

Using Java applets is not the easiest way to make data accessible over an intranet or on the web. A server-side DLL such as those Delphi builds will often be quicker and easier to deploy. JBuilder's equivalent would be Servlets, Java components that run on compatible web servers. It is early days for Servlets and I would expect them to be more prominent in the next JBuilder.

Visual Café has a database edition. There is a database project wizard and it stepped smoothly through several dialogues to create an applet linked to an SQL Server database. The result not only allows data view and navigation, but also includes a Query By Example button that lets you enter criteria for searching the data. Database components in Visual Café include a connection manager, data navigator, and adapters for using stored procedures, validating data and displaying calculated fields. These work with any JDBC driver. Symantec's solution to overcome the limitations of the JDBC-ODBC bridge is a piece of middleware called dbAnywhere. This has a JDBC driver, the idea being that you connect to dbAnywhere through JDBC, and dbAnywhere then connects to the data using ODBC or its own drivers.



▲ **Visual C++ does ISAPI DLLs, but with few concessions to rapid development**

To use Visual Age for database work, you need tthe high-end Enterprise version, unless you are happy to write your own JDBC data access code. The Enterprise edition includes data access beans. Using these, you can create data access classes for any JDBC data source. IBM provides a JDBC driver for DB2, an IBM server database available on a number of platforms. The Select bean is a non-visual component for making a connection, while the DBNavigator is a visual bean for navigating a dataset.

Java Futures

Java, Java everywhere; but where next for Windows?

Ironically, the one place Java has stalled is where it first succeeded: applets in web pages. They are used to some extent, but web developers are put off by performance and compatibility problems. The excitement is elsewhere. First, the use of Java for distributed applications is soaring, particularly as useful tools emerge. Java applications combined with a CORBA-compliant object broker have great potential, exploited by tools like Inprise Application Server, NetDynamics application server (NetDynamics is now part of Sun) and Apptivity from Progress Software. Second, Java APIs are extending the reach of the platform, for example in multimedia, email and commerce. Third, Java is going back to its roots as a platform for embedded systems.

Personal Java is designed for handheld computers. Jini is a plug-and-play networking system that will let any Java-enabled device easily connect and disconnect. You can expect to see Java in mobile phones, set-top boxes, cars, and in office and household equipment.

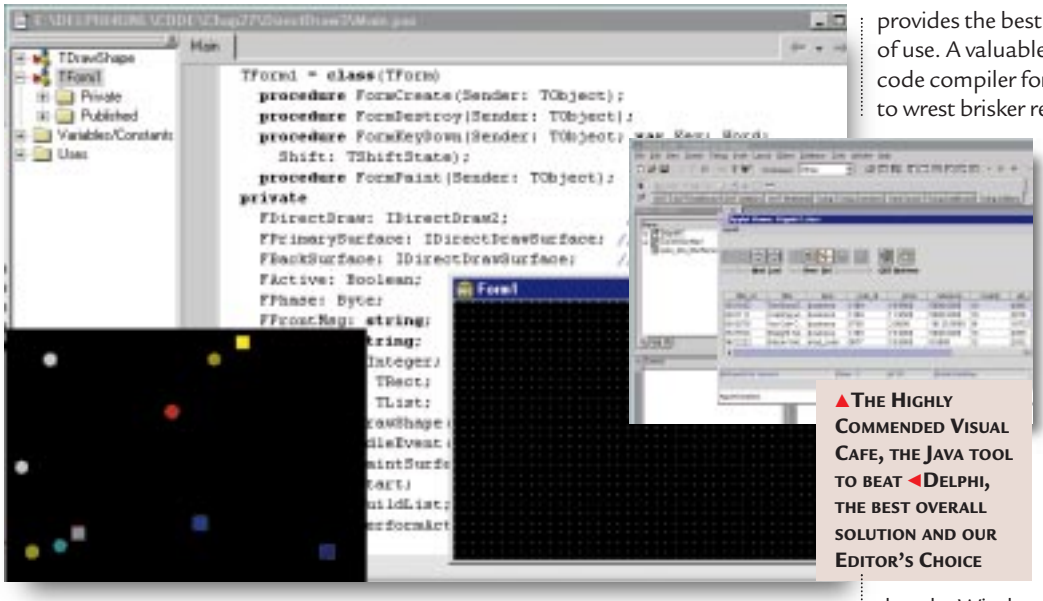
One company not likely to get much advantage from Java is Microsoft, and when it took out a licence to support Java in Internet Explorer, some suspected it would try to fragment the standard. Microsoft has in fact introduced extensions that make Java work with COM and with the native Windows API. It is also understandably reluctant to support newer Java features that compete directly with Windows and with COM, to such an extent that Sun and Microsoft became locked in legal dispute.

It appears likely that Microsoft's JVM will have to conform with Sun's standards, including the JNI (Java Native Interface), although whether this will mean significant changes to Visual J++, whose class library depends on native calls to the Windows API, is not yet clear. In principle, creating Java apps that depend heavily on native methods is not a breach of Java's specification, but goes against the grain of Java's direction. Even if Microsoft were to drop or freeze Java support, it will still run on Windows and, through third-party extensions, in Explorer, so the outcome is unlikely to influence Java's progress.

Another Java problem is that its cross-platform ability involves compromise. Java applications are typically less snappy and slick than Windows equivalents: projects like Corel's Java suite have been put on the back burner for performance reasons. A combination of faster hardware and clever compilation and optimisation will solve performance problems, but how soon is unclear.

Finally, there's industry politics. Microsoft will compete as best it can with Java. Other companies are puzzling out the long-term implications of Sun's control over Java: Sun is submitting Java as an ISO standard, but retains ownership of the platform. IBM, Novell, Oracle and Inprise are basing some or all of their future strategy on Java, so one way or another it will be a prominent part of computing as the new millennium unfolds.

Editor's Choice



▲THE HIGHLY COMMENDED VISUAL CAFE, THE JAVA TOOL TO BEAT ◄DELPHI, THE BEST OVERALL SOLUTION AND OUR EDITOR'S CHOICE

This is not a comprehensive group test of VB and Java tools; rather, it's a brief look at how they might be used for three common tasks: a document editor, a game, and a web-enabled database. As an added twist, it is an opportunity to look at Java's suitability for real-world tasks. All the products have something to commend them, but there are clear winners as well.

Only one provides a convincing solution in all three categories. The **Editor's Choice** is **Inprise Delphi**. It is not cross-platform but does let you build browser-independent web applications. It reaches all the way from RAD business apps to fast graphics using DirectX. It beats Visual C++ on ease of use and Visual Basic on performance. In the Java category, it is the **Highly Commended Visual Café** that provides the best performance and ease of use. A valuable feature is the native code compiler for Windows that helps to wrest brisker results from Java applications. For everyday use, this is still the Java tool to beat, although its editor and IDE have room for improvement. Visual Café looks less good if you are targeting large-scale distributed applications. For this, JBuilder, with its VisiBroker integration, looks a better choice. Looking at all six tools together, it is apparent that the Windows tools provide richer environments and faster performance. For most general-purpose applications, Windows currently provides a better solution than Java. But there are still reasons to use Java. One is for distributed and web applications, where Java is a natural fit. Another is that Java's design elegance makes it better for learning, and more productive for beginners and advanced developers alike.

Table of Features

Product	Visual Basic 6	Visual C++ 6	Delphi 4	JBuilder 2	Visual Café 3	VisualAge Java 2
Supplier	Microsoft	Microsoft	Inprise	Inprise	Symantec	IBM
URL	www.eu.microsoft.com	www.eu.microsoft.com	www.inprise.com	www.inprise.com	www.symantec.com	www.ibm.com
Tel	0345 002000	0345 002000	0118 932 0022	0118 932 0022	0181 317 7777	01256 343000
Price ex VAT	£69 / £386 / £916	£386 / £916	£78 / £415 / £1570	£79 / £409 / £1549	£185 / £494	£66 / £1299
Price inc VAT	£81 / £453 / £1076	£453 / £1076	£92 / £488 / £1845	£93 / £481 / £1820	£217 / £580	£78 / £1526
Language	Basic	C/C++	Pascal	Java	Java	Java
RAD	✓	✗	✓	✓	✓	✓
Code completion	✓	✓	✓	✓	✓	✗
Help format	Compiled HTML	Compiled HTML	Windows help	HTML viewer	Windows help	HTML web server
Data access	JET, ODBC, ADO	JET, ODBC, ADO	BDE, SQL Links	JDBC, DataGateway	JDBC, dbAnywhere	JDBC, DB2
Native code compiler	✓	✓	✓	✗	✓	✓
Switchable JDK	n/a	n/a	n/a	✓	✓	✗
JDK version	n/a	n/a	n/a	1.1.6	1.1.7	1.1.6
COM support	✓	✓	✓	✗	✗	✗
CORBA support	✗	✗	✓	✓	✗	✓
Version control	SourceSafe	SourceSafe	PVCS	PVCS	Third party support	Integrated
Rating	★★★★	★★★★	★★★★★	★★★	★★★★	★★★

Where several prices are shown, this is for Standard, Professional and/or Enterprise versions. Some features are not available in all editions.