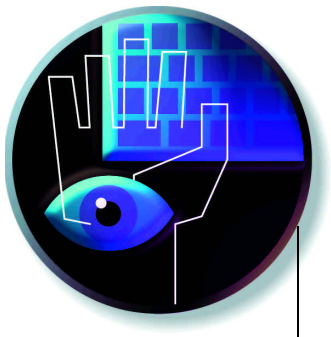


Active service

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

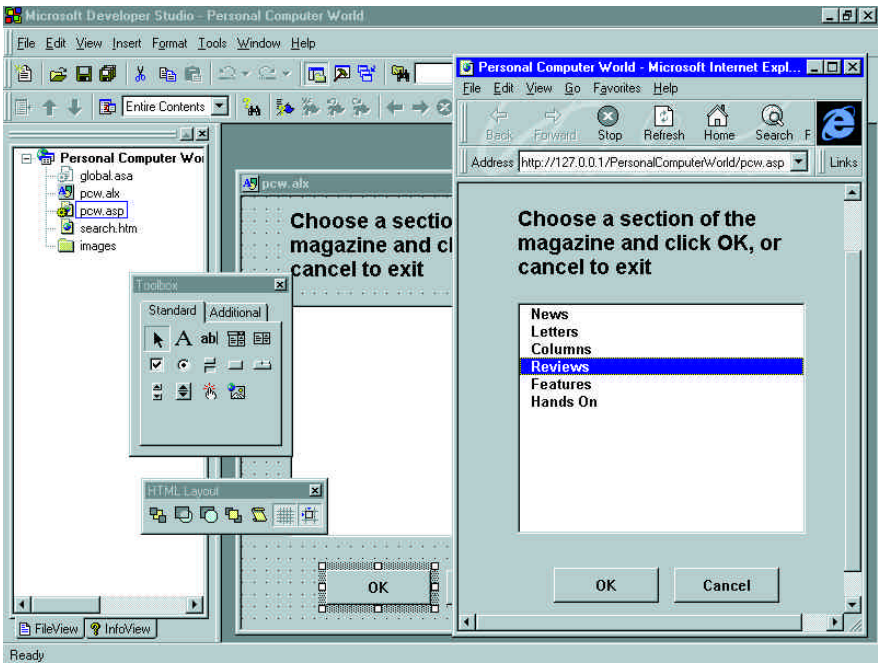
I'm sitting here looking at a sheaf of press releases and a stack of CDs which comprise the Microsoft Active Platform in its current, beta guise. The papers are an intricate display of verbal gymnastics: there are generous sprinklings of key buzzwords like open, standards-based, scaleable, multiple operating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

Where there's an O there's an A

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

The truth is more prosaic. For years Microsoft has been promising to rebuild Windows on an OLE foundation, and that strategy has not changed. In many cases Microsoft has simply replaced the word OLE with Active. So, ActiveX controls are OLE controls, OLE automation servers are now Active Servers, and similarly the OLE object model once known as Data Access Objects has become the ActiveX Data Object or ADO. With that in mind, here's a plain English summary of what is in the Active Platform.

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



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

p296 ➢

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

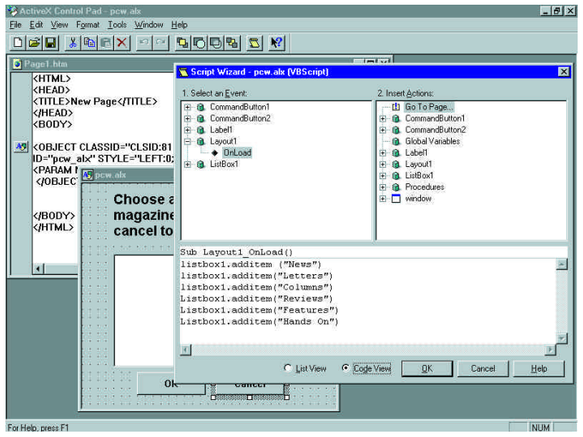
think of them as web page wizards. They are ActiveX controls but are not used at runtime and do not need to be downloaded to the client's computer.

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

The ActiveX Control Pad

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

The control pad does not only handle the placement of controls. Using the script wizard, you can also write code to bring the form to life. The scripting language can be either VB Script or JavaScript, although the two cannot be mixed on one page. In list view, the script wizard will write code for you based on your response to dialogs, or you can choose code view and bang out code in the old-fashioned way.



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

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

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

Will it work?

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

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

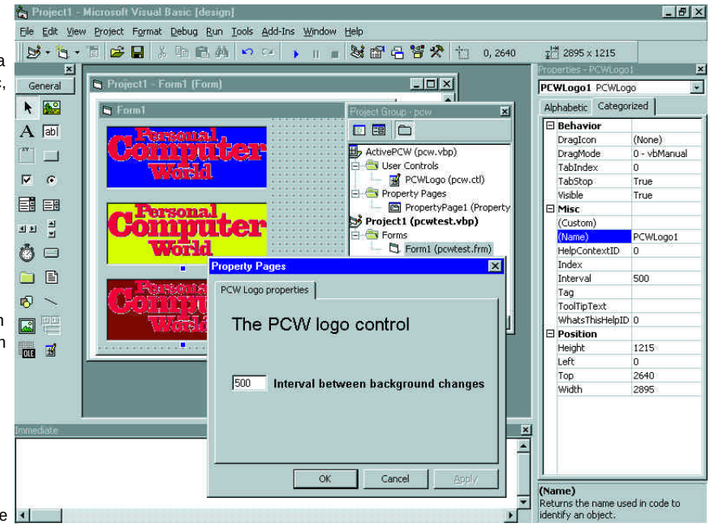
What about Web developers? It is these that Microsoft is courting most visibly, with its "site builder" initiative. But success is far from assured. Microsoft can do what it likes with Windows, but does not own the web. The grave weakness of its Active Platform is that, despite noises to the contrary,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

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

Visual Basic Control Creation Edition

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

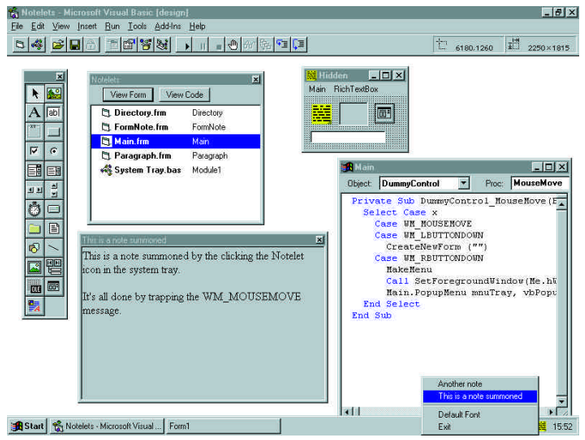
Since version 4.0 Visual Basic has been able to create OLE automation servers. You can declare an object class in a VB project, and then have other applications create objects of that class. Borland's Delphi 2.0 is similarly capable. The one piece missing in both products is the ability to create OLE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



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

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

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



Delphi

Delphi and Windows resources

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

Delphi can use standard Windows resource files (with a .RES extension) and indeed there are occasions when this might be essential: creating a screensaver for Windows NT, for example. It is yet another of those areas which Borland has scarcely bothered to document. Bizarrely, there is documentation in WINAPI.HLP, supplied with Delphi, that covers the Microsoft command-line resource compiler, but not for the Borland resource compiler actually supplied. You didn't know Delphi comes with a resource compiler? It does, and it is the executable called BRCC.EXE or BRCC32.EXE, the 16- and 32-bit versions respectively. The versions called BRC.EXE and BRC32.EXE are shells which are able to call both the resource compiler and the resource linker, RLINK, to bind a resource to an executable — but you do not need to know this since Delphi will do it for you.

To find out how these programs work, run them from a command line without parameters and the options are displayed.

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

p299 ➢

with Borland's C++ products. Using resources in Delphi takes several steps:
1. Create a resource script and compile it to a .RES file.
2. In your Delphi application, include the compiler directive:

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

where MYRES.RES is the name of your resource file. A good place to put it is in the project source below the similar directive {$R *.RES} which Delphi includes by default in all projects. The reason is that the application icon is stored in a .RES file of the same name as the project. It is best not to edit this generated resource file, since Delphi may overwrite your work.
3. Your Delphi code can now load these resources using API functions. Here is a simple example. The following resource script contains a string table with one string:

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

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

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

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

Visual Basic

More about the System Tray

James Talbut writes:

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

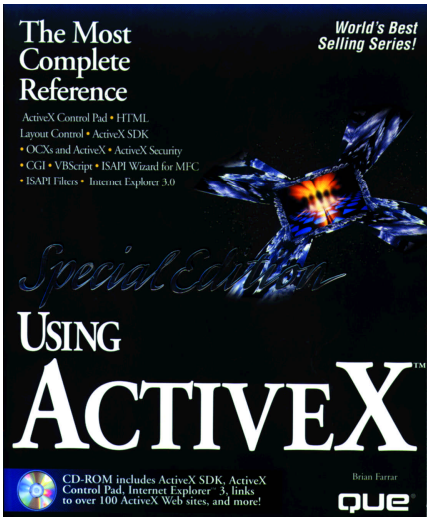
The system tray is controlled by an API call Shell_NotifyIcon, which takes a pointer to a NOTIFYICONDATA record as one of its parameters. This record includes fields for a window handle and a message identifier, the idea being that Windows sends that message to the specified window when
the user clicks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

In C++ or Delphi you would use a custom message handler, but VB does not offer that facility. The workaround is to hijack an existing message handler, and James suggests using a hidden picture box and the WM_MOUSEMOVE message. Then, you can write code in the MouseMove event that will respond to system tray events.

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

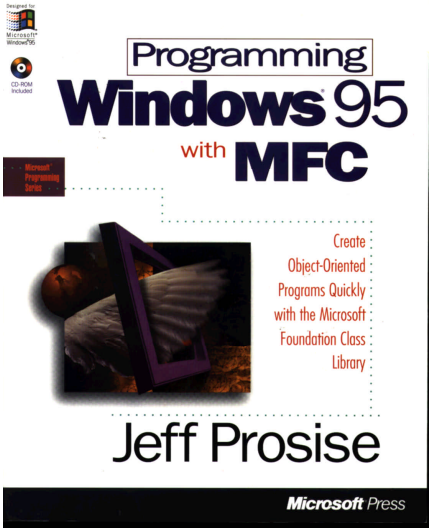
Books for Visual Programming

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



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

Programming Windows 95 with MFC by Jeff Prossie



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

PCW Details

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

Programming Windows 95 with MFC by Jeff Prossie (Microsoft Press), book and CD, £49.95 inc VAT.
Using ActiveX by Brain Farrar (Que), book and CD, £37.99 inc VAT.
Using VB Script by Ron Schwarz and Ibrahim Malluf (Que), book and CD, £46.99 inc VAT.