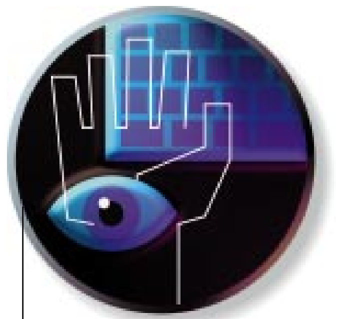


All together now…

We’re all web developers now, says Tim Anderson, who previews Visual J++, deals with Delphi components, lists VB Script limitations, and serves up the Windows 95 system tray.

Should you care about the internet? Over hyped and under-powered, at least for those suffering modem connections, it would be easy to dismiss it as being of little relevance for most developers. Easy, but wrong.

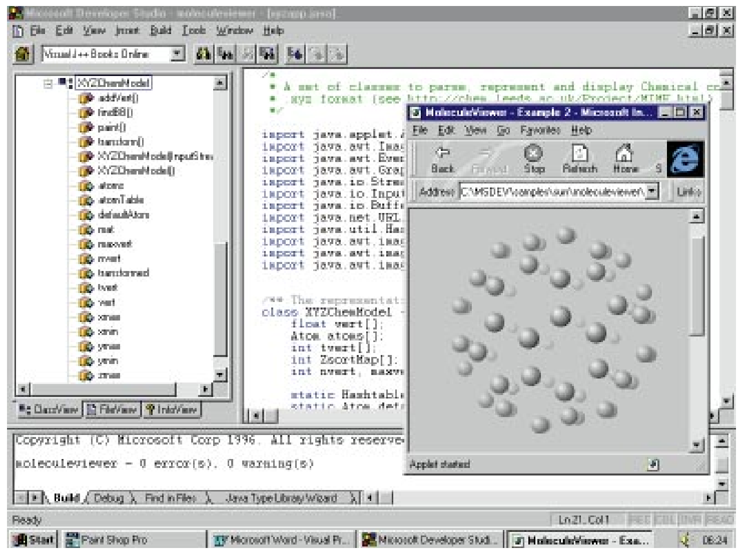
Here are three reasons why, to keep your skills marketable, you have to be web-savvy:

Firstly, HTML is here to stay. It is ironic that Hyper-Text Mark-up Language, designed to add a few simple formatting options for web display, is evolving into the new standard for rich-text documents. The closest previous contender was RTF, or Rich Text Format, used internally by the Windows clipboard. But HTML does forms as well as documents, can host Java applets or, in its Microsoft incarnation, ActiveX controls, and is scriptable with JavaScript or VB Script. It blurs the boundary between document and application. The final consideration is the sheer momentum of cross-industry support. For anyone planning a new software project, an HTML front-end must be a strong contender, particularly for database applications.

Secondly, users like browsers. Maybe the network computer will catch on, or maybe PCs running Windows will remain dominant. Either way, the browser is going to be the primary user interface. Once users discover they can manage files, run applications, get help and surf the web, all from the comfort of their browser, they will be reluctant to learn other kinds of interface. For developers, that means creating applications which work well in that context.

Thirdly, networks are intranets. Company intranets solve a lot of problems. Publishing documents, email, and database access are all comfortably handled by an intranet. If you buy a new server operating system, you are likely to find web-server software bundled with it, just as workstation software comes with a browser pre-installed. It is irresistible and companies that have not yet done so will inevitably install intranets alongside their Notes, Exchange, client-server or any other systems. Anyone developing software for use over a network should make it intranet-friendly.

So there are good reasons why software companies are falling over themselves to produce web software, and why you will see increasing coverage of web software tools in this column. The catch is that chaos is heading our way, with wars over standards, languages, objects and web servers. But we are all web developers now.



```
Copyright (C) Microsoft Corp 1996. All rights reserve
moleculeviewer - 0 error(s), 0 warning(s)
```

Visual J++ looks just like its C++ partner and creates cross-platform Java applications

Visual J++ preview

A late beta of Microsoft’s Java development tool has arrived in time for a brief preview. The package is hosted by the same Developer Studio used by Visual C++ and shares some of its tools. If you are comfortable with the Visual C++ environment, you will find the transition to Java easy. Each project can be viewed in a hierarchical class view, or file by file, and online documentation is fully integrated.

A clever touch is the resource wizard which converts compiled resources into equivalent Java code. There is still the problem, inherent to Java, that the Abstract Window Toolkit (AWT) Java class library does not support the range of controls available under Windows. Java projects can be either applets, hosted by a browser, or standalone applications which are executed from the command line using the supplied JVIEW tool.

Microsoft is licensed to produce the 32-bit Windows reference version of the Java virtual machine. The key point of interest is the integration between Java and COM, the object model behind OLE. This enables you to treat Java applets as COM objects and vice-versa. Visual J++ has a type library wizard which creates Java .CLASS files as an interface to ActiveX controls or OLE servers.

You can also expose Java interfaces as COM interfaces and use a tool called JavaReg to register the Java class as an OLE object. This means the Java class becomes accessible to OLE clients like Visual Basic or Delphi, as well as in web applications. Another interesting point is that when a Java applet is running in Internet Explorer, all its public methods and variables automatically become available to Visual Basic Script or JavaScript for scripting, as if the Java applet were an ActiveX control. The snag is that this integration only workd on platforms which implement COM, which essentially means Windows.

Visual J++ is a good Java development tool, whether or not you want the OLE features. Like Visual C++, it is not a visual environment in the same way as Visual Basic, Delphi or Optima. Borland’s Latte promises something more along those lines. As proof that Microsoft is serious about Java, though, it is more than enough.

Visual Basic

Visual Basic and the System Tray

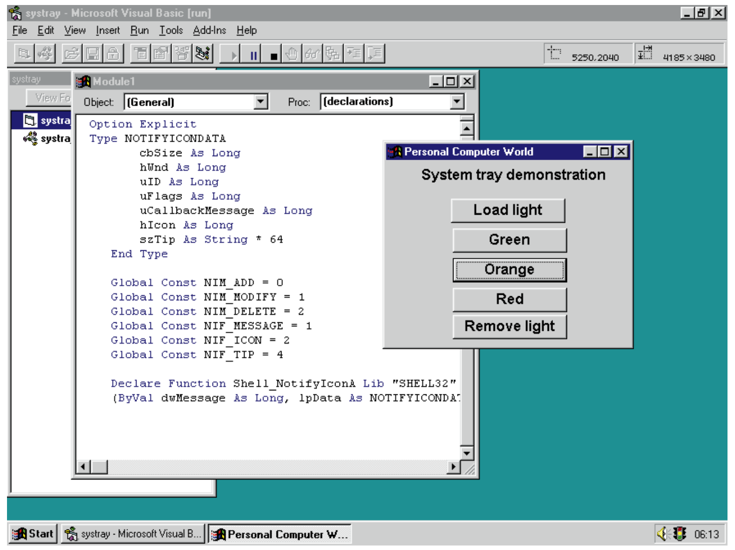
Sagar Shah writes: “*I recently moved up to Visual Basic Pro 4.0 (32-bit). While creating some small utility applications, I ran into a problem with the system tray on the taskbar. I cannot find any entries in the manual which tell me how to add to the system tray. I found the function Shell_NotifyIcon in the WIN32APTI.TXT file, and using this function I can add and delete a blank space but nothing more.*”

The system tray is a corner of the taskbar reserved for status display, in Windows 95 or NT 4.0. It is also called the taskbar notification area. Typically, a utility installs itself as an icon in the system tray, which automatically updates. For example, if you install a modem, a modem icon appears in the system tray when you go online and flashes when data is sent or received. When the mouse is over the icon, a tooltip shows more detailed information, in this case the number of bytes received. Double-clicking opens a dialogue of further options.

All this is done through the Shell_NotifyIcon function. The declaration in VB is:

```
Declare Function Shell_NotifyIconA
Lib “SHELL32” _
(ByVal dwMessage As Long,  lpData
As NOTIFYICONDATA) As Integer
```

The dwMessage parameter is one of three constants which tell the system to add, delete or modify an icon in the system tray. The second parameter points to a record type which can include an icon handle, a string for the tooltip, a window handle and an application-defined message identifier. The idea is that you define a message handler in your application which responds when the user clicks the icon. The message parameters indicate what type of



```
Option Explicit
Type NOTIFYICONDATA
        cbSize As Long
        hWnd As Long
        uID As Long
        uFlags As Long
        uCallbackMessage As Long
        hIcon As Long
        szTip As String * 64
    End Type

    Global Const NIM_ADD = 0
    Global Const NIM_MODIFY = 1
    Global Const NIM_DELETE = 2
    Global Const NIF_MESSAGE = 1
    Global Const NIF_ICON = 2
    Global Const NIF_TIP = 4

    Declare Function Shell_NotifyIconA Lib "SHELL32"
    (ByVal dwMessage As Long, lpData As NOTIFYICONDAT
```

systray - Microsoft Visual Basic [run]
Personal Computer World
System tray demonstration
Load light
Green
Orange
Red
Remove light

VB can easily control icons in the system tray, but responding to mouse clicks is more difficult

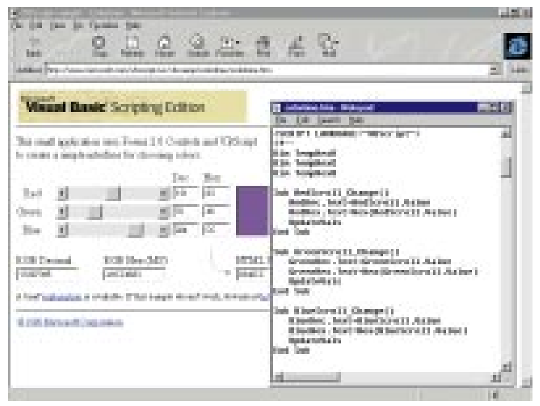
p310 ➢

Visual Basic Script — what it does not do

Now that Microsoft Explorer 3.0 is around, VB Script has become useful, particularly on an intranet where you can ensure compatible browsers. Microsoft has also clarified its limitations, some of which are for security reasons while others are merely shortcomings of the language. Here are some of the things VB script cannot do:

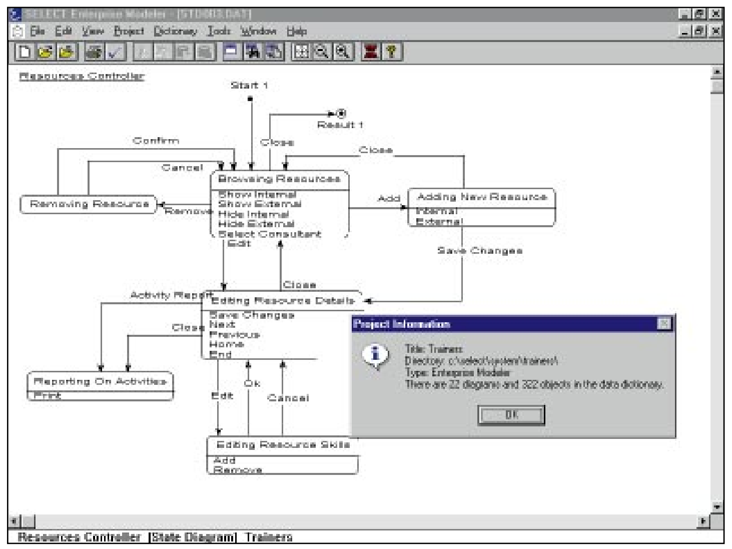
- No data access: data access from a web page needs either CGI scripting or special features of particular web servers.
- No debugging: you cannot even step through code.
- No control arrays.
- No classes.
- No OLE automation except the OLE interface to Internet Explorer itself.
- No file operations.
- No types, other than variants.
- No access to system objects like the printer or the clipboard.

Many of these limitations can be overcome by using ActiveX controls, which have unrestricted access to the system. A rogue ActiveX could cause lots of problems, which is why system administrators are watching nervously to see if the digital signature scheme for verifying ActiveX controls is successful in preventing viruses.



Visual Basic Script is fine for scripting, but with major limitations

SELECT for Visual Basic Enterprise: not for the faint-hearted



Title: Trainers
Directory: c:\select\system\trainers\
Type: Enterprise Modeler
There are 22 diagrams and 322 objects in the data dictionary.

mouse event has occurred. Your application window can be hidden so that it only pops up when needed.

Unfortunately, not all this functionality is available from Visual Basic. You can easily make an icon appear and set the text for the tooltip. But since VB has no way to intercept custom messages, you cannot make a dialogue appear when the user clicks the icon.

The way round this would be to use a control like MessageBlaster that adds this feature to Visual Basic. If you are content with more limited features, you can easily create an application like the example on our cover-mounted CD.

Note that you need to include an icon handle in the NOTIFYICONDATA record. You can obtain this in several ways: one is to use the icon or picture of a form or control; another is the LoadIcon API function; or there is the LoadResPicture function which works on icons stored in a resource file. The technique used in the example is to call LoadPicture to place an icon into an invisible image control, and then use its picture property to obtain an icon handle.

SELECT for Visual Basic

Bridging the gap between those who theorise about business object models and actual working systems is no trivial matter. SELECT is a set of tools based on object models developed by J Rumbaugh and Ivor Jacobson. There is a modelling tool, from which you can generate both SQL code and a set of Visual Basic forms and classes. There is also an automatic documentation feature that works with MS Word. SELECT needs the VB Enterprise edition, while other versions work with Forte and C++.

Unlike other tools which you can pick up and drop as required, committing to SELECT is almost a way of life. It replaces the free-and-easy VB style with a rigorous development process.

The best advice to those considering a system such as this is a careful evaluation procedure including full consultation with others actually using the system.

DELPHI

Using components in Delphi

When it comes to components, Delphi users have a difficult decision. The most obvious solution is to use VBX controls in Delphi 1.0, or OCX/ActiveX in Delphi 2.0. These component types are abundant, mainly thanks to the popularity of Visual Basic, and now that ActiveX plays a key role in Microsoft's internet strategy, you can expect them to proliferate.

Often, the component you want is only available in VBX or ActiveX form. The second advantage of these Microsoft standards is that they can be hosted by several different tools. For example, a VBX can be used by Delphi, Visual Basic and Visual C++, provided you use the 16-bit versions.

Unfortunately, these benefits are balanced by several problems, one of which is compatibility. Some VBX vendors assume that their controls will be hosted by Visual Basic and that not all their features work in Delphi. In particular, data-bound VBXs lose their data-aware functions in Delphi, if they work at all. Another problem is getting at certain VBX properties, such as string properties which Delphi converts to Pascal string types, cheerfully truncating them if needs be. OCX controls present a different set of problems, but since there are several different levels of OCX compliance, compatibility is by no means guaranteed. Finally, you cannot easily build either VBX or OCX components in Delphi. Borland has considered providing an OCX wizard to convert Delphi native components, but so far it has not emerged.

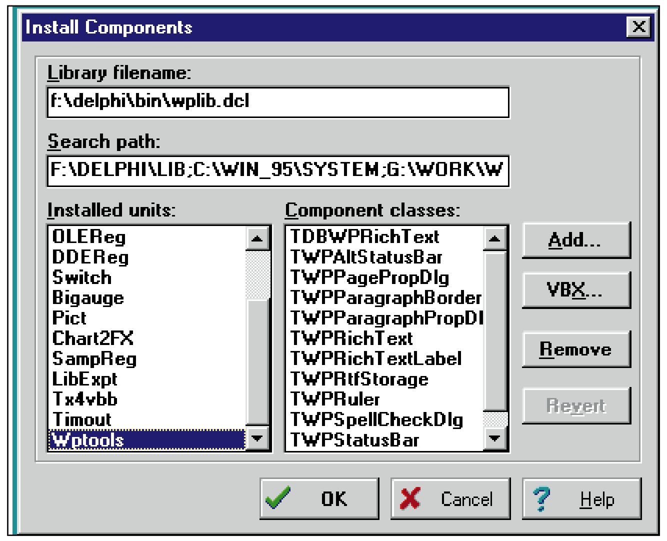
To avoid the problems with VBX and ActiveX you can extend Delphi's Visual Component Library either by coding your own custom components or obtaining add-ons from a third party. Coding your own is fairly easy, although harder than straight Delphi programming. The advantage of native components is full compatibility and the most efficient interface between the component and the rest of your application, and therefore the best performance. The disadvantages are that the component can only be used in Delphi, and if you have several applications using the same component, inefficient use of disk space because the whole component is compiled into every executable you create. If this becomes a major problem, you can move parts of the code into dynamic linked libraries so it is shared between applications.

Frankly, in most projects a VCL component is preferable where available. Availability is the problem, but the situation is improving. Delphi 2.0 includes the QuickReport component which is vastly more efficient than wheeling in ReportSmith.

Crystal Reports 5.0 also comes with a VCL, although the huge Crystal DLLs do not compare with QuickReport for efficiency. Other leading component vendors have been slow to support Delphi but may find themselves losing sales to smaller upstarts as a result. A good example is WPTools, a VCL-implementation of a rich text edit control. This one works in 16-bit Delphi as well as Delphi 2.0 and is not just a wrapper for the Windows 95 common RTF control: it supports large documents, fonts, styles, images and hypertext links, and full source is available to registered customers. Look out for a full assessment in a future article.

Delphi 2.0 16-bit?

Borland is considering an update to 16-bit Delphi, and is using the web to survey developers about what features they would like and whether they would buy it. TI would prefer to see resources go to developing Delphi for 32-bit Windows. There is still a big market for 16-bit development, but I sense that the tide has turned. Companies have both NT 4.0 and Windows 95 from which to choose and web developments favour 32-bit Windows. It is too late for 16-bit Delphi 2.0, and Borland risks getting it wrong again.



Library filename:
f:\delphi\bin\wplib.dcl

Search path:
F:\DELPHI\LIB;C:\WIN_95\SYSTEM;G:\WORK\W

Installed units:	Component classes:
OLEReg	TDBWPRichText
DDEReg	TWPAltStatusBar
Switch	TWPPagePropDlg
Bigauge	TWPParagraphBorder
Pict	TWPParagraphPropDl
Chart2FX	TWPRichText
SampReg	TWPRichTextLabel
LibExpt	TWPRtfStorage
Tx4vbb	TWPRuler
Timout	TWPSpellCheckDlg
Wptools	TWPStatusBar

Delphi components are installed by rebuilding the component library. But which component type is best?

Delphi Component Options

Component type	Pros	Cons
VBX, ActiveX, OCX	• Widely available •Shared between applications	• Not always compatible • Cannot easily create in Delphi • Version control problems
VCL	• Best performance • Can create or customise in Delphi •No version control problems	• May not be available • Not shared between applications

p313 ➢

A book for visual programming

■ *Rapid Development* by Steve McConnell

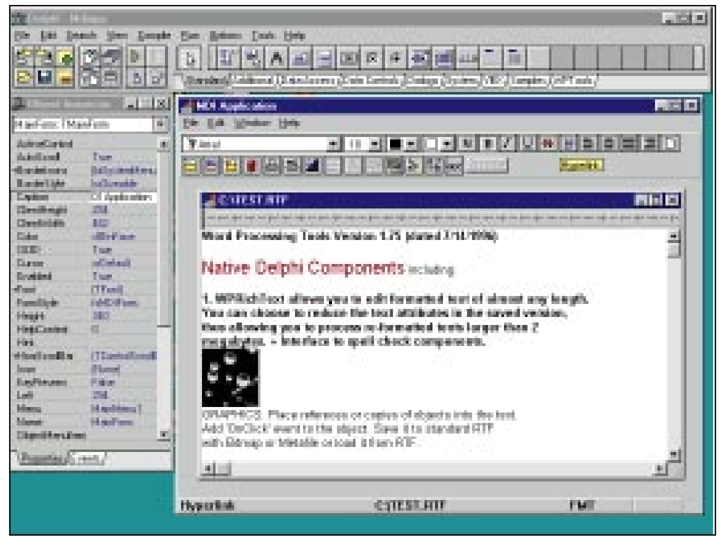
Software projects are notorious for going wrong. The concept of rapid application development seems to offer a solution by using tools that dramatically cut programming time, but there are still plenty of problems.

Whereas his previous book, *Code Complete*, studied the detail of coding and debugging an application, this one analyses the whole development process, exploring common reasons for failure and offering tips for a successful strategy.

Really, the book is mis-titled. It is not only about rapid development, but any kind of software project. If your emerging application is suffering from feature-creep, unrealistic schedules, tools that do not work, problems integrating work from different team members, heavy overtime and low morale, then McConnell will tell you why, how to rescue the project, and how to avoid repeating history next time around.

Although the author does not focus on specific languages, there are some good observations about visual programming. The key advantage of products like Visual Basic and Delphi is reduced coding time, but there are associated risks. The main ones are over-estimated productivity savings, failure to scale well as the project expands, and the encouragement of sloppy programming. The answer is to be aware of the limitations of a particular tool and to allow time for working around them. Another key point is that the larger the project, the less time is spent on coding as opposed to other elements like designing and testing. Therefore, visual tools yield their biggest benefits on small projects.

There are plenty of case studies and examples in the book, although it is a shame that most are invented rather than real-life projects. It is repetitive in places and at times dispiriting, as the author tends to focus on how *not* to do things. But this is excellent reading for developers, especially those who work in a team.



Word Processing Tools Version 1.75 (dated 7/11/1996)

1. WPRichText allows you to edit formatted text of almost any length. You can choose to reduce the text attributes in the saved version, thus allowing you to process re-formatted texts larger than 2 megabytes. - Interface to spell check components.

GRAPHICS: Place references or copies of objects into the text. Add 'OnClick' event to the object. Save it to standard RTF with Bitmap or Metafile or load it from RTF.

A third-party has produced a full-featured rich text control as a native VCL component

•PCW Contacts

Tim Anderson welcomes your Visual Programming comments and tips. He can be contacted at the usual *PCW* address or as **freer@cix.compulink.co.uk**

Visual J++ (price not yet announced) from Microsoft 0345 002000

WP Tools is shareware from Julian Ziersch, 100744.2101@compuserve.com

SELECT VB Enterprise Edition £2,995. SELECT Software 01242 229700

Rapid Development by Steve McConnell, (Microsoft Press). £32.49 from Computer Manuals 0121 706 6000