Brushing up the VB image

Where next for Visual Basic? **Tim Anderson** investigates VB Script and the likely features of Visual Basic 5.0, looks at a strange performance test, and tries out new balloons and buttons.

VB 4.0 may have some great new features, but many developers have found it slow and unwieldy thanks to its huge runtime files. Others have jumped ship to Borland's Delphi. On another front, Sun's Java looks set to be an Internet standard and perhaps popular for general development as well. Despite VB's huge installed base, the pressure is on Microsoft to restore its tarnished image. The company is responding, first with a cut-down VB for Internet scripting, and second with a brand new version of the main product. I spoke to Jon Roskil, Microsoft's director of marketing for Visual Basic, about the new developments.

"VB Script is a semantic subset of the Visual Basic for Applications language, but without file I/O. That makes the language safe, sand-boxed like Java. It will be part of Internet Explorer. You will have embedded VB script in HTML on a Web site, and the VB routine will execute on the browser. For example, you could do data-field validation on a Web form." Web solutions need to be cross-platform, and Microsoft aims to achieve this with third-party help. "Microsoft VB Script will be for all Windows platforms, 16 and 32-bit, and for the Mac. Other vendors will supply

Fig 1 Delphi vs VB — is it really faster?

On Microsoft's Web site are two documents prepared by the "Carnegie Technology Group", which benchmarks the performance of Visual Basic 4.0 against its most obvious competitors. One compares VB 4.0 (32-bit) with Delphi 1.0 and Oracle's Power Objects. The report's remarkable conclusion is that: "Visual Basic and Delphi are very closely matched in the language performance." For example, one of the tests measured loop performance, and VB was found to execute 1,000,000 loops in about 1020ms, while Delphi took 1330ms.

The result is so surprising that I set up a similar test on a similar PC. Carnegie Technology explains that the loop looks like this; where α is the number of loops:

```
i = 0
t0 = GetTickCount()
do while (i< α)
i = i + 1
loop
t1 = GetTickCount()
total = t1-t0
```

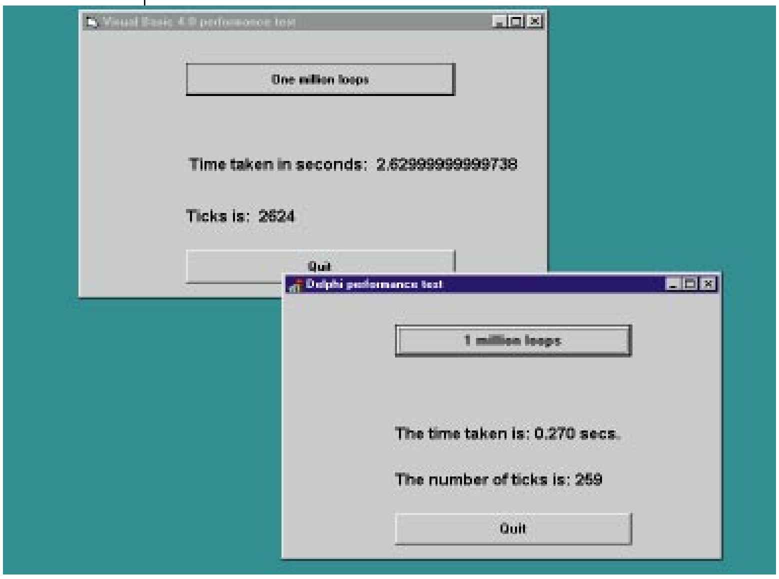
The result of our test was that VB took 2568ms to execute one million loops, while Delphi took 261. In other words, Delphi was nearly ten times faster, exactly what one would expect. This loop test is vulnerable to a clever compiler avoiding the loop by simply setting

```
i =α
```

and not bothering with the loop, although neither Delphi nor VB seem to do this. Traditionally, a loop to filter out prime numbers is preferred, since this can only be done by iteration. A routine to find all prime numbers between 1 and 32000 took VB 1934ms, against Delphi's 560.

After these brief experiments, I still find Carnegie Technology's report surprising. The experience of most developers is that Delphi's compiler yields a very substantial speed advantage.

In our tests, Delphi proved about ten times faster than VB on simple loops



Time taken in seconds: 2.62999999999738

Ticks is: 2624

The time taken is: 0.270 secs.

The number of ticks is: 259

implementations for further platforms. There will be a C source code reference as a free download on the Internet." Since VB is interpreted, the scripts will run on any platform for which runtime files exist. As with Java, this is a key advantage for Internet development. VB Script should be available by the time you read this.

As for VB 5.0, Jon expects it to be released before the end of 1996. It is further enhanced for OLE development. "We can create OLE controls in the next version of VB." A lot of this technology is already there in version 4.0, which can create in-process OLE servers, but the missing piece is an event-layer interface which goes on top. Naturally, Jon sees these OLE objects having a role on the Internet. "VB objects can be called from HTML. We have some pieces called 'shims' in Internet Explorer, which let HTML talk to ODBC or OLE. Internet Explorer will also host OLE controls and OLE document objects."

Speed is the key

VB 5.0 is rumoured to include the long-awaited compiler. While he will not confirm this, Jon admits that this is "on our wish-list, and very near the top. But a compiler is no panacea for performance issues. In client-server applications, data access speed is the key. And the VB language engine is fast. There are benchmarks available on our Web site which show VB against Delphi, PowerBuilder and Power Objects. The VB language engine is ahead on four out of nine tests." So why have developers found VB 4.0 slow? "We optimised VB for the 32-bit platform, not 16-bit, although it can create 16-bit applications." A tacit admission that VB 4.0 16-bit *is* too slow.

VB standalone, VBA for Microsoft Office, and now VB for the Internet. It sounds promising, except that most Internet users browse with NetScape, and NetScape is by no means certain to support VB Script. Performance is another concern. VB is certainly fast as interpreted languages go, as long as there is enough RAM to handle the runtime load. But despite Microsoft's optimistic performance tests (*see Fig 1*), it is nowhere near the speed of Delphi or C++. Maybe the promised compiler will close the gap. Otherwise, it's hard to imagine VB succeeding as a tool for developing OLE controls.

Getting Resourceful

Visual Basic 4.0 supports standard Windows resource files. These contain strings, bitmaps and other data, and are particularly useful for international projects. By changing the resource file, for example, you could display text in French instead of English. Resource files can improve form load times, since the data is not loaded into memory until required by your application. Since the resource files are standard, they may also be useful if you decide to port your application to another environment.

Resource files begin as scripts (.RC extension) and are compiled as 16 or 32-bit binary files (.RES). VB can only use the compiled type, and these must be 16 or 32-bit according to the version of VB 4.0 used. There's no resource compiler supplied with VB, but there is one supplied with most C++ packages, including Visual C++. Here's a simple example:

1. Run Visual C++ 4.0 and choose File — New — Resource script.
2. On the Insert menu, choose Resource, and select String Table.
3. In the grid which appears, double-click the top row. Enter an ID of HELLO_STRING and a caption, "Hello", and close the dialogue. On the next row, enter an ID of GOODBYE_STRING and a caption, "Goodbye".
4. Use Save As to save the resource. A good tip is to use a subdirectory of your VB project. Save it first as a resource script, and then as a compiled resource called VBENG.RES.
5. Now amend the captions to "Bonjour" and "Au revoir". Use Save As to save the amended resource to a new directory, and call it VBFR.RES.
6. Create a suitable VB project. This example just displays a label and a button to exit the application. Use Add File to add VBENG.RES to the project.
7. Add Const definitions to define the IDs used in the resource file. For example:

```
Const HELLO_STRING = 1
```

If you open the RESOURCE.H file created by Visual C++, you will find #Defines for each ID. You can use this as the basis for the VB Const definitions.

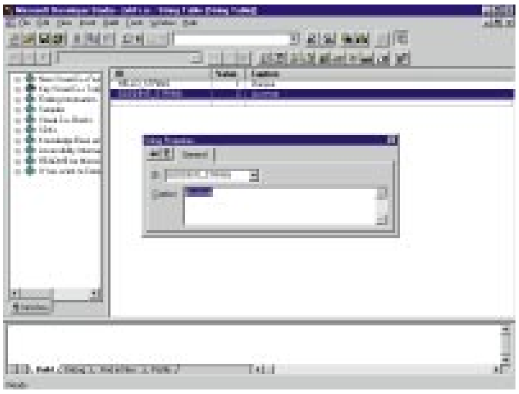
8. Now you can use the VB function LoadResString to access the resource file. For example:

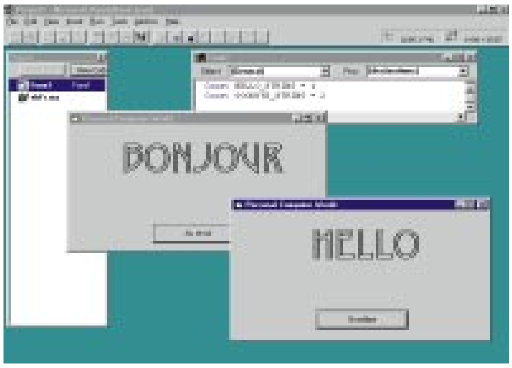
```
Label1.Caption =
LoadResString(HELLO_STRING)
```

To change the language of your application, remove VBENG.RES from the project and replace it with VBFR.RES. When you compile an executable, the .RES file will be bound into it, so it does not need to be distributed.

Finally, do not include an icon resource with an ID of 1, as this is reserved by VB for the application icon.

Resourceful





Top *There's no resource editor supplied with VB 4. In this example, Microsoft Visual C++ is used to create and compile the script*

Above *It's all done with resources — a multi-lingual VB application*

Balloons and buttons

The standard Windows button is square, drab and grey. No more, if Farpoint's ButtonMaker catches on. Supplied as 16 and 32-bit VBX, OCX and DLL, this button control has a border divided into user-definable segments so that an enormous variety of shapes and styles can be defined. You can also tweak the "grey area", by which FarPoint means the area within the control but outside the bit that gets clicked. Here you can place colours, patterns, pictures or even animation to bring your buttons to life.

If buttons are not enough, the ButtonMaker package includes a balloon control. Balloons are the ultimate tool-tip, and although not quite as flexible as buttons in shape, they can take on a variety of forms, including a fluffy thought-cloud.

Nice idea; but so what? I guess there may be occasions when a ButtonMaker button is just the thing to make an interface more attractive or intuitive, but many of the supplied examples merely look silly. They are also disappointingly slow to load, especially in OCX guise. Finally, it would be nice to see component vendors recognising the existence of Delphi by supplying VCL wrappers to use with their products.

Moving to Visual Basic 4.0

Alan Knox asks:

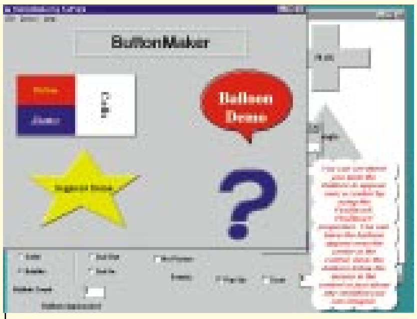
"I have been trying to use your tip on how to make a window appear on top of all others (PCW November 1995) and can't get it to work. I am using VB 4 but I wouldn't have thought this would make any difference. The declarations are in a Global.bas file and I set the topmost flag in the load section of the form I want to appear on top. Any ideas?"

Fig 2 Directives and declarations

```
#If Win16 Then
Declare Sub SetWindowPos Lib “User” (ByVal hWnd As Integer, ByVal hWndInsertAfter As Integer,
ByVal X As Integer, ByVal Y As Integer, ByVal cx As Integer, ByVal cy As Integer, ByVal wFlags
As Integer)
#Else
Declare Sub SetWindowPos Lib “User32” (ByVal hWnd As Long, ByVal hWndInsertAfter As Long, ByVal
X As Long, ByVal Y As Long, ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long)
#End If
```

A window can be permanently set on top by calling SetWindowPos. The function declaration printed in November's issue was for Visual Basic 3.0, but should work the same way in 16-bit VB 4.0. But the 32-bit version needs changes to API declarations. For a start, they are located in different libraries such as USER32.DLL rather than USER.DLL. Next, integer parameters generally change to long. Third, Win32 has case-sensitive function names. A good tip is to use the API text viewer to find the right declaration. Next, use compiler directives to bracket the declarations so VB sees the right one *(Fig 2)*.

In fact, Alan may have done this and still seen problems. There is a bug in VB 4.0 which causes the topmost setting to be lost when you task-switch to another application. The good news is that this only occurs in the development environment, and not in VB apps compiled to an .EXE.



A button for every occasion, with FarPoint's ButtonMaker control

Visual Components on the Net

The explosive growth of the World Wide Web means keen interest in HTML authoring tools. Visual Components, which supplies the popular Visual Tools Suite, has announced Formula One/NET which lets you embed a spreadsheet component into an HTML document. Users who have NetScape Navigator 2.0 along with the Formula One/Net add-on can use the worksheets interactively, entering new data and performing calculations.

At the time of writing, the product is not available. The announcement is interesting, however, since the concept of a visual Web component is a natural extension of the visual development model. There are numerous problems: connection of a Web worksheet to a backend database is not trivial, for example. Also, full use of Formula One/NET requires the Windows version of NetScape. Although handicapped by the current battle to establish Internet standards, visual components for the Web look likely to be the next big growth area in visual programming.

Visual Components Inc. has become a subsidiary of Sybase, putting the company into the same family as PowerBuilder, Watcom, and the Sybase database server products.

PCW Contacts

Contact **Tim Anderson** with your comments, queries and suggestions, either at the usual *PCW* address or email **freer@cix.compulink.co.uk**

Visual Components Europe
01892 834343

FarPoint's ButtonMaker costs £75 from **Contemporary Software** 01727 811999