Not just a **pretty face**

...that's Visual Developers Suite. **Tim Anderson checks it out, answers Delphi queries and solves a common Visual Basic problem.**

I do have misgivings about the hundreds of VBX and OCX controls on the market. It's not that they are no good: many are excellent and enable you to create a database manager, a web browser or a word processor in less time than it takes a C programmer to create a single "Hello world" window.

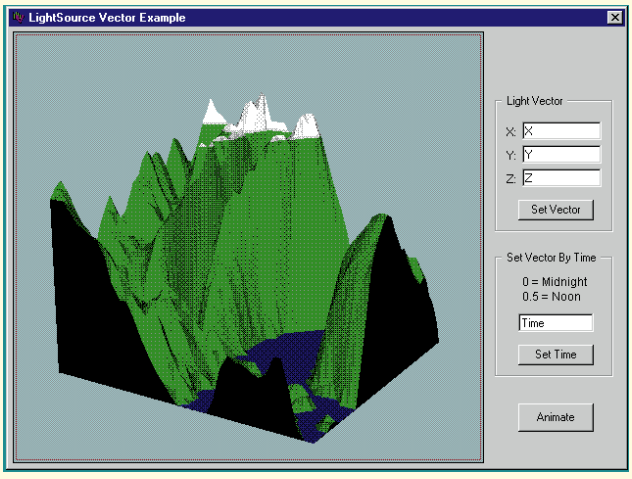
The problem is that every time you pop another component onto a Visual Basic form, your application grows more bloated and performance suffers. Canny developers will ask themselves, "Do I really need this control?" before committing to yet another OCX.

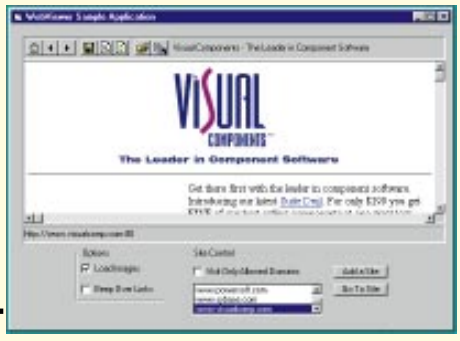
Farpoint's Tab Pro, now at version 2.0, is a case in point. Tabbed dialogues have become important in creating a clear, intuitive user interface and Tab Pro offers more than VB 4.0's native tab strip. It is supplied in every combination of 16-bit and 32-bit VBX, OCX and DLL, for use with virtually any Windows development tool.

The tab control is a container, unlike VB's tab strip which must be used in conjunction with another container like a picture box. Then there are more than 250 functions along with nearly as many properties which control the appearance of the tabs and which can look like a ring binder as well as a conventional tabbed dialogue. Tabs can be bound to a database to achieve a neat card-index style interface.

If you want a tabbed dialogue to look like a ring binder, Tab Pro is the obvious choice

Tab Pro is only around 300K and is well documented in two smart manuals. But do you really need it? Something like the Visual Developers' Suite, from Visual Components, makes better sense. No-one could accuse these controls of being merely decorative. Instead, this package includes five 16-bit and 32-bit OCX components, each of which is virtually a complete application in itself. There is the Formula One spreadsheet, First Impression charting control, Visual Speller, Visual Writer word processor control and, new in this version, WebViewer HTML control. The three most complex controls, for spreadsheet, charting and word processing, are among the best in their category.





First Impression, **(above)** *part of the Visual Developers Suite Deal, is an impressive charting component. WebViewer,* **(right)** *is new to the Visual Developers Suite but Microsoft's Internet Control Pack offers better functionality*

There are hesitations. Visual Writer is no longer indispensable now that a rich text control is part of 32-bit Windows and the WebViewer faces tough competition from Microsoft's freely-distributed Internet Control Pack. It is a shame that Visual Components has chosen to implement 16-bit OCXs, rather than the more widely supported VBX, for its 16-bit controls. But taken as a whole, the suite is excellent value and while these components will slow down your application, they also provide functionality that a VB developer could otherwise only dream about.

Do we still need Delphi?

Guy Robinson comments: "*As soon as Microsoft releases a compiler version of Visual Basic, Borland will have lost most of the advantage that Delphi currently possesses.*

With Microsoft controlling the operating system as well, Borland must ultimately lose the advantage. I found it surprising that Borland's Zack Urlocker (quoted in PCW*, May) wasn't more positive towards a cross-platform Delphi. Or is Java the company's intended cross-platform vehicle?*

I am an OS2 user, and if you talk to Mr Urlocker again you can tell him I would be one of the first to purchase Delphi for OS2 if it became available."

If you think cross-platform compatibility is important, and the rise of the Internet suggests that it is, then Java must be a more promising way forward for Borland than simply releasing versions of Delphi for other platforms. But I do not see Delphi being seriously threatened by a compiled Visual Basic. It is not just a matter of performance, it is the design and structure of Delphi that is richer and more elegant than Visual Basic. Another advantage is that Delphi is equally suitable for small utilities or major applications. VB has its own strengths, and a compiled version should address the performance issue, but Delphi will not lose its niche.

Combo Box defaults

Brendan Breen asks: "*I have a ComboBox in a dialogue. I want to set its style to DropDownList. When the dialogue is displayed I want to show a default value in the combo. But it always appears blank. I have tried all sorts of things but none of them work. Any ideas?*"

Writing to the Text or SelText property of a combobox does not work when its style is "csDropDownList". The solution is to write to the ItemIndex property. For example, you could put this into the dialogue's Show event:

```
ComboBox1.ItemIndex := 3;
```

Delphi 2.0 and OLE

Peter Harris queries Delphi's OLE capabilities: "*We are currently developing software using Borland's C++ (4.52) and OWL to develop a graphical interface to a specialised database. We were very interested in developing the front end in Delphi and purchased V2.0. However, we seem to have come across a fairly major limitation of Delphi2, in that it will not act as an OLE2 server and client. We need to be able to embed bitmaps and suchlike in our application windows, and also allow linking/embedding of our graphical stats results into other apps — specifically, word processors. We've been unable to find anything in the Delphi documentation about this. I wonder if you have come across this problem, or a way around it?*"

Delphi 2.0 will act as an OLE 2.0 server and client, but the server bits do not yet support embedding, at least not as implemented in the visual component library. On the client side, there is the TOleContainer which is to be found on the System tab of the component palette. For documentation, placing one of these on a form and then pressing F1 brings up all that Borland has seen fit to provide.

For example, you could create a link to a bitmap file with the following line of code:

Automating Delphi with OLE

Step by step, here's how to create a OLE automation server in Delphi:

1. Start a new application or DLL and save it as, for example, MYAPP.DPR. DLLs are in-process servers that run in the same address space as the calling application.

2. From Delphi's file menu, choose New and select Automation Object from the dialogue.

3. Enter a class name, for example MyObj. By default, Delphi will make the OLE class name the same as the application name, so the new OLE object will be MyApp.MyObj.

4. Choose the instancing. Internal is a rarely-used setting for OLE objects that are not available to other applications. Single instancing means that each instance of the server can only export one instance of the OLE object. Multiple instancing, which is required for DLLs, allows multiple instances of the OLE object.

5. Add OLEAuto to the uses clause in the project source. If it is a DLL, follow it with this section, observing case sensitivity:

```
exports
DllGetClassObject, DllCanUnloadNow,
DllRegisterServer,
DllUnregisterServer;
```

This is all you need. You don't need to add OLE objects and methods to the exports clause. Contrary to the documentation, you don't need to call Automation.ServerRegistration in the project source.

6. In the Automated section of the new OLE object, add the methods, properties and functions that are to be exposed, defining them in the implementation section in the normal way. There are limitations in terms of which types and declarations are allowed and normally these will be caught by the compiler if you try to use them. Here's an example type declaration:

```
type
  MyObj = class(TAutoObject)
  private
    { Private declarations }
    MyVar: integer;
    function GetMyProp: integer;
    procedure SetMyProp(iParm:
integer);
  automated
    { Automated declarations }
    function MyMethod(iParm:
integer): integer;
    property MyProp: integer read
GetMyProp write SetMyProp;
  end;
```

Note that you cannot access fields directly in an OLE object. You have to use property access methods.

7. Finally, the OLE automation object must be registered. Applications can be registered by running them with a /regserver parameter. You can register a DLL using Microsoft's REGSVR32.EXE utility, or failing that by calling the exported DllRegisterServer function. This need only be done once.

You can easily test the OLE object. Here is some example Visual Basic code:

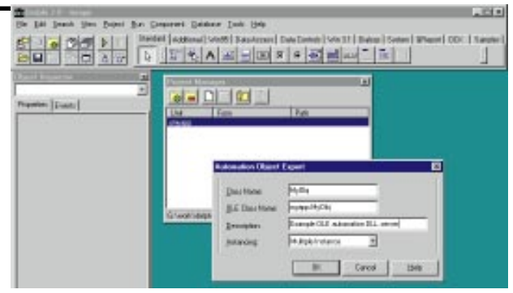
```
Dim myOLEobj As Object
Set myOLEobj =
CreateObject("myapp.myobj")
myOLEobj.myprop = 345
MsgBox "The property was set to: " &
str$(myOLEobj.myprop)
```

Why use OLE automation servers?

Performance of OLE servers is good, particularly in-process servers, but they are not as quick as standard DLLs. So why bother?

Firstly, because programming OLE objects is easy and intuitive, compared to ordinary DLLs which require case-sensitive function declarations.

Secondly, OLE objects bring with them the benefits of object-orientation, inheritance aside. Thirdly, OLE objects have a greater degree of language independence. Fourthly, as OLE progresses it should be possible to do things like remote automation using the objects you have developed.



Delphi's Automation Object Expert lets you enter the key characteristics of a new OLE automation object. Later, you have to register the object in the Windows registry

Visual Basic tip: detecting the IDE

How can you detect whether your Visual Basic application is running in the development environment or standalone? For example, you might want to enable extra debugging code when running in the IDE. Here are two ways to do it. The easy way is to use inspect App.Exename. App is a global object with several useful properties. For example, App.Path returns the directory from which the executable is run. App.Exename returns the name of the project, when running in the IDE, or the name of the executable when running standalone. If you give the project and the executable different names, then hey presto! you have an easy way to detect which is running.

If you would rather show off your Windows API skills, there is another method to use. All VB applications have a hidden parent window. In the IDE, this has a window class of ThunderMain, but in standalone executables the class is ThunderRTMain. Here's a function that exploits this difference to detect which is running:

```
' Declarations for 16-bit VB - amend for 32-bit.
Declare Function GetClassName% Lib "User" (ByVal hwnd%, ByVal lpClassName$,
ByVal nMaxCount%)
Declare Function GetWindowWord% Lib "User" (ByVal hwnd%, ByVal nIndex%)
Global Const GWW_HWNDPARENT = (-8)

Function isDev () As Integer
Dim ParentHwnd As Integer
Dim ParentClass As String
Dim iClassLen As Integer

ParentHwnd = GetWindowWord(form1.hWnd, GWW_HWNDPARENT)
ParentClass = String$(33, 32)
iClassLen = GetClassName(ParentHwnd, ParentClass, 32)
ParentClass = Left$(ParentClass, iClassLen)
If ParentClass = "ThunderMain" Then
isDev = True
Else
isDev = False
End If
End Function
```

```
OLEContainer1.CreateLinkToFile('C:\te
st.bmp',False);
```

OLE will then splutter and whir and the bitmap will be displayed. If the bitmap is later updated by another application, you can update it with:

```
OLEContainer1.UpdateObject;
```

In such a simple example, you could get similar results more efficiently using the LoadFromFile method of the picture in a standard image control, but the OLE approach has advantages. For instance, the OLE container will work with any OLE server on your system and supports things like in-place activation. You can save OLE objects to disk using TOLEContainer's SaveToStream method .

What about using Delphi as an OLE server, supplying information to display a graph or chart in a word processor document? Since this is not supported by the VCL, it is not easy to do in Delphi. The latest OWL or MFC-class libraries specifically support this OLE feature, so in this case C++ is a better option. What Delphi does support is OLE automation, so you could create an application that would supply the required data to any OLE automation client (see above; *Automating Delphi with OLE*).

If the client could also draw charts, as Microsoft Excel or Lotus WordPro can, then you could write code on the client side to extract the data and draw the graph. An insertable OLE object would be a better solution but an easy way to implement such a thing will have to wait for future versions of Delphi and Visual Basic. These will be able to create OLE controls; OLE automation objects with a visual interfa

PCW Contacts

Tim Anderson welcomes your Visual Programming comments and tips. He can be contacted at the usual *PCW* address, or at **freer@cix.compulink.co.uk** or **http://www.compulink.co.uk/~tim-anderson/**

Contemporary Software 07000 422224 (FarPoint Tab Pro 2.0; £99 plus VAT).
Visual Components 01892 834343 (Visual Developers Suite; £235 plus VAT).