A touch of **Class**

Visual Basic 4.0 gets inheritance, thanks to Class Assist. And where now for Delphi? Tim Anderson gets the lowdown from Borland's Zack Urlocker.

Sheridan's Class Assist is an ambitious, intriguing Visual Basic 4.0 add-on. It overcomes a key VB weakness — implementing inheritance, and supplies an easy way to tap the power of the Windows API, too. It requires Windows 95 or NT 3.51 to run, although code generated with Class Assist and 16-bit VB 4.0 runs under Windows 3.1.

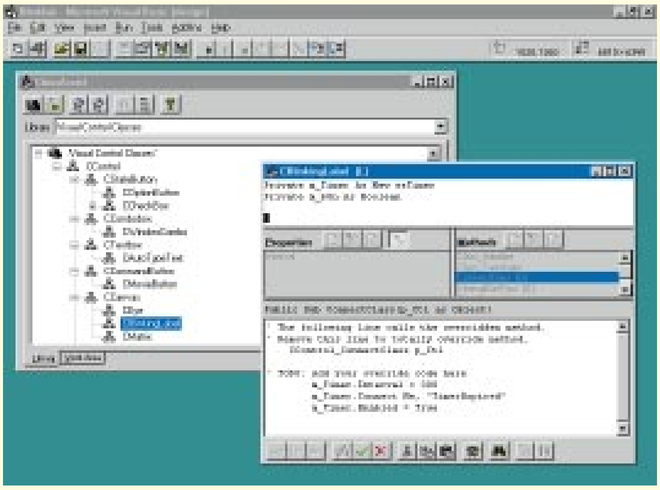
Fig 1

```
Public Sub Show()
MsgBox “MyApp “ & Chr$(169) & “ 1996 by Me”
End Sub
```

Fig 2

```
‘ The following line calls the overridden method.
‘ Remove this line to totally override method.
   CAboutBox_Show
‘ TODO: Add your override code here
Beep
```

The Class Assist IDE shows classes in a hierarchical list. Double-click a class to edit its properties and methods, or to create new ones



Class Assist installs itself as a VB add-in. There is an executable file which runs the Class Assist integrated development environment and which opens automatically with VB, plus several supporting OCX controls and DLLs.

The Class Assist IDE is a browser and editor for classes stored in Access .MDB library files. Source control is built in since classes are generally checked out to a work area for editing. A Team Development Manager controls user access privileges but there is no integration with SourceSafe.

Initial impressions are good. When you open or create a library, Class Assist shows classes in a hierarchical list.

To create a new base class, right-click on the top level and choose New class. To inherit from an existing class, right-click on the class name and this time the new class will be based on the existing one. In the ensuing dialogue, you can add new properties and methods, and override methods from the parent class as required. You can then move classes into the current VB project by drag-and-drop or from a pop-up menu option. You can also import existing class modules, in order to inherit from previous designs or to avoid writing code in Class Assist's primitive editor.

Under the bonnet, it is less appealing. The best way to see how Class Assist works is via a small example: imagine you have a CAboutBox class with just one method, Show. We are keeping things simple, so here is the code (*Fig 1*).

However, for some applications you require an about box that beeps when you close it. In ClassAssist you create a new class, CBeepAbout, derived from CAboutBox. Choose the option to override the Show method, and the code shown in *Fig 2* is displayed.

Inheriting from ancestors

In this case, the new method inherits the functionality of its ancestor, as well as adding its own code. To effect a total replacement, delete the call to CAboutBox_Show.

For this to work, the generated .CLS file which is pasted into your VB project has to include the code from any ancestor files. In this example, ClassAssist generated the following methods in the CBeepAbout class in addition to the standard Initialize and Terminate:

```
CAboutBox_Init()
CAboutBox_Term()
CAboutBox_Show()
CBeepAbout_Show()
Show()
```

The point to grasp is that when you call an ancestor method, you are not really calling the method in the ancestor class but a copy of that method which Class Assist has pasted into the derived class.

If Class Assist is used correctly, the two should be the same, but it is all rather clunky. In cases where an extensive class hierarchy is designed, with several generations of

Where next for Delphi?

Urlocker: "We typically win any technical evaluation."

We talked to Zack Urlocker, Borland's director of Delphi product management, about future plans. Will there be a version 2.0 of 16-bit Delphi? For Zack, it's an ironic question: Delphi 2.0 was originally meant to be a straight port to 32-bits, but according to Urlocker "It developed into something better. There's no decision yet about a revised 16-bit product."

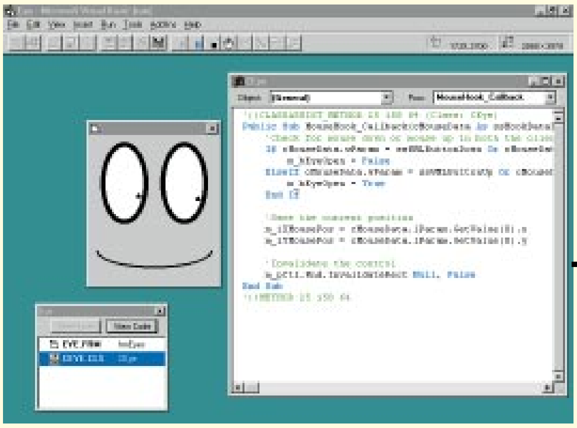
Asked about other platforms, like OS/2 or the Macintosh, Zack commented that "Windows has won on the client side. We might license the Visual Component Library (VCL) for another platform." Even so, Zack claims the compiler has the potential to target other platforms and could, for instance, compile for PowerPC, Alpha or MIPS.

Delphi 2.0 has a problem with OCX controls. "You can write OCXs in Delphi but it's a lot of work," Zack agreed. "By later this year we will make it easy."

Delphi 2.0 already contains components for creating OLE automation servers. What is envisaged includes an Expert for making OCXs and a means of converting a Delphi visual component into an OCX. But despite the prevalence of OCX as a Windows-universal component, it is not yet a firm standard. OLE lacks support for inheritance and has a long way to go in its evolution.

Will Delphi also support OpenDoc and CORBA-compliance? "We have cautious support for OpenDoc. But we've never had a customer ask for it," says Urlocker. Some developers dream of a Delphi C++ version; but there are no plans for such a thing. "C++ developers find the Object Pascal language is up to strength and we want it to be easy to learn. We couldn't achieve the same ease of use in C++."

Another problem is that the great complexity of C++ means more work for the compiler, so Delphi's near-instant compilation would be impossible. "But Java has a similar object model to Delphi. There's no multiple inheritance, or pointer problems. We've licensed Java and our goal is to encompass the same tools as in Delphi, but with the Java language," says Urlocker.



```
'{{CLASSASSIST METHOD IS 150 64 (Class: CEye)
Public Sub MouseHook_Callback(oMouseData As ooHookData
    'Check for mouse down or mouse up in both the client
    If oMouseData.wParam = wmLButtonDown Or oMouseDat
        m_bEyeOpen = False
    ElseIf oMouseData.wParam = wmLButtonUp Or oMouseD
        m_bEyeOpen = True
    End If

    'Save the current position
    m_lXMousePos = oMouseData.lParam.GetValue(0).x
    m_lYMousePos = oMouseData.lParam.GetValue(0).y

    'Invalidate the control
    m_octl.Wnd.InvalidateRect Null, False
End Sub
'}}METHOD IS 150 64
```

The eyes in this VB application follow the mouse pointer, even when it is over another application. It's done using WinAPI hook oblets, along with Class Assist and a couple of Visual Base controls

derived classes, there will be a huge proliferation of duplicated methods in your project. Even so, it delivers what it promises: inheritance for Visual Basic.

If that's what it takes to get inheritance, you might decide to do without it. But Class Assist has two other tricks up its sleeve.

One is Visual Base Controls, a set of basic OCX controls which can be linked to VB classes in order to modify their behaviour. When you drag a Visual Class onto a VB form, Class Assist places the control and also creates a new OLE server class module. For this reason you need the Professional, or Enterprise, versions of VB 4.0 to use visual classes.

At runtime, and before it is drawn on the form, the visual control calls methods in the associated OLE server. This means that VB code can modify the visual control at a lower level than is normally possible; for example, drawing a different style of button or listbox. You can also intercept and respond to any Windows message sent to that control. In effect, Visual Base Controls enable you to create your own custom controls with VB code.

Introducing oblets

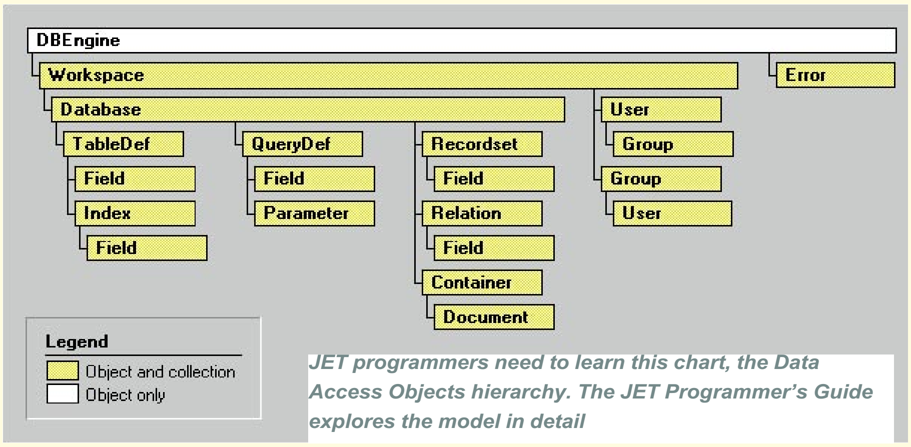
Visual Base Controls make extensive use of WinAPI oblets, the third key element in the Class Assist package.

Oblets are in-process OLE automation objects which encapsulate the Windows API. As an example, *Fig 3 (page 313)* shows how to set a form to be always on top, using oblets. Note that this is the

Books for Visual Programming

Not before time Microsoft has published the *Jet Database Programmer's Guide*, by Dan Haught and Jim Ferguson.

Eagle-eyed readers will spot some overlap with the Data Access Objects (DAO) SDK included with Visual Basic and Visual C++. A poster insert displays the DAO hierarchy in more detail than any of the online charts. There's a good chapter on security, and plenty of tips for optimising performance with both local and remote data access. Although spoilt by a weak chapter specific to C++, this title is recommended for any database developer using JET, the database engine behind Visual Basic and Access.



JET programmers need to learn this chart, the Data Access Objects hierarchy. The JET Programmer's Guide explores the model in detail

- Visual C++ developers will welcome *Inside Visual C++ 4.0*, a major update of Kruglinski's standard tutorial. It has grown by 300 pages, of which over 100 are devoted to OLE in all its incarnations: OLE automation, OLE structured storage, and the OLE Component Object Model.

Kruglinski does not aim to be a comprehensive reference but to give a clear introduction, with examples, using the Microsoft Foundation Classes throughout. In particular, this is an excellent guide to the document/view architecture espoused by MFC and the VC++ AppWizard.

- *Doing Objects in Visual Basic 4.0* will help VB developers make use of version 4.0's new OO features. There's a common-sense introduction to object-orientated programming and masses of advice on subjects like coding standards and user-interface design. One section describes how to fake inheritance in VB; another explains how to build a VB AppWizard add-in. Along the way, a relational contact management system is developed, using the techniques described and storing the data in a JET .MDB. There is useful technical material here but the book's strength is in the theoretical framework it provides, especially for those already familiar with VB 3.0.

Fig 3

```
Dim MyWnd As New ssWnd ‘ declare the oblet
MyWnd.Attach Form1.hWnd ‘ attach it to the form
MyWnd.SetWindowPos ssSWPHwndtopmost, 0, 0, 0, 0, ssSWPNomove Or ssSWPNosize ‘ call the SetWindowPos API function
MyWnd.Detach ‘ clean up
```

entire code. There is no need for declarations or constant definitions: all that is needed is a reference to the WinAPI oblets.

There are 24 oblets which encapsulate most of the Windows API, including areas traditionally difficult for VB, such as callbacks and hook procedures. Working with the API becomes a matter of interacting with the properties and methods of these oblets, using familiar dot notation. Another advantage is that oblets are non-visual, so you can use an ssTimer oblet in a procedure without needing a timer control on a form.

This is cool stuff, but there's a price to pay. Oblets are OLE servers, and instantiating an OLE object takes significant time. Once the object is created, performance is good, although slower than direct API calls. For example, we wrote a routine using ssDC, ssRect and ssWnd oblets, making repeated calls to FillRect. The inner loop took about 50ms for oblets, as opposed to 20ms for direct calls. At those speeds it may not matter. But the whole routine, including oblet creation, took 450ms with no equivalent overhead for the direct approach. That is 20 times slower. Careful application design is needed to avoid creating oblets, or any other OLE objects, at time-critical moments.

Class Assist is a superb extension to Visual Basic 4.0 and demonstrates what can be done with OLE. If you're happy with VB's performance and want more power, look no further. On the other hand, if you like the idea of inheritance, custom controls and an object-orientated approach to the Windows API, maybe Delphi is worth another look.

Secret Rich Text Format

M. Hodges writes: "*I write DOS programs which create text files intended for import into Word 6. I would find it a great advantage to incorporate RTF formatting control codes but Microsoft has not been helpful: its technical support people do not seem to have heard of RTF. You refer to a 'Developer Network CD'; I'm not familiar with this. Is there any other way I could get hold of RTF documentation?*"

The document is called "Rich Text Format specification" and is Microsoft Product Support Services Application Note 1/95 — GC0165. It should be obtainable from the Word product support team.

The Developer Network CD is a quarterly subscription service (call 0800 96 02 79 for details). It contains documentation for most Microsoft products along with bug lists, books and a host of further documentation and examples. It is excellent value for Windows developers, but is not much use for DOS work.

PCW Contacts

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

Class Assist £175 from Contemporary Software 01727 811999.

Books

All books available from **Computer Manuals** 0121 706 6000 (prices shown include VAT).

Jet Database Engine Programmer's Guide Book and CD £37.49; Microsoft Press.

Inside Visual C++ 4.0 Book and CD £41.99; Microsoft Press.

Doing Objects in Visual Basic 4.0, by Deborah Kurata. Book and disk £37.49; Ziff-Davis Press.