A new Deal

Tim Anderson checks out the new Formula One spreadsheet control in the latest upgrade to the Visual Developers Suite Deal, answers VB and Delphi queries and hides a blinking caret.

Visual Components has upgraded its Visual Developers Suite Deal, a collection of ActiveX controls for Visual Basic and other ActiveX clients. These are heavyweight components, each being almost an application itself. They are supplied as both 16-bit and 32-bit OCX controls. The runtime versions can be distributed royalty-free, making the Suite excellent value if you need this kind of functionality.

Cream of the crop is Formula One, now at version 4.0, a spreadsheet control which does a remarkable job of emulating Excel. It can read and write files in Excel format up to version 7.0, as in Office 95, but there are limitations: Formula One does not understand Excel charts or macros, for example. A large number of worksheet functions are supported, and the ability to move sheets to and from Excel is a valuable asset. Formula One has its own drawing tools and can link with First Impression, the charting control in the Suite Deal, to create charts. You can place buttons, checkboxes and drop-down listboxes on sheets.

New in version 4.0 is support for double-byte character sets, HTML export, and Uniform Data Transfer, an OLE standard which lets you drag and drop data between applications. Formula One is not a data-bound control, but it has built-in ODBC support so you can query an ODBC database and the results appear in a worksheet.

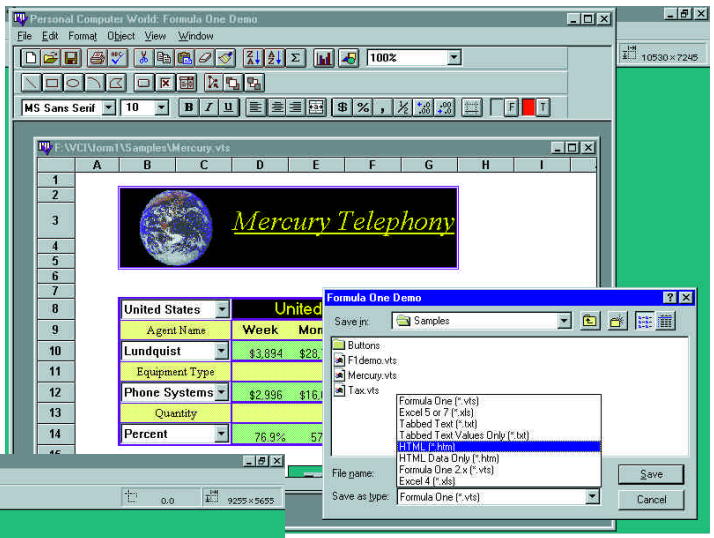
Version 4.0 includes several new ODBC functions. Another nice touch is the workbook designer, a fully-featured spreadsheet application which pops up on demand to enable you to create workbooks interactively.

Formula One is superb and its main competition is from Excel itself. Excel is a strong development tool and its worksheets can be embedded in other applications and controlled programmatically. Excel is the more powerful, but in comparison Formula One is small, nimble and royalty-free.

First Impression, the Suite's charting component, is updated to version 2.1. Not much has changed, mainly the support for double-byte character sets and Uniform Data Transfer.

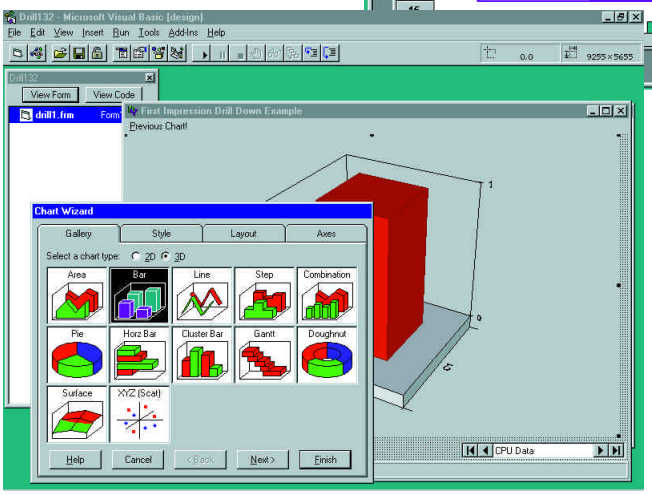
Visual Writer is a word processor control now at version 3.1. It's good, but not of the quality of Formula One. Under Windows 95 or NT, Visual Writer has to compete with the built-in Rich Text Control which does the same job of displaying formatted text with embedded bitmaps. Visual Writer does have some extra functions like fields, zooming, spell checking via the supplied Visual Speller control, print preview, and a ready-made button bar, status bar and ruler. It also has some quirks. Rich Text Format (RTF) is supported, but it prefers its own proprietary format. This is a disadvantage, especially since it will not accept .RTF as a valid format when bound to a document database. Also lacking is any kind of HTML support. For Windows 3.1 developers, though, Visual Writer or something like it is all but essential if you need to display formatted text. It's a shame the supplied 16-bit version is an

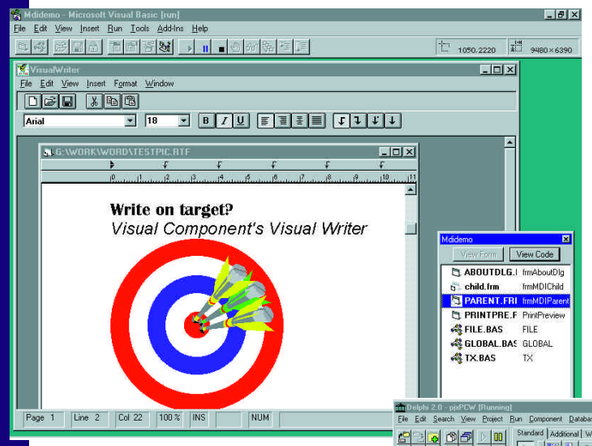
p316 ➢

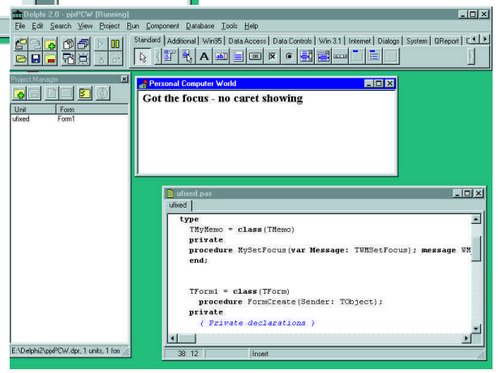


Right Now at version 4.0, Formula One will save in HTML format

Below First Impression's chart wizard offers a range of layouts and styles







Left Visual Writer is invaluable for 16-bit Windows but less useful in Windows 95

Below A solution to the flashing cursor problem. The TMemo object is subclassed, creating a component which is sufficiently general to be used many times over

OCX, as the VBX control type is more widely supported in Windows 3.1.

Carets and messages

The following question is asked by reader, Deborah Pate: "*How can I stop the cursor flashing in a TMemo component on a form with no other control that can accept the focus? Setting it to read-only does not help.*"

This is a fair question, although I am not sure why you would *not* want the cursor flashing in a memo control that has the focus. Anyway, this is the kind of thing that should send you scurrying to the Windows API. One thing you must realise is that what most people call the cursor, Windows calls the caret. There are eight functions specifically concerned with this little flashing creature. For example, you can control the blink rate with SetCaretBlinkTime. Hiding the caret is just a matter of calling the right function. That is:

```
HideCaret(Memo1.handle);
```

The remaining problem is where to call the function. The obvious place is in the OnShow event method for the form but it doesn't work. The memo component receives a SetFocus message after the form shows and that helpfully reinstates the caret.

The OnPaint event does the trick but this is not the best solution. In certain circumstances the memo control can receive a SetFocus message without the form's OnPaint event firing, and back comes the caret. If you call HideCaret in enough places you can probably make it watertight, but it's not elegant programming.

The best answer is to trap the SetFocus message itself. To do this you need to subclass the TMemo object, giving it a custom message handler. You can either do this entirely within your application, or create a new custom memo component and install it on the component palette. For a one-off, the first approach is fine, the only snag being that because you create the control at runtime, you cannot place it visually or use the Object Inspector. How it works is shown in Fig 1.

Of course, you will also want to set other properties and perhaps write event code for the memo control, all of which you can do in code. Sledgehammer to crack a nut? Maybe. But once you have learnt how to subclass Delphi controls, many other problems can be easily solved.

You can also build up a library of customised components which can be used many times over. For example, you could create a memo with a Boolean ShowCaret property that turns caret display on or off. In the long term, the productivity gains are enormous.

Input Pro

Once upon a time, it was Aware/VBX. FarPoint has renamed this set of data-entry controls to the more natural Input Pro. It is an unglamorous collection, but is also one of the most useful for anyone doing data entry forms in Visual Basic or other ActiveX clients. A VBX version is also supplied.

There is not much extra functionality in Input Pro, as opposed to Aware/VBX. The main difference is the move to ActiveX. There are eight controls including currency, date and time, masked edit, and a memo control which overcomes the normal 64Kb limit. All are data-aware. The main purpose of InputPro is for validating data entry (never an easy task): its controls greatly simplify matters. For example, the DateTime control rejects invalid dates and times, can limit their range, and can auto-complete partial entries.

Fig 1 Trap that SetFocus

1. In the type section of the form unit, declare the following object:

```
TMyMemo = class(TMemo)
  private
  procedure MySetFocus(var Message: TWMSetFocus); message WM_SETFOCUS;
  end;
```

2. In the public declarations for TForm1, include:

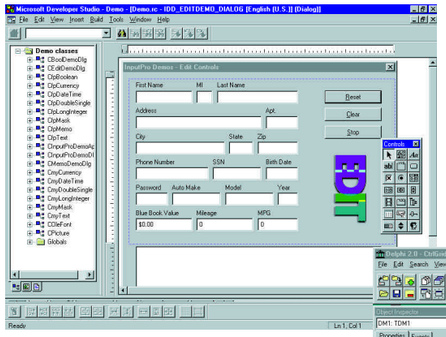
```
Memo1: TMyMemo;
```

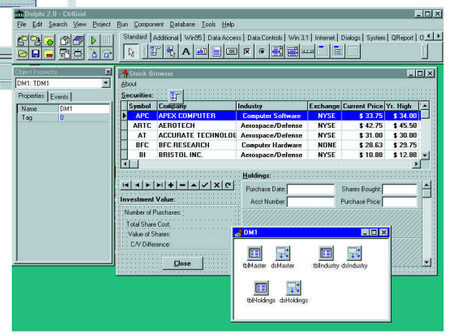
3. In the implementation section, include:

```
Procedure TMyMemo.MySetFocus(var Message: TWMSetFocus);
begin
inherited; {call the default handler for this message}
hidecaret(self.handle); {hide the caret}
end;
```

4. In the FormCreate method, include:

```
Memo1 := TMyMemo.create(self);
Memo1.Parent := self;
HideCaret(Memo1.handle);
```



Above FarPoint's Input Pro gives fine control over validation during data entry

Right Behind those drag-and-drop data controls lurks the Borland Database Engine, a substantial application in its own right

Rolling your own

Reader Richard Hustwayte writes:

"My project will require some databases to be made — nothing complex like client/server but simple, flat-file databases. I have looked at two versions of Delphi: the standard version (about £70) and the desktop version (about £230). The latter version is advertised as having the Borland Database Engine. What is this? And if I don't have it, am I unable to create database applications?"

All versions of Delphi come with the Borland Database Engine. This is a library of functions designed to simplify database work by acting as an intermediary between your application and the driver software which actually accesses the data. One benefit is that you can use data-aware components, so you can create simple database applications without writing any code. A number of database drivers are available for the BDE and it can also use ODBC drivers, the Windows standard for database access.

The BDE is good, but there is a cost involved. The BDE is a substantial piece of software and adds a considerable overhead to your application. Additionally, not all database drivers work well with it. In particular, it is not a good match for data in Microsoft Access format. For several reasons, then, you might not want to use it.

The most efficient alternative for a simple database is to roll your own using Delphi's file functions. If that does not appeal, there are database libraries such as Sequiter's CodePascal which provide a lightweight alternative. Then again, the BDE comes in the box and is fairly easy to use, so most Delphi developers do not look elsewhere.

Which Delphi book?

Darren Davies writes: *"I'm just about to buy Delphi Developer 2.0. I was wondering if you could recommend a book to learn how to program in this language? I've had quite a bit of experience with Pascal for DOS and object-oriented Pascal for Windows, but not much with visual programming."*

For someone with programming experience, a good choice is *Delphi 2.0 Unleashed* by Charles Calvert (SAMS). At 1,400 pages, it goes some way to compensating for Delphi's poor documentation.

A terminal problem

"I'm trying to get my new Terminal Program to automatically log in to a BBS. How do I get MSCOMM to wait for a login prompt before it enters the user details like Login name and Password?" asks Aaron Hodgson.

The MSCOMM custom control, which is similar in Visual Basic 3.0 and 4.0, offers two ways of intercepting data. The first technique involves a program loop which continually checks the receive buffer, a technique called "polling". Fig 2 shows what it looks like in pseudo-code. While it is a useful technique, it is difficult to write a well-structured application if it spends much time sitting in a loop like this. Another method is preferable, which is to use the OnComm event to respond to incoming data. This event fires whenever a communication error or event occurs. You can respond with a select case statement, like that shown in Fig 3 (page 319).

Your code should respond to all the

Fig 2 Pseudo-code for "polling"

```
Begin do loop
DoEvents or Sleep to allow windows to run other processes
Check InBufferCount property
If there is data, read input property and add to string buffer
Check buffer is not too full and correct if necessary
Check for time out, data complete, broken connection or other errors
End do loop
```

Java books

■ *Professional Java Fundamentals*
by Sly Cohen, Tom Mitchell and others

Most Java programmers are already skilled in another language: often C++. This book is aimed at that readership, providing a concise introduction to Java and focusing on its distinctive features. Beginning with a description of the Java language and object-oriented programming, it goes on to explain packages, threads and streams. Five chapters are devoted to the Abstract Window Toolkit, including a detailed explanation of various layout managers. The most advanced chapters cover networking, building libraries, implementing an application framework, and interfacing with C++.

There seem to be lots of poor Java titles around, and in contrast here is a knowledgeable and well-judged guide which complements rather than repeats what is easily obtainable online. Recommended.

■ *Using Java (Second Edition)*
by Joseph Weber and others

The flash on the cover states: "Covers new JDK 1.1 features" which is a bold claim since, at the time of writing, the JDK 1.1 was still in beta. You will find some useful material on JDBC database classes and a little on remote method invocation but, of course, much of JDK 1.1 is not actually included. What you get is over 1,000 pages which take you step-by-step through Java's tools, language, classes, applets and applications, graphics and layout, security and more. There is an emphasis on Sun's tools rather than third-party contributions, although the online version includes a chapter on different development environments.

Overall, *Using Java* is a thorough guide, although at times rather ponderous and unexciting. On the CD, you get online versions of four other titles covering JavaScript, Visual J++, CGI scripting and HTML, along with additional chapters and example Java applets. As a one-stop Java reference library, this book is hard to beat.

p319 ➢

Fig 4 WaitFor function

```
Function WaitFor(sWaitString As String, lTimeout As Long) As Integer

Dim lStartTime As Long
Dim sBuffer As String
Dim iOldThreshold as integer

lStartTime = Timer
iOldThreshold = Comm1.RThreshold
Comm1.RThreshold = 0
‘ prevents comEvReceive firing

Do
DoEvents ‘ or call Sleep API function
 If Comm1.InBufferCount > 0 Then
 sBuffer = sBuffer & Comm1.Input
 ‘ should check for buffer too large
 End If

 If InStr(sBuffer, sWaitString)  0 Then
 WaitFor = 0
 Exit Do
 End If

 If Timer >= (lStartTime + lTimeout) Then
 WaitFor = 1
 ‘ you can define constants and report errors
 ‘ using the return value
 Exit Do
 End If

Loop

Comm1.Rthreshold = iOldThreshold

End Function
```

Now you can write code like this:

```
If WaitFor(“login: “, 60) = 0 Then
‘ waits for up to 60 seconds
Comm1.Output = “qix” & Chr(13)
MsgBox “Successfully posted response”
Comm1.Rthreshhold = 1
‘ Enables comEvReceive event
Else
MsgBox “Login error”
Comm1.PortOpen = False ‘ Closes port
End If
```

possible events in order to trap communication errors. You also need to check that the string buffer is kept to a reasonable size. Using the CommEvent it is possible to write a reasonable communications program in Visual Basic, and there is an example called VbTerm that comes with Visual Basic.

Although the event-driven approach is better for most purposes, Aaron’s particular problem can easily be solved by polling. You can write a WaitFor function that doesn’t return until a particular piece of data has been sent, or until an error has occurred. An example of this is shown in Fig 4.

Note that if you have also written code to respond to the OnComm event, you need to ensure that events of the type comEvReceive do not fire when the WaitFor function is running. You can do this by setting the Rthreshold property to zero.

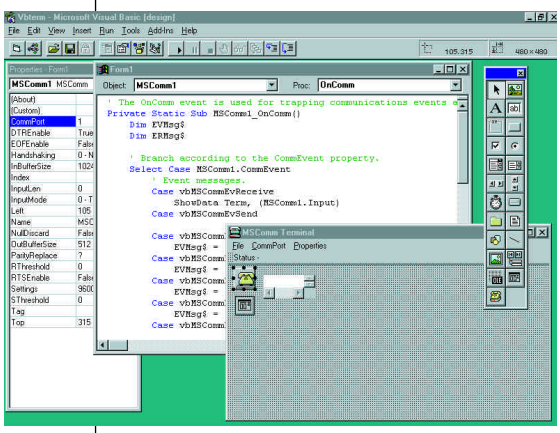
Finally, communications code is tricky, mainly because so many things can go wrong. At one extreme, poor lines and dropped connections cause difficulties, while the opposite problem is data coming in so fast that some part of your software cannot keep up. If this last problem occurs, Microsoft makes two recommendations. One is to extract data immediately the OnComm event fires, without bothering to check the type (see Fig 5). Also, the MSCOMM control may not always be satisfactory and as a last resort you can call the Windows API directly. This is well covered in the first edition of Daniel Appleman’s *Visual Basic Programmer’s Guide to the Windows API* but not in the second, 32-bit edition, although there is some material on the CD which accompanies the book.

Fig 3 A select case statement

```
Select Case Comm1.CommEvent
Case comEvReceive
  sBuffer = sBuffer & Comm1.Input
  ‘ or send to data processing function
Case comRxOver
  MsgBox “Error: receive buffer
 overflow”
Case comTxFull
  MsgBox “Error: transmit buffer full”
End Select
```

Fig 5 Extract data

```
Sub Comm1_OnComm ()
Static ReceiveBuffer As String
ReceiveBuffer = ReceiveBuffer &
Comm1.Input
Etc...
```



Above The VBTerm sample comes with Visual Basic and demonstrates the use of the MSCOMM control

PCW Contacts

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

Visual Developer’s Suite Deal is £235 (plus VAT) from Visual Components 01892 834343
Input Pro (FarPoint) is £105 (plus VAT) from Contemporary Software 01344 873434

Professional Java Fundamentals, by Sly Cohen, Tom Mitchell and others is £32.49; ISBN 1-861000-38-3, published by Wrox Press.
Using Java (Second Edition) by Joseph Weber and others costs £56.49 (incl. VAT); ISBN 0-7897-0936-8, published by Que.
Delphi 2 Unleashed by Charles Calvert costs £54.95 (incl. VAT); ISBN 0-672-30858-4.
These books are available from Computer Manuals 0121 706 6000