Lower your overheads

In search of some lean, mean data access, Tim Anderson bemoans the overhead of the Borland or Microsoft approach, and looks for something slimmer and faster.

The beauty of languages like Visual Basic or Delphi is that good-looking utilities can be knocked together quickly, without the need to understand the intricacies of the Windows API. In the real world, many such applications manipulate data: Visual Basic took off only when version 3.0 provided the built-in database functions via the JET engine (the same data access DLL which powers Microsoft Access). Borland's "VB Killer", Delphi, which borrows this same idea, uses the Borland Database Engine (BDE) to serve dBase, Paradox and Delphi itself.

JET and BDE are both core technologies for their respective vendors and they work well, as you would expect. Both are capable of handling several data formats, can plug into external database drivers and support a full range of network and client/server features. It sounds great, but how many of VB's 1.5 million users are developing client/server database applications? Often, the need is for easy access to local data within the context of an application that might be anything: comms package, video catalogue, game or multimedia presentation. Both JET and BDE represent overkill for this kind of general purpose use and while that may not matter, as long as there is plenty of spare horsepower in your PC, it can be significant, especially if you want to distribute your software as shrink-wrap or shareware.

Another factor is the volume of disk space occupied by the runtime files required to make it all work. Delphi is one of the worst offenders here, because alongside your application you need two install disks for the Borland Database Engine and even more if you need the ReportSmith reporting tool. There are two ways to avoid this overhead. The hardest, but most efficient, is to roll your own database code in native VB or in a DLL. Failing that, there are add-ons that aim to be leaner, meaner and faster than the standard items.

I've been looking at SuccessWare's Rocket, a set of DLLs and VBXs aimed primarily at VB and Visual C++, but also usable in Delphi. (Clipper developers may be familiar with the underlying technology, then called the SixBase driver.) The memory savings are real. I created a minimal JET database application, opening a single table in an MDB and displaying a couple of fields in bound text boxes. Using HeapWalker, Microsoft's utility supplied with Visual C++, gave the following results on a 486DX2/66:

Application	Heap memory used	Load time
VB/JET 2.5	1.29Mb	4.2 secs
VB/Rocket	652Kb	2.2 secs
Delphi/BDE	1.3Mb	3.4 secs
Delphi/Rocket	544Kb	1.6 secs

For anyone familiar with dBase, Clipper or FoxPro, programming with Rocket is straightforward since it is firmly in the xBase tradition. Data is stored in DBF files and you can choose between Clipper or FoxPro format using Rocket's Replaceable Database Engine. The main Rocket .DLL contains functions which match the core xBase set, for example sx_Skip, sx_GoTop, sx_Use. You can include a subset of xBase functions in index definitions. For example: UPPER("SURNAME") + LEFT(UPPER(FORENAME),3).

Including functions in index definitions is not my favoured practice (although in xBase, UPPER is hard to avoid). But the capability is vital for compatibility with

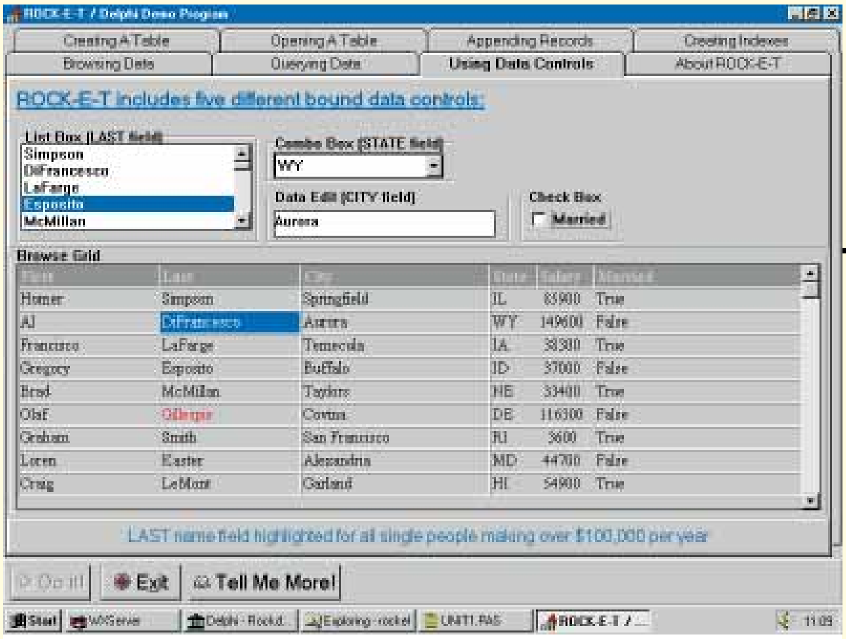
Using Rocket with Delphi

The shrink-wrap Rocket which I received from SuccessWare did not work with Delphi and errors occurred when I tried to install the VBX controls. Help was found on CompuServe, where SuccessWare have a section in the COMPB forum: there I found a file called ROCKET.PAS which installs a non-visual Rocket component into Delphi. This small component sets up all the types and functions used by Rocket; it also makes a call to Rocket's sx_setStringType function which tells Rocket to pass standard null terminated strings back to the calling application rather than VB-style strings.

What SuccessWare didn't mention was that ROCKET.PAS refers throughout to the multi-user version of Rocket and I had been sent the single-user version. A search-and-replace operation sorted that problem. The company did advise changing the names of the Rocket VBXs from 6Brow and 6Data to sxBrow and sxData respectively — apparently, Delphi is not happy with VBX components beginning with a number.

With these steps completed, Rocket and Delphi worked fine. But it's worth noting that using VBXs intended for Visual Basic in Delphi is not always smooth going. The situation is likely to improve as Delphi becomes better established. But better still, by the time you read this, SuccessWare expects to have a VCL component that will render the Rocket VBXs obsolete for Delphi programmers.

Left *This demo program, downloaded from CompuServe, shows that Delphi and Rocket can be a capable combination*
Below *Rocket brings Clipper-style programming to VB developers*



ROCK-E-T includes five different bound data controls:

List Box [LAST field]: Simpson, DiFrancesco, LaFarge, Esposito, McMillan

Combo Box [STATE field]: WY

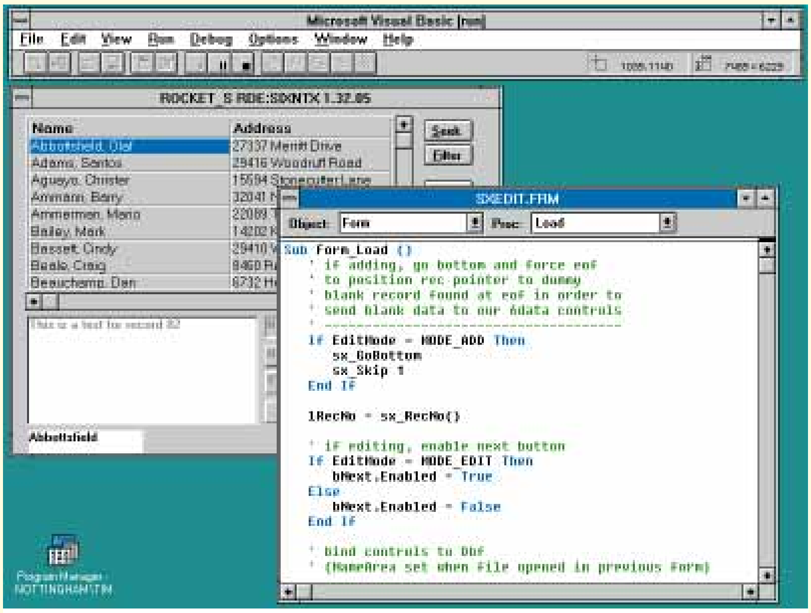
Data Edit [CITY field]: Aurora

Check Box: □ Married

Browse Grid

First	Last	City	State	Salary	Married
Homer	Simpson	Springfield	IL	83900	True
AJ	DiFrancesco	Aurora	WY	149600	False
Francisco	LaFarge	Temecula	IA	38300	True
Gregory	Esposito	Buffalo	ID	37000	False
Brad	McMillan	Taylors	NE	33400	True
Olaf	Gillespie	Covina	DE	116300	False
Graham	Smith	San Francisco	RI	3600	True
Loren	Easter	Alexandria	MD	44700	False
Craig	LeMont	Garland	HI	54900	True

LAST name field highlighted for all single people making over $100,000 per year



Name	Address
Abbottsfield, Olaf	27337 Merritt Drive
Adams, Santos	29416 Woodruff Road
Aguayo, Christer	15694 Stonecutter Lane
Ammann, Barry	32041 N
Ammerman, Mario	22089 T
Bailey, Mark	14202 K
Bassett, Cindy	29410 V
Beale, Craig	8460 Ri
Beauchamp, Dan	6732 H

This is a test for record 82

Abbottsfield

Object: Form   Proc: Load

```
Sub Form_Load ()
   ' if adding, go bottom and force eof
   ' to position rec pointer to dummy
   ' blank record found at eof in order to
   ' send blank data to our &data controls
   ' -----------------------------------------
   If EditMode = MODE_ADD Then
      sx_GoBottom
      sx_Skip 1
   End If

   lRecNo = sx_RecNo()

   ' if editing, enable next button
   If EditMode = MODE_EDIT Then
      bNext.Enabled = True
   Else
      bNext.Enabled = False
   End If

   ' bind controls to Dbf
   ' (NameArea set when File opened in previous Form)
```

existing indexes in old data files. Rocket has four sx_Eval functions which parse and evaluate xBase expressions, enabling them to be used by Visual Basic. It's messy, but a comfort for any xBase programmers struggling to come to terms with Windows.

A minimal Rocket application is simple: after adding ROCKET.BAS and a couple of VBXs to a project, a DBF file is opened with the sx_use function. Sx_use returns an integer representing a work area, identifying the table that has been opened. Each Rocket custom control has a Dbf property and by setting this to the value of the work area, the control is bound to the open table. A Fieldname property identifies the required field. Now the Rocket functions can be called to step through or search the data, with the displayed values being updated automatically. The bound controls supplied are a text box (which can also display pictures, stored by Rocket as compressed BMPs), check box, combo and list boxes, and a browse (grid) control. The browse control is free from the limitations of VB's grid, and claims to display up to one billion rows. Cells can be individually colour-coded and you can edit displayed data. Finally, two things that are difficult in native VB are easy with Rocket: data-aware controls have a Mask property which controls data formatting using standard xBase picture strings; and there's a set of encryption functions for automatic data encryption.

Rocket is rather good and fills a gap in both VB and Delphi. You can also use Rocket controls in Visual C++. It competes with Sequiter's CodeBase range, which includes CodeBasic for VB, CodePascal for Delphi or Pascal, and CodeBase for C/C++. For a VB programmer, the Rocket solution has additional features and is a little easier to program. Under C/C++, the offer of full source code makes CodeBase a superior choice: all the CodeBase products are supplied with the CodeReporter xBase reporting tool, whereas Rocket has no reporting features. Both products are xBase-orientated and, in a sense, outdated in their approach to data management, but they do come into their own for projects where the overhead of all that format-independent SQL handling is simply too high.

Acrobatics

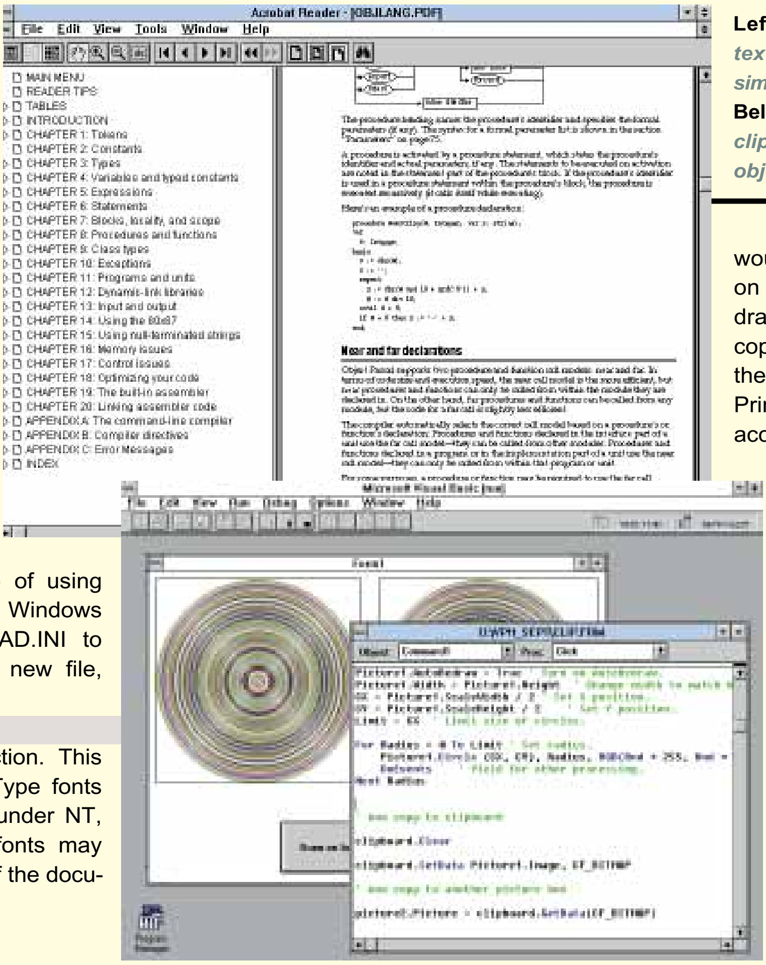
Adobe Systems has worked hard to establish its Acrobat reader and associated Portable Document Format (PDF). Readers of this column may well have come across this utility since Borland uses it extensively for Delphi and C++ online manuals, and the *Visual Basic Programmer's Journal* publishes all its back issues in PDF format. Acrobat gives a close on-screen portrayal of a printed page: if you print Borland's manuals from Acrobat, for instance, you get something very close to the book manuals that can be bought separately. This close match between the printed word and online form makes Acrobat documents easy and cheap for the publishers to create.

I mention this here in the hope that visual tools vendors will read it and take notice. If so, please do not use Acrobat for online documentation. Why? The reason is that good online documents are not at all similar to printed books. The priorities for programmers are quick access to relevant information, good cross-references via hotspots, clearly legible screen fonts and easy copying of example code. Microsoft seems to understand this better than anyone and items like the Office Developer's Kit, the Developer Network CD-ROM, and the Books Online supplied with Visual C++, are a joy to work with. Sections are short, the display is highly configurable; a collapsible outliner helps navigation, and you can carry out fast searches across a user-defined range of books. By contrast, Acrobat restricts searches to one book at a time and text is in book-width lines that will not wrap. To further aggravate matters, Adobe has not yet developed Acrobat for Windows NT although it does promise to do so. Although still a minority taste, NT is widely used by developers because of its great stability. I've nothing against Acrobat, but I don't believe that it is a suitable tool for online programming documentation.

In the meantime, there is a way (undocumented) of using Acrobat under NT. In the Windows directory, copy ACROREAD.INI to ACROEXCH.INI. Edit the new file, and add the line

```
ATMOption=1
```

to the [AdobeViewer] section. This makes Acrobat use TrueType fonts and thus allows it to run under NT, although the substituted fonts may mess up the appearance of the documents.



Acrobat Reader - [OBJLANG.PDF]

- MAIN MENU
- READER TIPS
- TABLES
- INTRODUCTION
- CHAPTER 1: Tokens
- CHAPTER 2: Constants
- CHAPTER 3: Types
- CHAPTER 4: Variables and typed constants
- CHAPTER 5: Expressions
- CHAPTER 6: Statements
- CHAPTER 7: Blocks, locality, and scope
- CHAPTER 8: Procedures and functions
- CHAPTER 9: Class types
- CHAPTER 10: Exceptions
- CHAPTER 11: Programs and units
- CHAPTER 12: Dynamic-link libraries
- CHAPTER 13: Input and output
- CHAPTER 14: Using the 80x87
- CHAPTER 15: Using null-terminated strings
- CHAPTER 16: Memory issues
- CHAPTER 17: Control issues
- CHAPTER 18: Optimizing your code
- CHAPTER 19: The built-in assembler
- CHAPTER 20: Linking assembler code
- APPENDIX A: The command-line compiler
- APPENDIX B: Compiler directives
- APPENDIX C: Error Messages
- INDEX

Near and far declarations

Microsoft Visual Basic [run]

```
Picture1.AutoRedraw = True ' Turn on AutoRedraw.
Picture1.Width = Picture1.Height ' Change width to match h
CX = Picture1.ScaleWidth / 2 ' Set X position.
CY = Picture1.ScaleHeight / 2 ' Set Y position.
Limit = CX ' Limit size of circles.

For Radius = 0 To Limit ' Set radius.
    Picture1.Circle (CX, CY), Radius, RGB(Rnd * 255, Rnd *
    DoEvents ' Yield for other processing.
Next Radius

' now copy to clipboard

clipboard.Clear

clipboard.SetData Picture1.Image, CF_BITMAP

' now copy to another picture box

picture2.Picture = clipboard.GetData(CF_BITMAP)
```

Left *When you re-size the text-viewing area in Acrobat, it simply makes the text smaller* **Below** *Copying a bitmap to the clipboard via VB's Clipboard object*

Clipping VB Pictures

Nigel James writes with the following question: "I enjoy reading your VB section in *PCW* — it has some useful info. I've been writing VB programs for about 12 months now, mostly for use at work (the Bodleian Library Map Section, Oxford). I would appreciate your advice on how, or whether, graphics drawn on a picture box can be copied to other applications via the ClipBoard. Presumably the PrintScreen key cannot be accessed in code, so how can I do it?"

VB has a Clipboard object which you can use for most clipboard operations. To place an item on the clipboard, first clear any existing data using Clipboard. Clear and then use the SetData method, specifying the format required. The example program in Fig 1, adapted from VB's documentation, shows how a graphic can be drawn on a picture box, copied to the clipboard and pasted into a second picture box. It requires a form with two picture boxes and a command button:

It's important to set the picture box's AutoRedraw property to True before attempting to retrieve the image otherwise you will not get the results you expect. The reason is that drawing to a picture box with AutoRedraw set to False does not update the persistent bitmap for the picture box. Another point is that you don't need all the code in Fig 1 to copy an image from one picture box to another — you can do that directly with:

```
picture2.Picture = picture1.Image
```

The clipboard is essential, though, for pasting images into other applications.

Fig 1

```
Sub Command1_Click()
Dim CX, CY, Limit, Msg, Radius  ' Declare variables.
Const CF_BITMAP = 2
Picture1.ScaleMode = 3   ' Set scale to pixels.
Picture1.AutoRedraw = True ' Turn on AutoRedraw.
Picture1.Width = Picture1.Height  ' Change width to match height.
CX = Picture1.ScaleWidth / 2 ' Set X position.
CY = Picture1.ScaleHeight / 2    ' Set Y position.
Limit = CX  ' Limit size of circles.

For Radius = 0 To Limit ' Set radius.
    Picture1.Circle (CX, CY), Radius, RGB(Rnd * 255, Rnd * 255, Rnd * 255)
    DoEvents    ' Yield for other processing.
Next Radius

' now copy to clipboard
clipboard.Clear
clipboard.SetData Picture1.Image, CF_BITMAP

' now copy to another picture box
picture2.Picture = clipboard.GetData(CF_BITMAP)

Picture1.AutoRedraw = False ' reset
End sub
```

PCW Contacts

Rocket is available from QBS on **0181 994 4842** and costs £115 for a single user or £195 for the multi-user version. ☎

CodeBase, CodeBasic and CodePascal is available from Sequiter on **0181 317 04321**. ☎

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