Error message

Software is getting more difficult to install and use, and Tim Anderson is not amused.

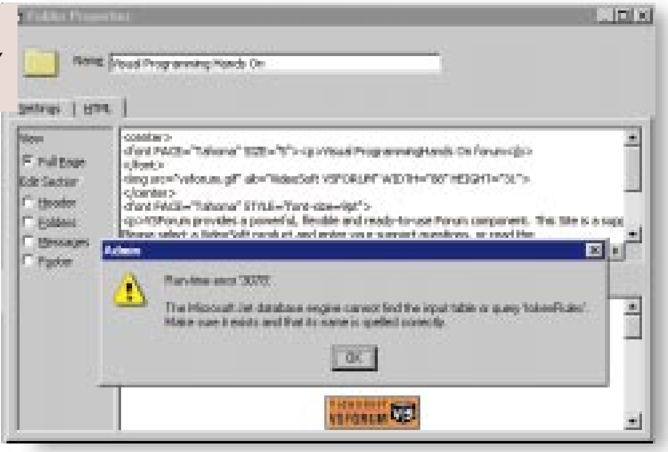
In 1993, Visual Basic 3.0 was as revolutionary for programmers as the graphical user interface was for users. Used sensibly, it was reliable, powerful and above all easy, especially compared to the intricacies of Windows development in C. Six years on, and with vast increases in both processing power and storage space, you'd think that programming would be even more accessible.

So what's gone wrong? Something has, because as a software reviewer I notice that software is increasingly difficult to install, once installed it's less reliable, and when an application is completed, deployment is more difficult than it was in 1993. I recently struggled to install a development suite with a four-figure price tag, documentation in a mish-mash of Windows help, HTML and a proprietary format, and with a setup routine that had no chance of completing without errors. It's strange, because in terms of features Visual Basic 6.0, for example, is vastly improved on its six-year-old counterpart. So is Delphi 4.0 when put next to Delphi 1.

The first problem seems to be that despite the best efforts of Microsoft and Inprise, the extra layers of complexity needed to provide those features are detrimental to performance and reliability. The second problem is that Windows itself has version control and registry difficulties that are not going away, although the next Windows NT might be improved. The third point is that the advent of the web and distributed computing has introduced a new set of development issues, without removing many of the old ones. Fourth, commercial pressures and the ease of web updates cause many vendors to release products that strictly should be beta or even alpha releases.

Is Java the answer? It could be, as the language itself is more productive than most alternatives. Java has its own problems though, and despite the hype Microsoft's enhancements aren't the main ones. Two bigger issues are performance and pace of change as Sun releases a constant flow of new APIs. Java development tools like VisualAge and JBuilder are even worse than Visual C++ when it comes to making a fast Pentium run like an old 386.

I've no desire to return to Visual Basic 3.0. Distributed, object-orientated, web-based applications are the future, and exciting possibilities abound. But things have to change before the world's software ceases to work, not because of the millenium bug, but in an avalanche of incomprehensible error messages.


Run-time error '3078':
The Microsoft Jet database engine cannot find the input table or query '[illegible]'. Make sure it exists and that its name is spelled correctly.

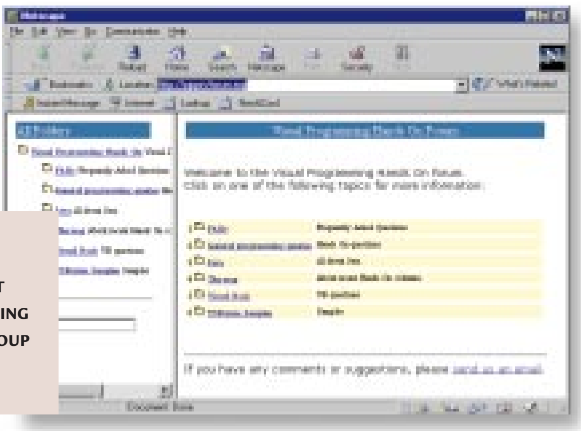
▶Oops! VSForum falling at an early hurdle

Forum for discussion

VideoSoft has an excellent set of products; however, VSForum is disappointing. Described as a server component, it's a new angle on the add-in scene, being a server-side component for ASP (Active Server Pages) rather than the usual ActiveX control. Active Server Pages are an extension to Microsoft's Internet Information Server for creating dynamic web sites. VSForum provides an instant discussion forum, using an ActiveX DLL to store messages in an Access database, serving up the data as HTML to create hyperlinked discussion pages. Sadly, VideoSoft has abandoned its printed manual, but it installed easily onto an NT server already loaded with IIS and all the ASP extensions. That is, until I made some edits to the demonstration discussion page — the admin utility presented an unhelpful message and exited without saving any changes. It's a known bug, fixed in the later release on VideoSoft's web site.

Perhaps more serious than the error itself, though, is the admin utility, which appears to be written in VB and lacks the error handling required to catch the error and exit gracefully.

It's as well I downloaded the latest version, as certain features like message threading, probably considered essential in discussion software, weren't implemented in the first shrinkwrap release. The patch was 5Mb, most of which was duplicate Microsoft runtime library files. It does work though, and I


Welcome to the Visual Programming Hands On Forum.
Click on one of the following topics for more information.

If you have any comments or suggestions, please send us an email.

▶The fixed VSForum: not bad, but nothing like a newsgroup

CROSSING THE PROCESS BOUNDARY

Ian Gregory has written in with a problem: "Using Delphi 4 I've been writing an editor, and I've set up the project source so that only one instance can be run at a time, using some code from Inprise's Delphi developer support site. This now means that if the user tries to run a second instance, that instance is terminated and the first brought to the front. If the second is invoked via a file association, the first instance is brought to the top but has no way of knowing that it must load a file. I've tried using SendMessage to send the command line of the second instance to the first, via a PChar typecast as an integer. Although the first instance receives the message, it cannot be used to open a new editor window. Is there a way around this?"

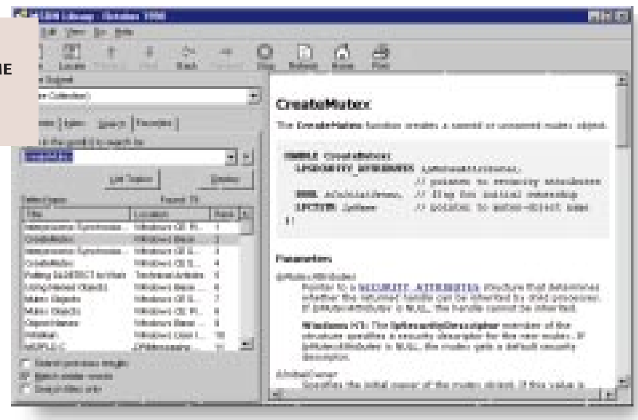
Only one instance of Ian's application, which is probably MDI (Multiple Document Interface), should ever be needed. As Ian states, attempting to start another instance brings the first instance to the front. This works fine using a mutex (*see main text, below*). But what if the second instance is started with a command-line parameter, such as when the user double-clicks the associated document type in Explorer? Now the second instance has to pass a string parameter to the first instance. The problem is, each instance has its own process and its own address space. Strings are pointer types, but pointers are only valid within a process. If you follow Ian's example, the second instance will receive an address which it cannot easily access, or most likely attempt to use random data in its own address space.

One thing is puzzling. Why is it that you can successfully pass a PChar typecast as an integer with a message like WM_SETTEXT? You can use FindWindow to obtain the handle of another instance of your application, and use SendMessage to set the caption of its main window. If this works (which it does), why can't you do the same with your own custom message? Because messages like WM_SETTEXT are a special case. For compatibility with 16-bit Windows, where applications all share one address space, the API copies the data across the process boundary. If you trap the message when it arrives in another process, you'll find the integer value of the pointer has changed; the pointer that arrives isn't the pointer you sent. There are no such special favours for your own custom message. If you use this technique, be sure to use SendMessage, not PostMessage. PostMessage will free the string before the message gets processed.

The solution for Ian is to find a way of passing data between processes. There are a bewildering variety of ways to do this, including COM automation. One easy way is to use WM_COPYDATA. This message exists precisely for the purpose of copying data across process boundaries. It's particularly useful because it works between 16- and 32-bit applications. Fig 1 (*p270*) is an example. For this to work, place a call to UseWMCopyData in the initialisation part of an application's main form unit. Build the application and run two instances. The second instance will update the form caption of the first, using WM_COPYDATA.

I've used FindWindow for the sake of brevity — it's not the best approach. It may not catch two instances started near-simultaneously, as the window may not have been created. Best to use a mutex to discover if another instance is running, and if it is, exchange custom messages to find its handle. You can use the handle of the main form, as here, or use the application handle, which is the handle of the hidden window used by every VCL-based Delphi application.

successfully created a discussion which worked in both Netscape and Internet Explorer. A good point is, it was easy to search messages and display results. One bad point is that the administration is not web-based, so you have to do it at the server where VSForum is installed. Another is that the Forum is fairly slow retrieving and displaying messages, probably because using Access via ODBC is sub-optimal. Furthermore, there's a better way to do discussion groups and that's with a private news server, although it's more complex to set up. Finally, VideoSoft wants a licence for every forum you set up, whereas FrontPage comes with an unlimited discussion web wizard that performs better and costs less. VSForum is a little more flexible than the FrontPage discussion wizard, thanks to a set of tokens which let you edit the discussion pages' content, but most will find FrontPage or other options better value. I'd feel differently if VideoSoft supplied the full source to the ActiveX DLL so users could customise VSForum, but it's not there. ActiveX components are a great idea, but this one's not VideoSoft's finest hour.



▸**MSDN — DON'T LEAVE HOME WITHOUT IT**

Mutex mysteries revealed
To discover if an instance of your application is already running, the correct 32-bit way is to create a mutex — short for 'mutual exclusion'.

[FIG 1] **Using WM_COPYDATA**

```
{start of unit omitted}
type
TForm1 = class(TForm)
protected
procedure WMCopyData(var Message: TMessage); message →
WM_COPYDATA;
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMCopyData(var Message: TMessage);
var
filename: string;
cds: CopyDataStruct;

begin

cds := CopyDataStruct(pointer(Message.lParam)^);

filename := strpas(pchar(cds.lpData));
form1.caption := filename;

Message.Result := 1;
{don't call inherited message handler}

end;

procedure UseWMCopyData;
var
hTarget: hwnd;
lpzFilename: pchar;
cds: copydatastruct;
lResult: integer;

begin
hTarget := FindWindow(nil,'My Application');

if hTarget > 0 then begin

lpzFileName := stralloc(256);
 try
 strcopy(lpzFileName,'This is a new file name');
 cds.dwData := 0;
 cds.cbData := strlen(lpzFileName)+1;
 {length of string including null terminator}
 cds.lpData := lpzFileName;
 lResult := sendmessage(hTarget,WM_COPYDATA,application. →
handle,integer(@cds));
{must use sendmessage}
 finally
 strdispose(lpzFileName);
 end;

if lResult <> 1 then
 showmessage('Operation ^
failed');
 end;

end;
initialization
useWMCopyData;
end.
```

[FIG 2] **Preventing a multiple instance**

Place this code at the end of the main form unit:

```
initialization
hMutex := openmutex →
(MUTEX_ALL_ACCESS,False, →
'Visual Programming →
Hands On');
 if hMutex <> 0 then begin
 showmessage('Already →
running - bye!');
 application.terminate;
 end
 else
 begin
 hMutex := createmutex →
(nil,False,'Visual →
Programming Hands On');
 if hMutex = 0 then →
showmessage('Error creating →
mutex');
 end;
```

Mutex objects are like traffic lights. Their only powers are first, to be seen, and second, to change state. Their visibility is high because they're global to the system.

A mutex is identified by a unique string. Having decided on a string, you call CreateMutex for a handle to a new or existing mutex object, or OpenMutex for an existing one. The mutex object is destroyed when the last handle to it is closed. Handles can be specifically closed with CloseHandle, or the system will destroy them when the process that created them terminates.

Mutexes are primarily used when synchronising threads. A mutex can be either signalled or nonsignalled. If it's signalled, a thread can call a wait function such as WaitForSingleObject, get ownership of the mutex, and set its state to nonsignalled. Other threads calling the same function will have to wait until the first thread has released the mutex before they can get ownership.

If you want to prevent mutiple instances of an application, you don't need wait functions. Check for the existence of a mutex identified by your unique string. If it exists, then only another instance can have created it. If it doesn't exist, create it and run. Fig 2 shows Delphi code.

PCW CONTACTS

VsForum costs £411.25 (£350 inc VAT) from Contemporary Software on 01344 873434 www.videosoft.com

Tim Anderson can be contacted via the PCW editorial office (address, p10) or email visual@pcw.co.uk