February, 1999

Editor's View

Lose Your Assumptions

The only way to really learn Visual FoxPro is to stop trying to make it into whatever language you used to use.

By Tamar E. Granor, Editor

"It's tempting to try to make the tool work the way one is used to working, but probably more productive to learn how to use the tool as it is. "

The quote comes from a relative newcomer to Visual FoxPro. He was trying to understand how a particular part of the product works and his basis for comparison was his last language (in this case, C++). He was struggling with the process of losing his assumptions in order to truly learn VFP.

His problem seems pretty common these days. I've seen a tremendous increase lately in the number of people learning VFP for the first time. Many of these new VFP users come from FoxPro 2.x or Clipper backgrounds, so along with making the move from a procedural language to an object-oriented one, many are also coping with the jump from DOS to Windows applications. A lot of them are expressing frustration that their tried-and-true techniques don't work or don't work well in VFP. (As the case above shows, however, not everyone learning VFP is from the Xbase camp. The inclusion of the product in Visual Studio has also brought a group of developers more familiar with other Microsoft programming environments. They bring a whole different set of experiences to their learning.)

Those of us who went from FoxPro/DOS to FoxPro/Windows remember the difficulties of working in a strange new world where you had to relinquish some of your control to the operating system and where lining things up on a form or report changed from a no-brainer to a complex task. At times, it seemed that we'd never be as productive in the Windows environment as we were in DOS.

The step from FoxPro/Windows to Visual FoxPro was even larger. Much of what we knew, especially about designing input forms, had to be thrown away. In other areas (like data handling/conflict resolution), we didn't *have* to throw out what we already knew, but all the experts told us to and using the old way was a lot harder in VFP. As with the DOS to Windows jump, many of us despaired of ever understanding VFP the way we had FoxPro 2.x.

Somehow, those of us who made the transition several years ago survived and find ourselves more productive in the VFP environment than we were in FoxPro 2.x. We've learned that relinquishing control of a feature (say, colors or printing) to the operating system just means one less thing we have to code into our applications. We've found that OOP means more work up front, but a bigger payoff later when our carefully crafted objects can be dropped into applications with no more work than setting a few properties.

Now, a whole new wave of developers is struggling with the same issues. What can we tell you to help you up the steepest parts of the learning curve?

First, of course, keep at it. For some reason, OOP is one of the "a-ha" subjects. That is, you work at it and work at it and work at it and it never seems to make sense and then, one day, it just falls into place. I've heard that from enough people to believe that it's a pretty common experience.

Second, get rid of your assumptions. Plenty of things in VFP do work as they did in FoxPro 2.x (especially on an individual level – functions like LEFT() and MOD() haven't changed). However, the way you build applications and think about them is different. Just because you always began your forms with SCATTER MEMVAR and then issued GETs against those variables in 2.x doesn't mean you should do so in VFP. (In fact, you shouldn't. Use buffering.)

Third, despite the first two points, in many ways, object-oriented development isn't a lot different than procedural.

In procedural languages, you write code, and when you write something for the second or third or fourth time, you realize that you're going to need it a lot. So you pull it out, give it some parameters to make it abstract, and make a function out of it. Once you get the function working, you use it wherever you need it and never think about what's inside again.

The process in object-oriented languages is more or less the same. When you find yourself doing the same things again and again, you pull it out and create a class from them. The difference is that you can abstract not just one process at a time, but a collection of related processes. Most important, you have a way to store the data with the functions (methods) that operate on it, so the calling routines don't have to know anything about the way the data is stored. Once you've built and tested your class, you never have to think about what's going on inside again. You just create objects and call their methods as needed.

Finally, perhaps most importantly, keep in mind that those of us who sound like we know what we're doing have been here for a while. We didn't learn it overnight; we just started before you. You *will* get it, but it's not instantaneous. Just let yourself do what the quote at the top says-learn to use the tool as it is-and after a while, when you have to go back to FoxPro 2.x, you'll wish for all the things you've learned in VFP.

Updates! Already?

Although many of you are probably just breaking the shrink-wrap on Visual FoxPro 6 or Visual Studio 6, there are already updates available. A service pack for all of Visual Studio fixes some bugs with runtime distribution. Although these fixes don't directly affect Visual FoxPro, you may want them anyway for the updated .DLLs. More important to VFP developers are updated versions of the Setup Wizard, the Component Gallery and several other tools that come with the product. To download these pieces, check out http://msdn.microsoft.com/vfoxpro/downloads/updates.asp. For the Visual Studio Service Pack, go to http://msdn.microsoft.com/vstudio/sp/default.asp.