[Start preparing for FoxPro 3.0 with the PTX Editor and Visual Properties]

“What can I do to prepare for FoxPro 3.0?” is a question I’m hearing more and more. After Dr. Fulton’s sneak preview of features at last fall’s DevCon, and more recently, a taste of object-oriented screens at the spring DevCast, our appetites have been well-whetted. But it’s still going to be many months before we see the next version of FoxPro for real.
One of the answers to this question is “Pick up Visual Basic and play with it. You don’t have to build applications, but just get used to the look and feel - how the interface is built, what types of tools you have available, and so on.” But suppose this solution, for whatever reason, doesn’t work.
This month’s Cool Tool is a pair of related utilities that will start to give you a feel for what we might see and use in FoxPro 3.0.
Many people have found working with the Screen Builder awkward. With it’s multitudinous dialogs, objects, prompts and so on, it’s confusing to learn and seems to get in the way more than it ought to. Furthermore, there are some features that are difficult or impossible to use. For example, you are limited to the tiny edit region in the Comment snippet - it can’t be expanded into an edit window like the other snippets. This makes typing or modifying comments longer than about 40 characters a real nuisance.
In the meantime, one of Visual Basic’s features is a properties editor that enables you to view and modify the properties of a particular object using a single browse window-like control. The left side of the window lists all the properties of an object - such as the code attached to particular events (in FoxPro, we know these as snippets like When and Valid) and physical attributes (like fonts and colors). The right side lists the current value - a function call or code snippet, the name of the font, the width of the pen line, and so on.

PTX Editor

Our first Cool Tool is the PTX Editor by Ken Levy (yes, he’s at it again.) PTX is an acronym for “Power Tools Extension.” The Power Tools Extension Editor provides a cleaner mechanism for using the Power Tools and modifying the surfaces used by the Power Tools. Furthermore, it provides the ability to run additional programs that extend the capabilities of the Power Tools.
For example, to take the example mentioned above, with the PTX Editor, you’ll be able to select a Comment snippet and call it up into a full Edit window. Furthermore, in order to move from the Valid snippet of one object to the When snippet of a second object, you’ll just select a different item in a list box and have the snippet available in the editing window, instead of having to go through multiple dialogs for each object.

Installing and running the PTX Editor

The PTX Editor consists of a single .APP file, from which a few other files are created when necessary. To install, place PTXEDT.APP in FoxPro’s path and issue the command

```
on key label rightmouse do PTXEDT
```

(You can use a different hot key, but since we’re going to mousing around, RIGHTMOUSE is going to prove handy.)

Then, open a Power Tool surface, such as a screen, menu, or report, position the mouse over the surface somewhere, and click the right mouse button. The examples that follow will use screens because they’re more visual, but it’s important to remember that these tools work for ALL the Power Tools.

[Screen shot: ROCK1: A sample screen in the Screen Builder]

[Screen shot: ROCK2: Clicking the rightmouse button while the mouse is over the screen brings up the PTX Editor.]

The PTX Editor dialog displays several popups and an edit region. The Object popup contains all of the objects in the surface - SAYs, GETs, boxes, lines, and so on in screens, fields, text objects and box/line objects in reports, and so on. You can move from one object to another simply by selecting the appropriate object in the popup.

[Screen shot: ROCK03: The Object popup displays all of the objects for the current Power Tool surface.]

The Snippet popup contains all the snippets for that particular object. If the object is the underlying surface - such as the screen itself - the choices that display in the popup are the screen level snippets - Setup, Cleanup, Activate, Deactivate, Show, etc. On the other hand, if the object chosen from the Object popup is an object like a pushbutton, the Snippet popup will contain choices for the When, Valid, and Message snippets, to name a few.

[Screen shot ROCK04: The Snippet popup displays all of the snippets for the currently selected surface or object.]

Some of the notation used in these two popups is a bit confusing. For example, the "General" choice in the Snippet popup refers to the Comment snippet. Furthermore, the "FocusMessage" refers to the "Message" snippet.

PTX Features

PTX doesn't only provide an easy way to navigate between surface and object snippets. As mentioned. Selecting the Comment snippet (or any other snippet, for that matter) for a particular object and then selecting the Edit pushbutton on the right side of the PTX dialog will bring forward a full screen Edit window.

[Screen Shot ROCK05: The Edit pushbutton in the PTX dialog displays a full screen Edit window.]

You don't have to use the Object popup to select an object. Suppose you can't remember the name of an object, but you know what the object looks like (ever had one of those screens?) Although the PTX Editor dialog is modal, you can "push it aside" and then right-click on the desired object. A message will display - "Searching for object" and when the object is found by the PTX Editor, that information will display in the PTX Editor dialog.

Another feature is the ability to verify the contents of a snippet. A lot of developers are pulling code out of snippets and placing it into functions that are called as expressions from the snippet. One major reason for doing so is that a simple typo in a snippet will not be caught during screen generation and thus forces the developer to regenerate the screen simply to correct the spelling. The Verify pushbutton displays a yellow light when the snippet has been changed. After the snippet has been verified, the stop light will show green if the snippet is valid or red if the snippet isn't valid.

[Screen shot ROCK06: If a snippet is Verified and does not pass, the snippet will display in a full screen Edit window and the appropriate error message will be displayed.]

A third feature is the Beautify pushbutton. It simply calls BEAUTIFY.APP, but it's handy to have it here where we're doing all our snippet editing. The Browse pushbutton provides direct access to the underlying table (named PTXDATA) that contains PTX-style records for each object in the surface. When you're done using the PTX Editor, Save your changes (if desired), and then Close. You'll be returned to the surface you had initially brought up.

This tool is all well and good - it provides a much cleaner interface and additional functionality to our design surfaces, but we're not done yet. The unnamed popup in the upper right corner next to the "PTX" pushbutton contains the names of PTXs. The last feature we're going to talk about are these "PTXs."

Remember, this tool is called the "PTX Editor" and, strictly speaking, we should be able to edit a PTX, whatever that is...

A PTX is a program (a PRG, an SPR, or an APP) that can be run while you are in the PTX Editor dialog. These programs EXTEND the Power Tool's capabilities. In other words, a PTX - a program - extends the way you work with a particular Power Tool. For example, some clever soul might figure out a way to provide cut and paste capability within the Menu Builder, and write a program that automates it. This program would be a PTX, and we could find a prompt like "MB Cut&Paste" in the popup. (This is an example - not an existing tool - so don't call me wondering where to find it...)

[Screen shot: ROCK07: The PTX popup displays the names of PTXs (programs) that can be run while in the PTX Editor.]

The PTX Editor comes with a couple of simple PTXs automatically installed. The About PTX just runs an "About..." screen. The Configuration PTX is even easier - it's an APP that consists of the line

```
MODIFY FILE SYS(2019)
```

so that you can change the current CONFIG.FP file.

[Screen shot: ROCK08: Running the Configuration PTX displays the CONFIG.FPW file in an Edit window.]

The third PTX included with the PTX Editor is the 3D PTX. Selecting it issues a dialog that allows you to select a 3D GENSCRNX directive that will automatically be placed in the object instead have having to type that code in the Comment snippet yourself.

[Screen shot: ROCK09: Running the 3D PTX when a box is the current object.]

[Screen shot: ROCK10: The 3D PTX automatically inserts the appropriate 3D GENSCRNX directive in the Comment snippet.]

Properties PTX

A more complex example of a PTX is the Visual Properties PTX, which is our second Cool Tool of the Month. Remember that PTXs are simply programs. The PTX Editor comes with a few examples, but the idea behind the tool is that anyone can write a PTX. In fact, the PTX Editor is a public domain utility, like the GENX tools, in order to encourage others to write their own PTXs and contribute them to the community.

Andrew Ross MacNeill has been working in concert with Ken and Steven Black to create a PTX that displays all the attributes of an object in a control much like Visual Basic or Access. That’s right, there’s no more waiting for FoxPro 3.0 to get used to the idea of being able to view and modify the properties of an object.

Installing the Properties PTX

The Properties PTX (“Props”) extends the Power Tool by providing a “properties sheet” for an object, like Visual Basic. Props is based on a properties table - a dictionary, if you will - that determines which properties are appropriate for a given object. For example, the “Valid Snippet” property will display for a GET, but not for a box. In other words, properties are object-specific.

[Screen shot: ROCK11: Running the Properties PTX displays a list of properties for the currently selected object.]

In order to use Props, you must install it in the PTXEDT database. Remember, the PTX Editor only comes with three PTXs by default. If you want to add more, you need to add them yourself. They don’t just magically appear.

All you need to do is add a new record to PTXEDT.DBF. (This DBF is automatically created the first time you run the PTX Editor.) The values for the fields are: Code = 1, Name = xxx, where xxx is the English name you want to display in the PTX popup in the PTX Editor dialog, and Progname = the fully qualified path for the PTX if it’s not in the FoxPro path.

Hint: PTXs are ordered in the PTX popup alphabetically. If you have a PTX that you want to use a lot, use a name close to the front of the alphabet, or, even better, being the name with a space.

Using the Properties PTX

Once the Properties Sheet is displayed, you can perform several functions. First, you can move from one object to another by selecting clicking (left mouse button, that is) in the “X” column in the first row. Doing so will display a popup of available objects in the surface (remember, Props is not just for screens - but for all Power Tools.)

[Screen shot: ROCK12: Clicking on the “X” column in the first row of the Properties Sheet displays a popup of all available objects in the surface. In this example, the GET object m.cNaThird has been selected.]

The value for each property can be changed. The tool used to do the changing depends on the type of object. If you select a property that has an expression, you’ll call up the FoxPro Expression Builder, and if you select a procedure, you’ll call up a full screen Edit window. You can also edit some properties right in the Properties Sheet - for example, font size, pen width, fill colors, and so on.

We can create some interesting pyrotechnics with this capability. For instance, suppose we wanted to change the “Is Dead” checkbox to a radio button so that we can have three choices: Alive, Dead, and Not Quite Sure (for Elvis, JFK and Jimmy Hoffa.) We’d click in the Object Type row and select the desired object type from the resulting popup.

[Screen shot: ROCK13: Clicking on the “X” column in the Object Type row displays a list of available objects to choose from.]

After changing the checkbox to a radio button, we’ll have to save our changes in the PTX Editor. Realize that making the change in the Properties Sheet actually went into the SCX and changed Object Type from 14 to 13. We’ll have to use the Screen Builder to add our additional Radio Buttons.

[Screen shot: ROCK14: Saving the changes in the PTX actually modifies the SCX. Here, the checkbox has been changed to a radio button (with only one choice, so far.)]

Another example would be to turn a check box into a pushbutton.

Well, that’s plenty for this Cool Tool Double Feature. We’ll explore the PTX Editor in more detail, and talk about creating PTXs in future issues of FoxTalk.

Where to find it

PTXEDT.APP and PROPSPTX.APP have been placed in the public domain. They can be found on this month's Companion Disk or downloaded from CompuServe's FoxForum.