What's up with theTime?

As a Visual Basic developer I wondered how to achieve some of the simple effects of one the basic standard Windows apps - the clock program. For instance, you can double click on the face of the clock to eliminate it's title bar. And without the title bar you can still click and drag the clock around your screen.

How do you do that in VB? You can't change a form's border property at run time and you can't move a form that has no title bar. And what about adding menu items to the system menu? And popping up a menu on a right mouse click without having any menu bar? These are some of the questions I had when I started developing TheTime. My goal was to make a more interesting clock program to replace the standard one and learn a few tricks about VB along the way.

Need... More... Information...

One of the first things I did when learning VB was to visit the local bookstore and look for anything that went beyond the standard VB intro subjects. After slogging through a pile of VB books I ended up plucking down my hard earned plastic for two books - "The Visual Basic Programmer's Guide to the Windows API" published be Ziff Davis Press and "The Visual Basic How-To (second edition)" published be the Waite Group Press. Both are excellent books that complement each other well and tell you pretty much everything you need to know beyond the basics (ahem) of VB. My setup program is derived from the setup program that comes with the "VB How-To" book while the "VB Programmer's Guide to the Win API" helped considerably in understanding bitmaps, palettes, drag 'n drop, and much more.

I've also been a subscriber to the Microsoft Developer Network CD and wouldn't leave home without it. Browsing through this CD taught me more about palette management then any sane person would ever want to know plus I discovered the MsgBlaster custom control which makes it possible to subclass any VB control. The source code to the MsgBlaster control is provided on the MSDN CD. I eventually modified MsgBlaster because I wanted a few more features such as the ability to unsubclass controls on the fly. I called this new version ModBlaster (ok, not very original) to avoid confusion with the original version. I've supplied the changed source code and leave it up to you to get the MSDN CD for more information.

I also subscribed to "Microsoft Systems Journal" and "Windows Tech Journal". Articles from both magazines appear on the MSDN CD but it's nice getting them a lttle sooner and in printed form (after all you can't sit and read the CD everywhere you might want to). One "Windows Tech" article in particular helped me understand how to fake a title bar when the window has none (see "Sprucing Up the Standards" by Richard Levaro, July 1993). And I should mention that the "Visual Basic Programmer's Journal" is critical for keeping up with the latest VB news, tips, and events.

Get Pizazz!

Finally, I have to plug a little custom control that I wrote called Pizazz. Pizazz.vbx solved several problems for me. First, as much as I loved VB, I hated the way most of my projects would end up with half a dozen VBX's, including stuff like spin.vbx and threed.vbx, just to get a decent interface, and I still wouldn't have a tab control. And most controls are not graphical, which means every control you put on the screen consumes extra windows resources. I would have a form with a bunch of 3D panels from threed.vbx and put picture boxes with loaded bitmaps inside them and before I knew it I was out of resources.

So what does Pizazz do? First it has both a graphical (PZLabel) and a windowed (PZPanel) control which are virtually identical. This allows the developer to pick and choose graphical or windowed controls as best suits their needs. In general, I try to build forms using mostly the graphical controls because they use less resources. A good example of this is the BevelOption

form in theTime which has 3D panels, tabs, spinners, option buttons, and regular buttons all built from graphical PZLabels. And only one VBX is required!

Another problem solved is flicker free painting. Most controls paint directly to the screen device context (DC). While this is easy to do, it can cause really annoying flicker under certain circumstances. If you're into building user friendly interfaces, you should avoid flickering output, especially if this is a multimedia app with lots of colors and textures. You will not see a Pizazz control flicker because the output is first drawn to a memory DC and then "bitblted" to the screen DC. Search the MSCD for "flicker free output" for more info.

Speaking of textures, the ability to tile a bitmap allows Pizazz controls to be totally filled with textures. Textures are in and you can't use them effectively if you can't tile. So just say "NO" to any picture type control which can not tile bitmaps.

Pizazz costs only $15 and can be ordered directly from Compuserve (GO SWREG). The registration ID is 6551. See the Pizazz.hlp help file for more information.

Titlebars? We don't need no stinking titlebars!

The first problem I solved was letting the user add and remove the title bar and borders by double clicking on TheTime's window. You may have noticed that a VB form can't normally change it's border property and/or remove the title bar and system control box at run time. And of course I needed a way for the user to move the window around when the title bar wasn't there anymore.

Adding and removing the borders at run time was easy to accomplish by changing the window's border style using the SetWindowWord API function. Here's a snippet of code to do that:

```
Dim l As Long
Dim Border as Integer
If Border Then
   l = WS_OVERLAPPEDWINDOW Or WS_VISIBLE
Else
   l = WS_VISIBLE
End If
l = SetWindowLong(hWnd, GWL_STYLE, l)
```

Moving the window without a title bar was a little more difficult to do. My original solution was to write a fair amount of code to support outlining and moving the window in the same manner that Windows does when application's title bar is dragged. This worked ok but I knew there had to be a better way.

This is where the ModBlaster custom control that I derived from the MsgBlaster control on the MSDN CD started to come in handy. With ModBlaster I was able to capture all the form's mouse events. I specifically trapped the form's WM_NCHITTEST message but and returned HTCAPTION if there was no border and subsequently no title bar. Doing this effectively turns the entire client area into a title bar, so clicking and dragging the client area moves the window! And the code is so simple, check it out...

```
Select Case MsgVal

   Case WM_NCHITTEST
      ' if there's no title/border and the click is in the client
      ' area then change it into a title bar click so the window
      ' can be moved by clicking and dragging it
      If ReturnVal = HTCLIENT And Not Border Then
```

```
        ReturnVal = HTCAPTION
    End If
```

I also used the ModBlaster control along with a few Windows API calls to support the drag and dropping of bitmaps, and to add a submenu to the form's system menu, and to provide a few other features that would otherwise have been pretty darn difficult to implement in pure VB code. I leave it up to you dear reader to examine the supplied source code and figure it all out.

Later...

That's all. Have fun storming the castle! And don't forget to get a registered copy of Pizazz.

Ben Jones
Visual Bits Software
P.O. Box 243
Watertown, MA 02272

CIS: 70402,3651
INTERNET: 70402.3651@compuserve.com