Developing Embedded Visual Basic 3.0 Applications for Win CE 3.0 and the Pocket PC

Jonathan C. Mull[1]
and
Kyle Lutes[2]
Computer (Information Systems) Technology Department, Purdue University
West Lafayette, IN 47907-1421 USA

Abstract

With the explosive proliferation of mobile computers, an education in modern computer programming languages and techniques should include exposure to application development for them. This paper's authors chose to do an independent study on eMbedded Visual Basic, a programming language being used with the WinCE operating system. One element of the study was participation in a Pocket PC programming contest sponsored by Microsoft, which concluded April 8, 2001. This paper discusses our experiences researching mobile computing platforms, learning eMbedded Visual Basic, and developing and submitting an application for the contest.

Keywords: eMbedded Visual Basic, Pocket PC, Windows CE, programming contests

1. OVERVIEW OF THE POCKET PC

According to the Gartner Group, sales of mobile computers in the United States will increase 300 percent to about 28 million over the next four years (Thornton 2001). Pocket PC is Microsoft's platform for the personal digital assistant (PDA) market. It currently runs version 3.0 of the WinCE operating system. Analysts predict that by 2004, the Pocket PC will own as much as 40 percent of the market, with Palm falling to 45 percent (Thornton 2001).

WinCE 1.0 was released in 1996. Numerous devices, including the Sega Dreamcast, support an improved version, 2.01, released in September 1997. The current version, 3.0, was released in Summer 2000. WinCE 3.0 can be considered Windows 2000 "light". It has sophisticated memory management, a scheduler for multithreaded programming, and most features of a modern OS (Yao 2001). It does lack the ability to process 16-bit applications, and support for a page file, but many familiar 32-bit APIs are readily available to the developer (Yao 2001).

On the horizon for WinCE is version 4.0, code-named "Talisker". It will use Kerberos to be more secure, will have support for some .Net services, Bluetooth for wireless communication, Universal Plug and Play, USB, better multimedia with DVD, and SSL (Arar 2001).

2. CONTEST OVERVIEW

The Microsoft Pocket PC Programming Competition was open to university students (enrolled as of Dec. 1, 2000) whose school is a part of the Microsoft Research University Relations Student Consultant program. Students need to register with the Microsoft Student-Dev.org group for their school. Anyone working directly or indirectly for Microsoft, and the university student consultants were ineligible to participate. The contest entry deadline was 11:59 pm PT April 8th, 2001.

Participants were asked to develop an original application for the Pocket PC to be judged on the following criteria:

- Innovation and Creativity – 30%
- User Interface Design – 30%
- Stability and Performance – 15%
- Wireless and Mobile usage – 15%

[1] mulljc@hotmail.com
[2] kdlutes@tech.purdue.edu

- Use of Exclusive PC Features (i.e. Activesync integration) – 10%

Prizes are to be awarded at both the school and national level. The winning team from each school will receive a Pocket PC device, valued at $500. At the national level, the winning five entries win cash prizes ranging from $5,000 for 5th place to $25,000 for 1st place. Those five winners also receive a trip to Redmond, Washington to meet with Microsoft recruiting, research, and/or the Pocket PC development team. Coach airfare, two nights lodging, and some local transportation are included. Winners will be notified by mail/e-mail on or about May 28, 2001. Complete contest rules are available online at http://acm-studentdev.cs.purdue.edu/pocketpc/rules.htm.

3. APPLICATION DESIGN CONSIDERATIONS

The most obvious restriction to developing applications for the Pocket PC is the small user interface. Pocket PC supports a screen size of 240 x 320 pixels, which is less than 1/4th the size of the minimal desktop PC screen size of 800 x 600 pixels. The use of tab strips, form show/hide techniques, and scrolling make interaction with size limitations fairly easy to solve for both developers and users. The touch screen interface has familiar events like “mouse up”, “mouse down”, “click” and “double click” to aid interaction. The small window on the device does support color, and respectable graphics. Pocket PC does not recycle memory space for forms that are shown and subsequently hidden or are no longer used. As a result, memory considerations quickly become an issue if an application uses numerous forms or graphics.

Each WinCE process garners 32MB of address space, significantly less than the 2GB for the typical desktop Windows application. Presently, the Pocket PC has only 16 to 32 MB of RAM available, so the hardware may be a bottleneck for memory hogging applications. Additional memory is available by using shared memory from a one gigabyte-sized address space.

Obtaining input from users can be accomplished through the use of the stylus and the Soft Input Panel (SIP) keyboard function, the SIP HWX handwriting recognition functionality, or keyboard input while Activesync-ed to the desktop. Because there typically is not an attached keyboard, it is beneficial to automate data entry and reentry with dropdown lists over combo-boxes, and help users complete phrases to minimize stylus character input.

4. DEVELOPMENT TOOLS

There are two development tools available for WinCE 3.0 -- Microsoft Visual Tools 3.0 and Microsoft Platform Builder 3.0. Embedded Visual Tools is freely available at http://www.microsoft.com/mobile/downloads/dev_all.asp. Information about obtaining Platform Builder is available at http://www.microsoft.com/windows/embedded/ce/tools/pb30faq.asp. Within Visual Tools 3.0, the developer can choose between eMbedded Visual C++ (eVC++) and eMbedded Visual Basic (eVB). Platform Builder, formerly Windows CE Embedded Toolkit, is used to create custom configurations of the CE operating system itself, and is needed by developers wishing to embed CE into a device like a digital camera or cell phone. Visual Tools 3.0 looks very much like Microsoft Visual Studio 6.0, and functions very similarly as well. Visual Tools not only allows the developer step through code on a linked device, which can be very slow, it also includes an emulator to run on the desktop. The emulator is indeed faster, and works with Windows NT 4.0 or Windows 2000, but not Windows 95, Windows 98, or Windows ME.

eVC++ is the more powerful of the two visual tools. There are very few differences between the functionality of eVC++ when compared to Visual C++ for Windows. eVB includes an integrated development environment (IDE) similar to that of Visual Basic 6.0 for Windows (VB), but the programming language is more closely related to VBScript than it is to VB, which means it is much less powerful and robust.

5. EMBEDDED VISUAL BASIC

eVB in the IDE is in appearance very much like VB 6.0. However, the similarities don’t last long. All variables in eVB are of type Variant, which can accept values for any data type (string, double, boolean, etc.). Unfortunately, object-oriented programming techniques, and user-defined types (UDTs) are not supported as they are in VB. The lack of these and other expected modern programming language features severely limit interaction with the WinCE API functions. For example, the ability to access the Pocket PCs Voice Recorder functions. eVC++ can be used to create custom DLLs to circumvent this problem.

eVB does use functions and methods like one would expect, and supports the use of dynamic arrays. File access is handled with an ActiveX control rather than direct code, and common dialog boxes are supported to provide a desktop feel. eVB can access data sources using ActiveX Data Objects for WinCE (ADOCE), and the Pocket PC supports the Pocket Access database. Debugging in eVB is also more difficult than with VB. eMbedded VB does not have a good exception handling mechanism, but the developer can still determine if a run-time error occurs by coding On Error Resume Next

before any statements that might throw errors, and testing the Err object's Number property immediately to see if anything has gone wrong. See Figure 1. This technique is crude but effective.

With its drag-and-drop interface development, common ActiveX controls, and the benefit of a familiar IDE, eMbedded VB is robust enough to develop applications easily, though lacking in advanced functionality. The biggest issues with development are ferreting out subtle syntax differences, and learning to interact with specific API references, such as Pocket Outlook's Object Model (POOM) (Tirschwell 2000).

```
'Continue processing code if an error
'occurs
On Error Resume Next

lFilename = App.Path & "/data.pck"

'This line will change the Err.Number if
'lFilename cannot be opened
File1.Open lFilename, fsModeInput
lHeadArray = File1.InputFields(5)
File1.Close

If Err.Number <> 0 Then
'An error has occurred, do error
'processing
    MsgBox
      "An Error has occurred.", vbOKOnly
    File1.Close
    Exit Sub
End If

'Resume system exception handling
On Error GoTo 0
```

Figure 1. File Input using eVB with Exception Handling

6. POCKET CLOCKIT

The concept for Pocket ClockIt arose from students' discussion of the Pocket PC contest. The Pocket ClockIt application aims to help PDA users log hours worked on specific tasks. This could be attorneys logging phone consultation time, programmers tracking time spent coding, engineers billing time spent between concurrent jobs, etc.

The Pocket ClockIt user-interface is relatively simple. See Figure 2. It employs the Timer control, which runs code every *x* milliseconds, to count up elapsed time and integrates it with Microsoft Pocket Outlook data existing on the Pocket PC. After logging a time block, the user can round the time spent to a range of intervals (5 minutes to 60 minutes), and set data about the activity during that time, any rate of payment information, and general comments. The user can then add this time log to new or existing Outlook appointments, tasks or contacts by adding data to the item's note property. Outlook appointments, tasks, or contacts can be created from within the Pocket ClockIt application. New Pocket Outlook items are created with common dialogs that Pocket PC uses to create new items. When Outlook is Activesync-ed to the desktop, any notes added to appointments, tasks, and contacts will be integrated with their desktop countertops.

The key elements to making Pocket ClockIt work were learning to use the file object, the Pocket Outlook Object Model, and the syntax of everything in between. The file object is an ActiveX control, and was easily included in the project, and most syntax is close to, if not identical to VB. Learning to use the POOM was a bit more challenging.

To access the POOM, one must logon to a POOM session. After which, the default folders storing tasks, appointments, and contacts are available. The properties for each item are manipulated by adding a collection of POOM items to an array, making changes, and calling the item's Save method to write them back to the default folder.

This study's greatest failure with Pocket ClockIt centered around the Pocket PC's Voice Recorder function. As mentioned earlier, eVB does not support UDTs or object creation, but it can use existing objects. The Voice Recorder, which was in the initial Pocket ClockIt design, would have allowed users to paste audio clips into their Pocket Outlook notes along with the time log data. The Voice Recorder API requires the use of a UDT, and as the deadline for entering the contest loomed, the authors had yet to learn to use C++ to create the .DLLs that are necessary to connect to the voice recorder API. The help files and developer message boards used throughout this independent study did not point in this direction, instead highlighting the ability to include existing .DLL files. Though able to point the application at the Voice Recorder's .DLL file, the necessary variables defined to interface with it do not natively exist in eVB. Hence, inclusion of the Voice Recorder functions in Pocket ClockIt version 1.0 was scrapped though they will be included it in any subsequent versions.

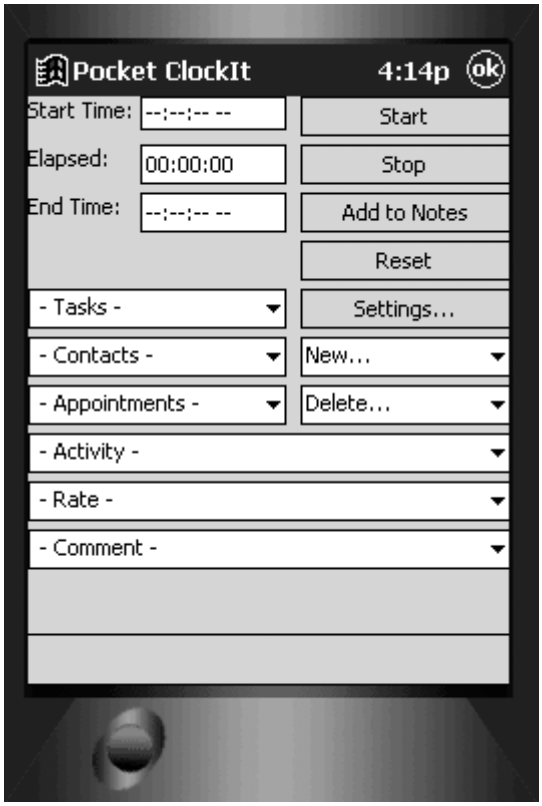

Figure 2. The Pocket ClockIt User Interface

Once the project was completed, submission to the Pocket PC contest was elementary. Student entrants needed only email to the StudentDev.org coordinator at their school an entry consisting of:

- The .vb file, or precompiled code to run on Pocket PC
- Any required .dll files
- Program documentation (i.e. screenshots and descriptions)

7. CONCLUSION

Entering the Pocket PC programming contest was a good experience, despite the sometimes-difficult task of learning new syntax. Visual Tools 3.0 (eVB and eVC++) are tools that current developers will be comfortable working with, though they are new and have limitations.

Should eVB be taught in the college classroom? Perhaps in conjunction with a Visual Basic 6.0 course, as an example of a related language used on a different operating system. It could also be taught in conjunction with eVC++ in a course on WinCE software development. But as a stand-alone language, eVB is just not rich enough to warrant an entire semester course.

Developing applications for mobile computers is a relatively new field of study, and the maturity of development tools and languages will only come with time. The influence of mobile computers on the future of IT cannot be denied, and curriculum teaching students to develop for the desktop, LAN, WAN, and WWW should begin to include the languages of handheld devices as well.

8. REFERENCES

Arar, Y. (2001, February 6). "Microsoft Pushes Windows for Non-PC Devices." PC World. Retrieved April 17, 2001, from the World Wide Web: http://www.pcworld.com/news/article/0,aid,40582,00.asp

Thornton, C. (2001, February 23). "Palm vs. Pocket PC." PC World. Retrieved April 17, 2001, from the World Wide Web: http://www.pcworld.com/features/article/0,aid,41466,pg,1,00.asp

Tirschwell, M. (2000). "Programming with eMbedded Visual Basic and the Pocket Outlook Object Model." Retrieved April 17, 2001, from the World Wide Web: http://www.microsoft.com/mobile/pocketpc/stepbystep/evbpoom.asp

Yao, P. (2001, January). "Windows CE: eMbedded Visual Tools 3.0 Provide a Flexible and Robust Development Environment." MSDN Magazine. Retrieved April 17, 2001, from the World Wide Web: http://msdn.microsoft.com/msdnmag/issues/01/01/CETools/CETools.asp