THE VISUAL PROGRAMMER

GEORGE SHEPHERD AND SCOT WINGO

The first round of tools for Windows® CE 2.0 is out, and this month we take a look at Visual C++® for Windows CE 2.0. In addition to introducing you to the Windows CE development environment, we're going to port a freeware MFC-based grid control to the environment to show what you face in porting your MFC applications to Windows CE.

The Development Environment

Visual C++ for Windows CE is an add-on to Visual C++ 5.0. When you install the product, it extends Visual C++ by adding the following features:

- An Intel-based Windows CE emulation environment.
- New targets for each of the Windows CE-supported processors (MIPS/SH and the emulation environment). **Figure 1** shows the new configurations and toolbars added by the environment.
- New AppWizards for Windows CE-based applications.
- A Windows CE-compatible port of MFC (version 4.20 with some modifications).
- A Windows CE-compatible port of ATL.

Visual C++ for Windows CE 2.0 also supports the older 1.0 and 1.01 versions of Windows CE. This article focuses on the Windows CE 2.0 support, but most of the features are also available for the earlier editions of Windows CE.

What's New?

Windows CE 2.0 adds many of the features missing from Windows CE 1.0 that make life easier for developers, such as COM and ActiveX™ support and many of the Win32® APIs. These make the tools much more usable. In fact, Visual Basic® now supports Windows CE; we'll take a look at that environment in our next column.

The Windows CE emulator is a really cool version of Windows CE that runs on your desktop. It gives you the convenience of being able to run and test your applications immediately without having to download every time to a real Windows CE-based device. Of course, to make sure your applications work correctly you still need to test with real devices, but the emulator takes much of the pain out of the early debugging stages. **Figure 2** shows the emulation environment.

Four Windows CE-specific AppWizards ship with Visual C++ for Windows CE:

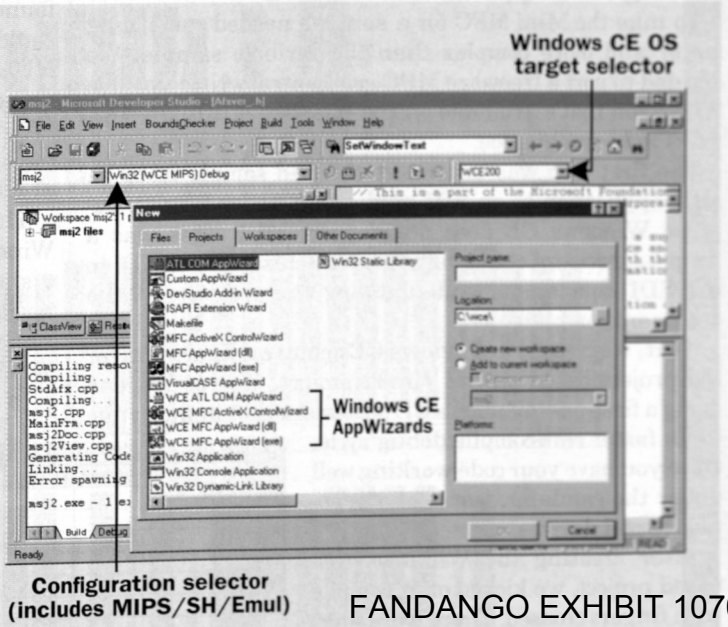
- WCE ATL COM AppWizard
- WCE MFC ActiveX ControlWizard
- WCE MFC AppWizard (DLL)
- WCE MFC AppWizard (EXE)

They are basically the same as their Win32 counterparts, except that they have different features that take advantage of the Windows CE operating system, such as the Windows CE help environment. **Figure 3** shows a page from the MFC executable project with some Windows CE-specific options.

Mini MFC

The version of Mini MFC that ships with the Visual C++ for Windows CE 2.0 has many more features than the previous release of the toolkit. The previous version supported only a handful of the core classes. This has been expanded, and some new classes have been added above and beyond those available in the standard MFC. To get a feel for which MFC classes are supported see **Figure 4**, which is a hierarchy chart. The green items are not supported.

Three categories of classes have been added to Mini MFC for Windows CE: command bars, object store, and socket classes. Command bars are a hybrid menu/toolbar that is part of the Windows CE common control library and wrap-



George Shepherd is a Senior Software Engineer at Stingray Software where he develops Visual CASE for developers using MFC and OLE. Scot Wingo is cofounder

ped by functions in the Mini MFC CFrameWnd class. The bars at the top of **Figure 2** illustrate the typical Windows CE command bars.

Instead of implementing a file metaphor, Windows CE provides an object store instead. Several new MFC classes were added to give the developer access to the object store:

- CCeDBDatabase encapsulates a database in the object store.
- CCeDBEnum enumerates the databases in the object store.
- CCeDBRecord provides access to a record in the database.
- CCeDBProp encapsulates a database property. Database properties are data items that consist of an application-defined property identifier, a data type identifier, and the data value.

In addition to these data store classes, a new class, CCeSocket, implements an asynchronous CSocket derivative.

For more information on Windows CE and the various development tools, see http://www.microsoft.com/windowsce.

Now that you've seen an overview of the Mini MFC, let's put the environment through its paces by porting a project from Win32 to Windows CE.

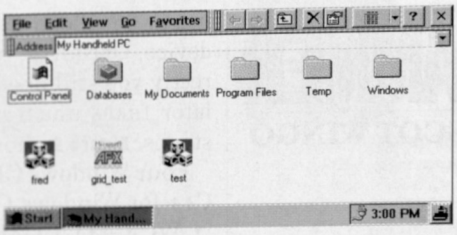


Figure 2 *Emulation Environment*

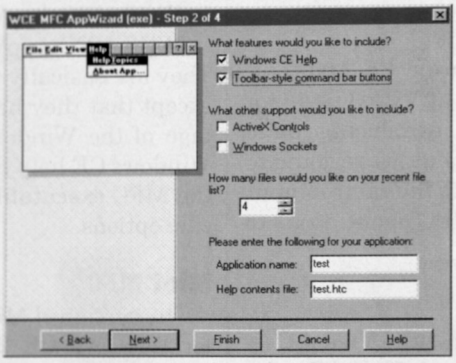


Figure 3 *Windows CE-specific Options*

Moving MFC Apps to Windows CE

To take the Mini MFC for a spin, we needed something larger and more complex than the Scribble sample. We decided to port a freeware MFC grid control written by Joe Wilcoxson that's available on the Internet at http://www.users.aol.com/chinajoe.

The first step was to convert the grid sample from an MDI application to an SDI application under Win32, because Windows CE does not support MDI. This was a simple exercise of changing some base classes from MDI to the SDI equivalents and updating the creation of the document template.

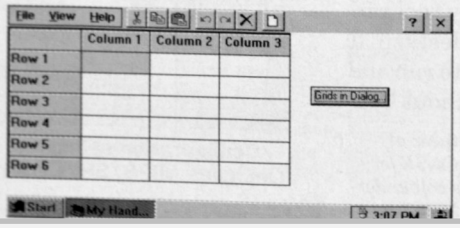
Next, we created a Windows CE debug configuration for the project based on the Win32 project. We recommend doing a first-pass Windows CE port under the emulator due to its faster edit/compile/debug cycle. Once you have your code working well under the emulator, test it on the actual target devices.

After creating the Windows CE-based project, we kicked off a compile with fingers crossed. There were literally hundreds of errors caused due to

- Windows CE requires Unicode strings. The freeware grid does not use Unicode strings, so we had to convert them. This included using the_T() macros, switching to the Unicode declarations for strings, and so on. This was by far the largest problem and took a while to build and test. We decided to go ahead and get the Unicode builds up and running on Win32 and then test again on Windows CE.
- Many of the printing APIs used by the freeware grid (such as IsPrinting and BeginPrinting) were not supported by Mini MFC. We #ifdefed these printing APIs out until printing is supported.
- Property sheets are not supported in Windows CE. The freeware grid sample uses numerous property sheets so we had to remove these parts of the samples.
- The freeware grid uses many cursor control samples that are not supported by Mini MFC. We disabled this functionality with #ifdefs.

After working around these issues, we were able to get a clean compile of the freeware grid and its sample. Then we ran it to see how it performed under the emulator. It started to draw slightly and then immediately ASSERTed. When we dug into the code to find the cause of the ASSERT, we found that several of the font metric calls were failing. We couldn't find a valid reason for the font metric calls to return NULLs, so we stubbed out the calls and put in hardcoded values. This article was written with a beta version of Windows CE 2.0 as well as beta versions of the development tools, so this problem will most likely be fixed in the production versions of the OS and tools.

The modified code that has been ported to work on Windows CE 2.0 is available from the *MSJ* Web site. **Figure 5** shows the control in action. The cursor handling still needs some improvement, but the control has largely the same functionality under Windows CE as it does under Win32. ◆



To obtain complete source code listings, see the MSJ Web site at http://www.microsoft.com/msj.

Have a question about programming in Visual Basic, Visual FoxPro, Microsoft Access, Office, or stuff like that? Send your questions via email to visual_