APPENDIX **B**

Installing DirectX and Using Visual C/C++

IN THIS APPENDIX

- Installing DirectX
- Using Visual C/C++

Installing DirectX

The most important parts of the CD-ROM that you must install are the DirectX SDK and runtime files. The installation program is located within the `DIRECTX\` directory along with a `README.TXT` file explaining any last-minute changes.

> **NOTE**
>
> You must have the DirectX 8.1 SDK or better to work with the source from this book, so if you're not sure that you have the latest files on your system, run the installation and it will tell you. However, I suggest using the latest version, DirectX 9.0, which is the SDK that I used for the final compilation of this book.

When you are installing DirectX, pay attention to where the installer places the SDK files, because you will have to point your compiler's Library (.LIB files) and Include (.H files) search paths appropriately when you wish to compile.

In addition, when you install the DirectX SDK, the installer will also ask whether you want to install the DirectX runtime files. You need the runtime as well as the SDK to run the programs. However, there are two versions of the runtime libraries:

- Retail—This version is the full release retail consumer version that you would expect a user to have. It's faster than the debug version. You can always install this one on top of the debug version later if you want.
- Debug—This version has hooks for debugging, and is what I suggest you install for development. However, your DirectX programs will run a little slower.

TIP

Note to Borland users (if there are any left): The DirectX SDK does have Borland versions of the DirectX .LIB import libraries. You will find these in the `BORLAND\` directory of the DirectX SDK installation. You must use these files when compiling. Also, make sure to go to Borland's Web site and read the README.TXT in the `BORLAND\` directory for any last-minute hints on compiling DirectX programs with Borland compilers.

Finally, DirectX is continually updated by Microsoft, so make sure you frequently visit their DirectX area at:

```
http://www.microsoft.com/windows/directx/default.asp
```

Using Visual C/C++ Compiler

First off, I have received countless emails in recent years from people that don't know how to use the C/C++ compiler. I don't want a single email about compiler problems unless blood is squirting from the screen and the computer is speaking in tongues! Every single problem was a result of a "rookie" compiler user. You simply can't expect to use a piece of software as complex as a C/C++ compiler and not read the manual—right, Jules? So please do so before trying to compile programs from this book.

Anyway, here's the deal about compiling: I used MS VC++6.0 for this book, so everything works perfectly with that compiler. I would also imagine that VC++ 5.0 should work, too, but don't hold me to that. If you have a Borland or Watcom compiler, they should work also, but you might have to do a bit of work getting the right setup to compile. My suggestion is save yourself the headache and get a copy of VC++ for $99. The bottom line is that the MS compiler is the best for Windows/DirectX programs—it just makes everything work better. I have used Borland and Watcom for other things, but for Windows applications, I don't know many professional game programmers that don't use MS VC++. It's just the right tool for the right job.

Additionally, the latest Microsoft C/C++ .NET compiler will work fine as well. However, please read the compiler manuals, because the initial setup is slightly different, along with some of the dialogs, compared to VC++ 6.0.

Compilation Hints

Here are some hints to set up the MS VC++ compilers; other compilers are similar.

- Application type—DirectX programs are Windows programs; more accurately, they are Win32 .EXE applications. Hence, set your compiler to Win32 .EXE application for *ALL* DirectX programs. If I tell you we are making a CONSOLE application, set the compiler for CONSOLE application. Also, I suggest that you make a single Workspace and compile all your programs in it.

- Search directories—To compile DirectX programs, there are two things the compiler needs: the .LIB files and the .H files. Now, listen up—set both search paths in the compiler/linker options to search the DirectX SDK Library (.LIB files) directory and Include (.H files) directory, so the compiler can find the files during compilation. However, this isn't enough! You must also make absolutely sure the DirectX paths are *FIRST* in the search tree. The reason for this is that VC++ comes with an old version of DirectX, and you will end up linking DirectX 3.0 files if you aren't careful. Furthermore, make sure that you *MANUALLY* include the DirectX .LIB files in your projects. This means that you want to see `DDRAW.LIB`, `DSOUND.LIB`, `DINPUT.LIB`, `DINPUT8.LIB`, and so on in the project! Very important!!!
- Error level setting—Make sure you turn the error level on your compiler to something reasonable, such as level 1 or 2. Don't turn it off, but don't put it on full blast. The code from the book is professional-level C/C++, and the compiler will think that there are a lot of things that I didn't want to do—believe me, I wanted to do them.
- Typecast errors—If the compiler gives you a typecast error on a line of code (VC++ 6.0 users beware), just cast it! I have received lots of emails from people that don't know what a typecast is. If you don't, look it up in a C/C++ book. I will tell you that some of my code might have slipped through without an explicit typecast here or there, and VC++ 6.0 could possibly get belligerent. If you get one of these errors, look at what the compiler expects and just put it in front of the `rvalue` in the expression, and that will fix it.
- Optimization settings—We aren't making full-release products yet, so don't set the compiler to a crazy optimization level—just set it for standard optimizations that prefer speed to size. Within VC++, the optimization dialog can be found in the Project, Settings dialog in the C/C++ tab, under the category of Optimizations.
- Threading models—Ninety-nine percent of the examples in the book are single-threaded, so use the single-threaded libraries. If you don't know what that means, read the compiler book. However, if I do use multithreaded libraries, I will tell you. To compile the examples from the chapter on multithreading, for example, you must switch to the multithreaded libraries. Within VC++, the threading model dialog can be found in the Project, Settings dialog in the C/C++ tab, under the category of Code Generation.
- Code generation—This controls what kind of code the compiler generates. Set it for Pentium Pro. I haven't seen a 486 except in ancient tomes, so don't worry about compatibility. Within VC++, the target processor settings can be found in the Project, Settings dialog in the C/C++ tab, under the category of Code Generation.
- Struct alignment—This controls the "filling" between structures. Pentium X processors like things in multiples of 32 bytes, so set the alignment as high as possible (16 bytes is currently the maximum supported by VC++). It will make the code a bit

bigger, but a lot faster. Within VC++, the target processor settings can be found in the Project, Settings dialog in the C/C++ tab, under the category of Code Generation.

- Compiler updates—Believe it or not, Microsoft updates everything, and I mean everything—even their compilers! Therefore, make sure you have downloaded and installed the latest service pack for VC++ or .NET from MSDN at `http://msdn.microsoft.com/`.
- Finally, when you are compiling programs, make sure that you include all the source files referenced in the main program. For example, if you see me include `T3DLIB1.H`, chances are there's a `T3DLIB1.CPP` that needs to be in the project.