Microsoft C++ and Visual C++ Review

C. H. Lochmueller and Charles Reese

Department of Chemistry, Duke University, Durham, North Carolina 27705–0354

Received November 4, 1993

Microsoft Visual C++ (MSVC) is a major upgrade from the earlier MS C++ 7.00 in terms of a development environment. MS C++ 7.00 was a DOS based system while MSVC is a true Windows program. The development environment consists of two major components, the Visual Workbench and the AppStudio. The Visual Workbench is the host program for nonresource development and AppStudio is used for the development of program resources. MSVC includes the Windows SDK and an improved version of the MS Foundation Class Library. It is supplied on 20 3.5-in. high density disks. This is the Professional Edition of the compiler which allows development of DOS as well as Windows programs. The Standard Edition version can only be used for Windows program development. Installation, following the instructions in the manual, was without incident. The hardware used was a VCOM 486DX2–50 PC equipped with 8 Mbytes of RAM and a 310 Mbyte hard disk. A full installation of the Professional Edition requires 50 Mbytes of disk space.

The hard copy documentation supplied with the software did not include documentation for the Windows SDK which did come with MS C++ 7.00. Documentation for the Windows SDK is available separately and may be useful for Windows developers who do not already have it. Extensive use was made of the hard copy documentation and the on-line help system. The documentation is very comprehensive for the classes supplied in the MFC library with clear explanation of the syntax of each of the public variables and functions of each class. No errors in the documentation were noted for the classes used by the author. Also, working example code is given for almost all of the public functions; this was very useful for getting the usage correct when first learning the language.

The Visual Workbench is the code development resource. It has an impressive array of features. It provides the editing environment which is very easy to use. Text is entered just as in any word processor but keywords are automatically converted to colored text as they are entered. This is makes the code easy to read and reduces the number of errors made. The editor is closely integrated with the on-line help system. Selecting a keyword or a library command and pressing the help key (F1) takes you directly to the on-line help for that item. Code fragments from the on-line help can be copied and pasted, making it a sure thing to get the syntax right. Also, when the compiler detects an error, double clicking on the error will take the user directly to the offending line.

Compiler and linker options are set using Windows dialog boxes. Most options are selected by clicking on a button or selecting an item from a list box. The compiler can build an impressive variety of project types. The list of project types include Windows EXEs, DLLs and COMs, static libraries (.LIB), Visual Basic Custom Controls (.VBX), and DOS programs. The compiler also has a large number of options to optimize speed or space in critical parts of code. The compiler supports the generation of code segments using P-code. P-code is partially compiled and then interpreted at run time to greatly reduce the space requirements of the code. It is used for non-time-critical areas.

Visual Workbench includes the AppWizard. The AppWizard generates all of the files and code needed as the framework for a new project. AppWizard options allow the user to specify all of the major Windows features that are to be supported. The programmer can select the major program options SDI or MDI, OLE client, printing and print preview, toolbar, status bar, and context sensitive help support. This skeleton application can be compiled and run. Of course it does not do anything, but it does provide a working frame to which the users code can be added.

The Visual Workbench also includes the Class Wizard which can be used to create new classes based on MFC classes derived from the CmdTarget class. These classes are the classes that can respond to Windows messages. The Class Wizard supplies a means of generating the message maps and message handlers for the Windows messages. The programmer selects the name of the class that is to handle a message and the message that is to be handled and the Class Wizard generates the appropriate message map entries and the skeleton functions to handle the message. The programmer can go directly from the Class Wizard dialog to the edit window at the location of the prototype code to add the application specific code.

The Visual Workbench hosts a resident debugger. Once an application has been built, it can be executed with the debugger with break points set to interrupt the execution at specific lines in the source code. The debugger has a number of trace, break, and watch options which can be used to monitor program execution and variable values. Many of the debugger control options are available on a toolbar which is very convenient. This Windows based debugger is a major improvement over the DOS based Code View system. One major improvement is the ability to use the Alt-Tab key to switch back and forth between the screen display of the application and the debugger. This allows one to see the errors in the screen rendering of the application.

The second major program supplied with MSVC is the AppStudio. This is a resource development tool similar to "Resource Workshops" supplied by other compiler vendors and was a major shortcoming of the earlier MS C++ 7.0 system. The AppStudio includes all of the development tools found in the other top line workshops. It supports drag and drop for almost everything including buttons, edit boxes, bit maps, and even menus. The Class Wizard can be accessed from the AppStudio, so a resource that can generate Windows messages, for example a button, can use the Class Wizard to connect to the code which is to handle its message(s).

There are many more tools which come with MSVC but which space does not permit one to describe. There is a Browser, library maintenance tools (DOS), profilers, heap walkers, memory spies, etc. There are also a couple of bugs in the compiler and linker. The compiler can "get lost" on certain

types of errors. This seems to happen most often when a closing brace is left out. The compiler will print out a large number of errors to the screen and then no longer respond to any command, including the "Stop Build" command. The only solution is to reboot the computer. Also, a bug was found in the linker options in that only around four or five extra external libraries can be specified. We talked to Microsoft about this and found that the problem is the linker options line is only allowed to be 255 characters long. Since the options line seems to be around 220 characters long before any libraries are added, this does not permit many additional libraries. The immediate solution to this problem was to use the library manager to combine the six libraries supplied by the tool box vendor into one library.

In general the authors were very impressed with this development system and believe it is a very good system for those just learning C++ programming as well as for experienced programmers.