PLAINTIFF'S EXHIBIT 2230 Comes v. Microsoft

CONFIDENTIAL

Windows 95 and Window NT 3.5 Portability Issues

1. **Memory Mapped Files**. In Windows NT, a memory mapped file is only accessible to processes that have called CreateFileMapping and MapViewOfFile for that particular file. In addition the file's memory region can be based at different virtual addresses in different processes. In Chicago, once a program creates a memory mapped file, that memory region is accessible to all programs. Thus, a Chicago memory mapped file is always at the same virtual address in all processes.

WordPerfect's coaching system needs the ability to save information captured from a "Windows Hook", to a memory block that is later accessed from a WordPerfect process. It is our understanding that the "Windows Hook" may be executed on whatever process is currently executing at the time the "Hook" is initiated. In Windows 95, it will be much easier to implement this functionality, because the address of a memory mapped file will be available to every process in the system. Consequently, the "hook" procedure can save the information to the memory mapped file address. In Windows NT 3.5, the "hook" procedure may be required to perform a CreateFileMapping and MapViewOfFile on each execution of the "hook" procedure. This would probably greatly degrade performance. Our Coaching system may not only not degrade gracefully, but would also degrade machine performance significantly.

2. **Memory Management in DLLS**. There is a difference in the way the SHARED data in DLLs is handled between Windows 95 and Windows NT 3.5. Initializing a SHARED variable with a pointer to another SHARED variable will work in Windows 95, but not in Windows NT 3.5.

It appears that it is easier to share memory between processes in Windows 95 than Daytona.

3. **Registry**. The registry file format is different between Chicago and NT. This means that it is not possible to do a RegSaveKey/LoadKey/RestoreKey from a Chicago machine to a Daytona machine, or vice/versa. Windows 95 provides some System Administration capability through system policies. We have not seen any information regarding system policies on NT.

Although we have not made a final decision on this, we were considering using this as a means of implementing system administration of registry settings. Whatever use we make of system policies would appear to be a problem for NT.

4. **Unicode and ANSI OLE**. Depending on the level you integrate into OLE 2.0 it has to be Unicode or ANSI. Between the two environments it becomes more difficult for code to operate the same way when we have to worry about ANSI or Unicode when interfacing to the OLE system. Under NT we have to supply Unicode strings for many APIs and for Windows 95 it has to be ANSI.

Thread safe OLE: We are planning on using "COM" extensively, and are concerned that this will be a problem since our applications may call us from multiple threads.

5. **Other Small Differences**. There will be numerous subtle differences that we will discover that have to be programmed around. For example we have already had seen a difference in our code that adds menu items. We modified the code to work on NT, and when we moved to Chicago we found that the code didn't work. This stems from the internal Unicode under NT to the ANSI API set under Windows 95. We were able to come up with an easy fix that worked on both platforms but there was effort involved. We found setting a global hook worked on NT, but brought Chicago to its knees. There will be differences in memory management, in addition to memory mapped files, that will require special attention.

6. **Relying on Windows 95 functionality to enhance product functionality.** In Windows 95 there exists potential to rely on the OS for certain functionality. For example, the Task Bar under Windows 95 influences the design of an SDI application because it provides for easy switching of tasks and switching between documents. If we were to rely on that capability and not put code in to switch between documents easily (even under SDI), our users would then not have the same experience with the product under Windows NT. Therefore, we may need to enhance the product simply because it runs under Windows NT to give it the same functional level as the product running under Windows 95. There could be many such instances with this type of impact in developing for both Windows 95 and Windows NT.

7. **Maintain two development and testing environments.** There will be additional effort required in maintaining two environments, testing, programming around subtle differences, etc.; costs such as having to purchase two different operating systems, buying hardware capable of developing for NT (not only for developers but for testing as well), and licensing of development tools.

8. **Issues that are potential problems.** Common controls are supposed to behave the same under both but we have not validated the current DLLs for consistency in behavior. Windows 95 help system running on Windows NT, same problem - consistency in behavior and functionality. Basically it requires a lot of validation to feel comfortable that Microsoft has handled all the issues of common sub-systems between the two platforms since they are not based on the same OS model.

9. **Development Environment.** You must use the Windows 95 SDK to get certain headers pertaining to the Common control set and Plug and Play messages, etc. VC++ 2.0 release does not have the changes in to support Windows 95 (headers). Under Windows 95 you must load the SDK and VC++ to get the environment setup for proper Windows 95 development. Patches are made available to keep VC++ 2.0 working under Windows 95.

While each of these issues can most likely be worked around it poses additional burden on the development process and requires more resources in order to get our code common between both platforms. We have even noticed that while we are developing we have behavior differences in the APIs themselves. We report these to Microsoft as we find them.

January 31, 1995

Microsoft Network (formerly Marvel)
General Notes

MS Network co-owned by TCI, 20%
Current partners; (partial list) Lotus, Symantec, Broderbund, Claris, Dell, Gateway, Packard Bell (no Compaq or IBM, yet)

Flexible business model
- Low cost of entry to partner with MS
  - ISVs own their area
  - Revenue split (currently)

	Novell	Microsoft
Cover charges	70%	30%
File Downloads	70%	30%
Subscriptions	70%	30%
Advertising	80%	20%
Physical Goods (MS takes and processes orders, collects $ through Network)	90%	10%

- Open platform (for development of forum)

Build a market
- Aggressive marketing (part of Win 95)
- Aggressive pricing (target $4.95 a month, 3 hrs. free, $2.95 or less an hour)
- Broad penetration goals (no duh, this is MS; goal at least 400k users first year)(will have 700 access points in U.S.; 90+% should have local call)
- Worldwide distribution (will be available in 35 countries, day one)

Win 95 Interface
- Standard APIs
- MS will provide tools and SDK for forum development
  - Blackbird (code name for dev. tool)
    - SGML based
    - Dynamic assembly of content (cool feature)
- Billing and support infrastructure built in
- Heavy OLE integration
  - Cool "shortcut" feature would allow us to put a Novell area button directly in our software. Essentially a pointer. "Online Enable" your application.
- MS Exchange is e-mail client
- Graphics use progressive rendering

Internet
- TCP/IP connection; not available in release, later, will be a flat rate
- Partnering with UUNET for Internet access
- Will offer web browser, FTP, USENET

Target audience
- PC Industry
- SOHO
- Families with Kids
- Enthusiasts