The popularity of DOS/4GW made Windows 95 game compatibility a little easier, but with higher stakes

devblogs.microsoft.com/oldnewthing/20230829-00

August 29, 2023

Raymond Chen

By far, the most popular so-called DOS Extender in the early 1990's was DOS/4GW. MS-DOS game compatibility occupied a very large portion of my time during Windows 95 development, so I saw a lot of DOS Extender banners, most frequently the DOS/4GW banner.

Now, you might wonder, "How did these games even run in Windows 95 if they came with a DOS Extender? Wouldn't the extender try to enter protected mode and fail, because Windows was already managing protected mode?"

The trick is that these extenders were really two programs bundled together. One was a protected mode server, and the other was a protected mode client library.

In the beginning, there was the Virtual Control Program Interface (VCPI), which was supported by expanded memory managers like EMM386. These expanded memory managers barely used protected mode at all: The only thing they cared about was getting access to memory above the 1MB boundary, so they set up some page tables in order to map extended memory into the expanded memory page frame, but did no other virtualization. MS-DOS ran in a virtual machine that had full hardware access, and the VCPI interface let an MS-DOS application say, "Hey, I'd like to take over complete control of the system now," and VCPI would say "Sure, no problem!"

The VCPI interface quickly faded in popularity because no protected mode operating system (like Windows 3.0 in enhanced mode) would let any program take over complete control of the system. You would basically be suspending the old operating system in order to let the MS-DOS program take over as its own new custom operating system. Windows 3.0 introduced a new interface called the DOS Protected Mode Interface (DPMI) which let MS-DOS programs request that their code execute in protected mode, but only in user mode. The DPMI provider remained in control of kernel mode.

Okay, so back to DOS/4GW. When it started up, the DOS/4GW extender looked around to see if a DPMI server was already running. If not, then it installed itself as the DPMI server. But if a DPMI server was already running, then it allowed that DPMI server to remain in charge.

The game communicated only with the DPMI client portion of the library, which it used to transition to 32-bit mode, allocate memory, and do all those 32-bit things that these 32-bit game wanted to 32-bit do.

In other words, we have the following block diagram:

	Inside Windows	Standalone
DPMI server (ring 0)	Windows	DOS/4GW DPMI Server

DPMI client (ring 3)	DOS/4GW DPMI Client
	⇅
Game code (ring 3)	Game

Both Windows and the DOS/4GW DPMI server implement the DPMI interface, so the DOS/4GW DPMI client used standard DPMI calls to communicate with both servers.

This was great for application compatibility, because if there was some issue with how the DOS/4GW client communicated with the DOS/4GW server, we just had to fix it once, and it fixed a lot of games. On the other hand, if the issue couldn't be fixed, it broke a lot of games.

High risk, high reward.

Miraculously, most games *just worked* despite running under a different DPMI server from what they were originally developed with. There were occasional issues with specific games. Popular ones include games which assumed that all memory was physical and games which assumed the interrupt flag was unvirtualized, but for the most part, things worked well enough that the remaining issues could be treated as app-specific bugs.