What were the intended uses of those icons in moricons.dll?

devblogs.microsoft.com/oldnewthing/20250505-00/?p=111143

May 5, 2025

Since Windows 3.1, the `moricons.dll` file contains, well, *more icons*. What were these icons for?

Windows 3.0 added the ability to run MS-DOS programs in a window. One of the preinstalled program icons was called *Windows Setup*, and if you ran it and selected *Set Up Applications*, it offered to search your hard drive for existing MS-DOS applications and "help run them more easily in the Windows environment." If you accepted this offer, it showed a list of recognized programs, and you could select which ones you wanted to set up.

The mechanism for recognizing programs was by searching for the names of well-known executables such as `123.EXE` for Lotus 1-2-3. Sometimes, there were multiple programs that used the same executable name. For example, `MAIL.EXE` was used by *PATHWORKS Mail for MS-DOS*, *cc:Mail for MS-DOS*, and *XcelleNet X/Mail for MS-DOS*. In that case, the program asked you, "Hey, like, I'm not sure which program this is. Can you tell me if it's any of these guys?"

Once it identified a program (possibly with help from you), it created a PIF file that described an MS-DOS configuration that worked best for that program, created a Program Manager group called *Non-Windows Applications* (if one didn't already exist), and created an icon in the *Non-Windows Applications* group which used the PIF file to launch that program.

In Windows 3.0, the icon for the program was just a plain gray icon that said "DOS".

In Windows 3.1, *Set Up Applications* gained a feature: In addition to setting up the PIF file, it also set an icon in Program Manager that represented the program. When you clicked it, it still ran as an MS-DOS program, but at least the icon was prettier.

Initially, these icons were placed in `PROGMAN.EXE`, but as the number of icons grew, it became clear that they needed their own home instead of squatting inside the Program Manager binary. So the icons started getting added to a DLL called `MORICONS.DLL`

because, well, they were *more icons*.

This `MORICONS.DLL` icon library has carried forward ever since. Windows itself created those Program Manager icons, and those Program Manager icons turned into shortcut files in Windows 95, and those shortcut files would then migrate forward as you upgraded Windows. In theory, the compatibility chain could have been abandoned with the introduction of 64-bit Windows since there was no upgrade path from 32-bit Windows to 64-bit Windows (clean installs only), and because 64-bit Windows didn't support MS-DOS programs any more, so you couldn't reinstall them onto your 64-bit Windows system.

I suspect this cleanup didn't happen because the focus of the 64-bit port was bringing Windows forward to 64-bit, and most of the porting was mechanical: You could port a large chunk of code without even understanding what it did. The focus was not "looking for old 32-bit code that we can delete."[1]

Furthermore, the entire DLL was only 12 kilobytes, and it didn't pose a security risk,[2] and there was a compatibility risk that somebody was still using those old-timey icons for their shortcuts. Better to let sleeping dogs lie and eat the 12 kilobytes.

Next time, we'll take a peek inside `progman.exe` and `moricons.dll` and enjoy the old-timey icons.

[1] Not that deletion didn't happen, but the deletion of 32-bit code was done when the mechanical porting was not good enough to get the code running. At that point, you had to make an engineering decision whether the effort spent reverse-engineering and debugging the 32-bit code was worth the benefit of having a 64-bit version available.

[2] Unless you consider "these icons are so ugly that if an attacker could get the icon to be displayed, it will cause lasting mental pain and anguish to everybody who sees it" to be a security risk.