In Windows 3.1 and Windows 95, what is a “grabber”?

devblogs.microsoft.com/oldnewthing/20231127-00

November 27, 2023

Raymond Chen

Twitter user @indiocolifa asked what “grabbers” were in Windows 3.1 and Windows 95.

Windows 3.0 Enhanced Mode introduced the ability to run MS-DOS programs in a virtual machine. This by itself was already quite an achievement, but it didn’t stop there. It also let you put the MS-DOS session *in a window*, and run it on the screen along with your other Windows programs.

This was crazy.[1]

Here’s how it worked.

The virtual display driver is the driver whose job it was to make each virtual machine think it had its own video card. When the program running in the virtual machine modifies video memory, the virtual display driver notifies another driver whose job it is to bridge the gap between the virtual machines and the graphical user interface. This bridge driver has a buddy application running on the Windows desktop, called `WINOLDAP`, one copy for each MS-DOS virtual machine.[2] The name `WINOLDAP` stands for “Window for Old Apps”, where “Window” referred to new-hotness GUI windows, and “old apps” referred to old and busted MS-DOS apps.

When `WINOLDAP` receives the notification that there was a change to video memory, it tells the “grabber” that the game was afoot.

Each video driver has its own grabber, short for “video frame grabber”. When called by `WINOLDAP`, the grabber communicates with its partner video driver to get the new video data, figure out what parts of the screen changed, and convert the raw video memory into Windows text and graphics calls in order to reproduce the contents in a GUI window. After obtaining the raw changed pages from the video driver, the grabbers try to be clever and do things like diff the previous and new screen contents to render only the parts that changed, and to convert large blank areas to simple `FillRect` calls instead of printing a lot of space characters. They also look for ways to re-use the old screen contents: If the screen appears to have scrolled upward one line, then instead of rendering the entire contents again, it did a `ScrollWindow` to move the existing pixels, and then painted the single new line at the

bottom. Of course, the effort to speed up rendering by minimizing the number of graphics rendering calls needs to be balanced against the extra cost of doing all the diffing and scroll detection, so there was a constant trade-off being made: The extra work you do looking for, say, scrolling optimizations, might end up costing more than it saved.

In Windows 95, I was the one responsible for the grabbers, and I made some improvements to text rendering performance, and a lot of improvements to graphics rendering performance. In particular, Windows 3.1 could not run graphical MS-DOS programs in a window, but Windows 95 could: The video driver folks were able to improve graphics virtualization, and my optimizations for graphics rendering made it possible to re-render the screen contents in real time.

Of course, if you had a low-end system, that real-time rendering might be only two frames per second, but that's good enough for a program that changes the screen relatively infrequently anyway, such as a program for printing greeting cards. You're not going to be playing DOOM in a windowed MS-DOS session.[4]

[1] Oh, and I should also note that this was 1990, so everything was written in assembly language.[3]

[2] We saw some time ago how that driver worked with the `WINOLDAP` program to feed the character-mode `Alt`+`Tab` interface.

[3] Also, we didn't have Unicode yet, so the code is littered with `IFDEF`s to support compiling the operating system for Western European single-byte character sets or for East Asian double-byte character sets.

[4] Even though you wouldn't want to play DOOM in a windowed MS-DOS session, that didn't stop people from doing it anyway, just to show it could be done.