How did Windows 3.1's virtual machine manager get the information to show in the text-mode Alt+Tab switcher?

devblogs.microsoft.com/oldnewthing/20211129-00

November 29, 2021

Raymond Chen

When I told the life story of the `SwitchToThisWindow` function, some people were curious about how the virtual machine manager knew about the current windows and their z-order, so that it could show them in the fake text-mode `Alt`+`Tab` switcher.

The virtual machine manager has a friend, namely the Windows program that provided the graphical user interface for the virtual machine, named `WINOLDAP`. It got that name because it is the program which provides the Windows user interface for old (MS-DOS) applications. If the MS-DOS virtual machine is running windowed, then this program's main window draws a graphical depiction of the contents of the virtual machine's screen. If the MS-DOS virtual machine is running full-screen, then this program's main window doesn't need to draw anything, but it is still there, ready to help out the virtual machine manager.

If you press the `Alt`+`Tab` hotkey while in a full-screen MS-DOS virtual machine, the virtual machine manager receives the hotkey and tries to emulate the graphical `Alt`+`Tab` user interface.

The text-mode `Alt`+`Tab` code asks the virtual machine scheduler to switch execution context to the Windows virtual machine, and then call it back once this has occurred.

In 16-bit Windows, the `PostMessage` function is special in that it is one of the few Windows function that could be called from a hardware interrupt handler. The callback function (now running in the context of the Windows virtual machine) saves the current virtual machine context, and then creates a new execution context that calls the `PostMessage` function, which from the virtual machine client's point of view looks like a hardware interrupt in the sense that the function is being called out of the blue, from an unknown context. After the call to `PostMessage` function returns, the callback function restores the original context and allows execution to continue normally.

The message is posted to the `WINOLDAP` window, and when it receives the message, it gathers information about the next window in the `Alt`+`Tab` list, as well as information about the user's color preferences. It then issues a kernel call into the text-mode `Alt`+`Tab`

code, passing this information along.

The text-mode `Alt`+`Tab` code uses the color information to render a text-mode version of the `Alt`+`Tab` user interface, showing the name of the window being proposed as the switch destination.

Program Manager
If the user completes the `Alt`+`Tab` sequence, and the destination is a full-screen MS-DOS virtual machine, then the text-mode `Alt`+`Tab` switcher performs a direct virtual machine switch to the target. Performing the switch directly avoids bouncing through the Windows desktop. It then returns control back to `WINOLDAP` with a return value that means "You're all done, do nothing more."

If the destination is not a full-screen MS-DOS virtual machine, then the text-mode `Alt`+`Tab` code returns control back to `WINOLDAP` with a return value that means "Switch to this window and return," and in response, `WINOLDAP` calls the `SwitchToThisWindow` function to tell the window manager to switch to the window in the style of `Alt`+`Tab`.

If the user continues the `Alt`+`Tab` sequence, either by tabbing forward or holding the `Shift` key to tab backward, then the text-mode `Alt`+`Tab` code returns control back to `WINOLDAP` with a return value that means "Get information about the next/previous window in the `Alt`+`Tab` order and call me back."

The communication between the text-mode `Alt`+`Tab` code and `WINOLDAP` basically takes the form of a captive thread, where the `WINOLDAP` function issues a kernel call which doesn't return until the user decides what they want to do next with the `Alt`+`Tab` sequence.

It's a rather complicated dance (all written in assembly language, which was the fashion at the time) to accomplish something that looks so simple and obvious to the end user.

Raymond Chen

Follow