computers are bad

https://computer.rip - me@computer.rip - fax: +1 (505) 926-5492

2021-07-21 the desqtop

I believe I have mentioned before that the history of early GUI environments for PCs is sufficiently complex and obscure that it's very common to run into incorrect information. This is markedly true of the Wikipedia article on DESQview, which "incorrects" a misconception by stating another incorrect fact. Since it's Wikipedia, the free encyclopedia that anyone can edit, I assume that if I correct it the change will be reverted by bot within seconds.

False claims about TopView aside, the Wikipedia article on DESQview makes most of the salient points about its history. That said, I would like to talk about it a bit because DESQview is a neat example of an argument I've made, and it happens to dovetail into another corner of GUI history that I'll bring up here and there.

DESQview was a multitasking GUI built by a company called Quarterdeck. It was released for DOS in 1985, so several years after Visi On, and right in the thicket of most of the DOS GUIs. DESQview is a GUI, though, only in the sense of the logical paradigm of user interactions. It actually runs in textmode, using the DOS extended ASCII box drawing figures to create windows and menus, and using letters and symbols as buttons. It's similar in this regard to the relatively modern Twin (Terminal Windows), and could be viewed as a souped up terminal multiplexer like tmux.

Despite running in textmode, DESQview has basically all of the WIMP (Windows, Icons, Menus, Pointer) behavior that we consider typical of a GUI. To be fair, by virtue of running in textmode it fundamentally lacks icons, but so did a number of other early GUIs that ran in graphics mode. Any one of us could sit down in front of a machine running DESQview and figure out the basic interactions without much trouble, something that can't be said of most terminal multiplexers. Here is an example of the philosophical divide between TUI and GUI, or more specifically between unguided and guided: terminal multiplexers like screen and tmux are unguided interfaces that expect the user to read the manual. More typical of the GUI, DESQview attempts to make most functionality fairly discoverable to the user.

So in that light, consider this sentence from the Wikipedia article: "DESQview is not a GUI (Graphical User Interface) operating system. Rather, it is a non-graphical, windowed shell that runs in real mode on top of DOS, although it can run on any Intel 8086- or Intel 80286-based PC."

It's not a *GUI*, it's a *non-graphical windowed shell.* It runs in real mode on top of DOS, which is true of basically all '80s GUIs including Windows. It has windows, and it's a shell, but it's not a GUI because it's non-graphical. To me, at least, this whole thing is a bit farcical. The desire here to exclude DESQview from the category of GUIs only serves to reinforce that the interaction concept that we refer to as the

“GUI” is actually quite divorced from the difference between text and raster displays. You can always employ ASCII art to pretend you have a graphical display, after all.

Another interesting component of DESQview to discuss is its support for DOS applications. We saw with Visi On that there is sort of a basic conflict involved in developing a DOS GUI: if it runs on top of DOS, users will want to be able to run their existing DOS software. But DOS software assumes full control of the machine and does not play well with multitasking. Visi On went the route of throwing DOS out the window and requiring that software be written specifically for Visi On [1]. DESQview went the opposite, more consumer-friendly route, of bending over backwards to work with the existing DOS stable.

DESQview had a significant leg up on this venture because its developer, Quarterdeck, had previously sold a DOS task-switcher called Desq. Task switchers are not really a familiar part of the modern computing landscape because of the ubiquity of multitasking operating systems. Back in the ’80s, though, most microcomputer operating systems were single-task and so the ability to run multiple programs at the same time could only be simulated. A task switcher created something like multitasking by doing exactly what it sounds like: switching out the tasks.

Specifically, Desq acted as a DOS TSR, or Terminate and Stay Resident. When launched, Desq installed an interrupt handler and then terminated. The interrupt handler fired when keyboard keys were pressed (remember at this point the keyboard on PCs was connected via the 8042 keyboard controller, which generated interrupts on each keypress). The interrupt handler could basically inspect each keyboard event and decide whether to act on it. In effect, a TSR could implement a “global hotkey.”

In the case of Desq, the hotkey resulted in Desq seizing control of the machine and stashing the contents of memory. It then presented a utility that allowed the user to select another task, which would be copied into memory and then jumped to. The effect was somewhat like switching windows, but you could only have one program visible at a time.

You might be wondering where that memory was stashed *to.* This gets into the peculiarities of x86 memory. By the time these task switcher utilities hit the scene, “extended memory” beyond the 1 MB real mode limit was fairly common on PCs. But, real-mode applications were unable to access this extended memory without putting in extra effort [2]. In practice, most DOS applications only ever used the real-mode-addressable memory, so task switchers could somewhat safely swap the first megabyte “basic memory” into the extended memory without the next application messing with it. Of course there was no guarantee, some applications did implement extended memory support and this generally made a program “incompatible” with task switching.

For Quarterdeck, DESQview was basically an extension of Desq, so it was natural to continue to support switching between conventional DOS applications. DESQview did much the same thing, loading and unloading DOS applications, but also using driver tricks to cause applications to “draw” text to their own windows. Like Desq, DESQview could “multitask” only the sense that it could react to interrupts, so the user was effectively “locked in” to the active window until the user triggered DESQview to seize control by use of a keyboard shortcut.

DESQview is an important example of a GUI system that is very much transitional between text and raster, and between TUI and GUI. Other similar examples include TopView, DOS Shell, and Norton Commander, the latter two of which were ostensibly file managers but grew to include a number of GUI features. Interestingly, though, DESQview appeared on the scene *after* the first text mode competitors. While raster mode has obvious advantages today for GUI software, there were huge additional challenges involved in using raster mode at this point in time. For one, it made compatibility with existing software extremely difficult.

Perhaps more importantly, though, the entire business computing world was on text-based machines, and text was mostly viewed as being perfectly sufficient. There just wasn't a lot of pressure to provide raster operating systems, because people hadn't really seen raster mode put to good use yet.

There are a couple of places to go from here, and you know that I will go to both of them: first, we will eventually need to get to the topic of Windows. I will probably discuss early Windows and TopView somewhat in parallel, because the comparison is interesting and because the competition of Windows and TopView represents yet another twist in the tumultuous partnership between Microsoft and IBM. In more of a fork, though, I will also start into a topic closely related to GUI history: network delivery of GUIs.

I said that DESQview dovetailed into another interesting topic, and it's network GUIs. DESQview was followed by DESQview/X... an X server. While this partially enabled the porting of X applications to DOS, it more importantly contributed to the first wave of thin client GUI systems.

[1] This isn't quite true, it actually is possible to run DOS applications under Visi On but with significant limitations that mostly prevented actually using the feature.

[2] If this sounds a bit amusing, keep in mind that we had basically the exact same problem years later with the 3-ish GB 32-bit limit. Memory beyond the first 3-ish gigabytes on a 32-bit machine could be used only if the application put in extra effort to support it (in that case by implementing PAE rather than XMS, the DOS extended memory API).