

## Re: memory reading and writing

**Source:** <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2004-07/1059.html>

---

**From:** f0dder (*f0dder\_spicedham\_at\_flork.dk*)

**Date:** 07/25/04

Date: Sun, 25 Jul 2004 23:46:04 +0200

Phew! As always, lots of cutting coming up :p

> *Admittedly, I have not actually tried "raw" floppy code myself*  
> *but I'm sure I heard someone on the OS development group (when I*  
> *could still see that, can't anymore) talk about many of the*  
> *floppy problems being more down to how Microsoft have coded it*  
> *than necessarily inherent...*  
>

Might be interesting playing around with this... while Microsoft certainly aren't doing things optimally all the time (trying to be very diplomatic ;)), I wouldn't have thought they would have done the floppy driver `_that_bad` if a much better implementation was possible. Humm, (testing under XP) it doesn't seem as bad as I remember it – I can run `tripex3` winamp visualization in the background, type in outlook, and copy files from a floppy. Hm, thinking about it, I haven't really been using floppies much since the Win9x days – and 9x sucked at just about everything ;)

>> *As for spinning up the CD drive, it's only explorer that freezes.*  
>  
> *Well, more like "only the application that's trying to do it*  
> *freezes"...*  
>

Yeah, of course – the only place where it has really annoyed me has been explorer, though. Usually when I open a file in some application I can't really do much more in that application before the file is open – but when `_explorer_` lags for a few seconds (CD or HDD spinup, slow network access time, ...) it somehow feels as if the operating system freezes, because explorer is a "part of" windows... this might not make much sense from a programmers viewpoint, but I'm still able to think like a user every now and then :). So well, an application can't do much until it has the data it needs, but it should still try to keep its UI painted (even if not "responsive"). In many apps the "responsiveness" doesn't matter too much, but in any kind of OS-integrated "pathfinder" it certainly does.

## alt.lang.asm: Re: memory reading and writing

- > *The taskbar can also sometimes refuse to react or change its*
- > *buttons sometimes...but, then, the taskbar is Explorer "in*
- > *disguise"*

>

Yeap. Seems to help a bit if you tick the option to have explorer windows as multiple processes or whatever. I think microsoft should have split the shell into "shell.exe" or "ui.exe"... while there might be a bunch of code reuse, this is easily handled by DLL files, etc. Oh, and there should have been better use of exception handling so that even when explorer windows are threads rather than processes, a single window crash should still not bring the entire process down (notice how this mostly seems to happen with Active Desktop – but then happens a lot).

- > *only the permanent OS-based things like the sound volume controls*
- > *seem to understand how to move and connect to your new taskbar*
- > *(not documented that Windows does this or just that third-parties*
- > *never account for the taskbar crashing as part of their design?)*

>

I'm positive I've seen a few third-party apps handle this gracefully... I wonder if it's a standard documented thing, or some hack... and would certainly be nice if more third-party apps handled it. Would be wonderful if windows was perfect, wouldn't it? :-)

- > *– so you'll probably want to reboot, anyway...*

>

Nah, just attach a debugger to the process and temporarily redirect execution to the code part that shows the tray icon ;-)

- > *Anyway, if I was writing a multi-tasking OS, then I'd be tempted*
- > *to simply NOT include any synchronous routines at all (though*
- > *careful design should be applied not to make this too*
- > *complicated a procedure to use :)...*

>

And there you hit the nail right on the head. Programmers don't want to do even slightly more complex code if their simple code works "pretty well" or "well enough". The "spin-up" issue can't be fixed at an OS level (well, perhaps if we do a very comprehensive paradigm shift), but has to be fixed at an app-level.

One of the big problems with windows is the WIN32 API. It is very clear that it suffers a lot from having very strong ties to the win16 days. Perhaps it's also "bad" that many of the later APIs actually offer a lot of flexibility – enough that many people stick to the old outdated ones. Compare POSIX and the WIN32 API and tell me which one will let you write the most powerful and flexible stuff :)

Would've been nice if MS had scrapped all win16 ties (and done support through an emulation layer), and never done kernel hacks to support thirdparty (even non-MS) applications etc etc etc.

Unfortunately (both for MS and the end-users), their monopoly status required them to keep support for a shitload of legacy \*crap\*, and this wasn't exactly achieved in the best possible way.

> and any "synchronous" behaviour is performed as a two-step  
> action: Call the asynchronous routine, block on the return value  
> (a routine to block on some "return code" would be made available  
> for this :)  
>

As far as I have understood, this is pretty much how NT works internally, below the win32 API layer... might have been a smart move of microsoft if they had exposed the NT native API, or at least written a better shim ontop of it than the win32 patchwork.

> Imagine, as you're typing your password, a  
> naughty program appears – is automatically given the keyboard  
> focus by "what does multi-tasking mean?" Windows without express  
> permission to change windows – and gets your password, sending  
> it over the 'Net, before you've even realised what's  
> happened...  
>

I've had that kind of thing happening before – pretty nasty.  
But I guess it might be a bit of trouble writing good heuristics for what to do with a new window – and it seems like "don't allow background application to steal focus" setting doesn't always work... haven't figured out exactly what causes it to malfunction and let a background app go foreground and steal focus :/

> Or, in fact, there was a question here about which would be the  
> fastest look-up routine for a command-line prompt to look-up its  
> internal commands...  
>

\*snipsnip\* – that exact example seems a bit silly, unless you're working on some *\_very\_* limited platform... but the mentality and ideas from such a discussion can (and often should) be applied to other problems, though. (Lots of people tend to ignore the background idea if the example chosen to discuss them seems "silly" to them – which is unfortunate.)

> then it went "full-screen" and showed you an "intro movie"  
> explaining the plot and that kind of thing *\_WHILE\_* the program  
> was installing in the background...  
>

I'm a bit mixed wrt. this kind of thing. It's very nice during first-time install, but can become very annoying when you need to re-install the game (new computer, re-installed OS, etc.) The really annoying thing is when the installer inserts some wait-states to make sure the entire intro video has been played. This becomes annoying when you, a few years later, install the game on you super-multi-terrahertz computer and 500MB/S supah

SCSI-ultra-wide-15-with-wings drive, and it still takes 15 minutes to install ;). It should really be an option at install time... and something I'd definitely let the game do the first time I install it. At later installs, I'd prefer leaving the installer running in the background, surfing the web or whatever while it finishes.

> *so, the game shows you this movie while it's installing, when you'd  
> be doing nothing else but sitting watching a progress bar instead,  
> anyway...*  
>

Yup. Nice the first time around, indeed. I think the original C&C did this too – or at least the installer UI was interesting ;)

> *This same basic idea could be used elsewhere, though, in  
> modified form...you can imagine, say, an application that offers  
> a menu and the user can choose to read "what's new?" or "take  
> the animated tour" or follow one of the "how to use ..."  
> tutorials...all of which happens \_WHILE\_ the application  
> installs in the background...*  
>

Yeah – something that does NOT slow the install time down (well, not very much anyway), and lets you do something that you would usually (if at all) *after* the install process is done. And if you lower the thread that does the install work a bit, it shouldn't affect the performance of the tour or whatever.

The windows OS install procedure really ought to have a little tetris game or something :) (oh, and thank heavens for unattended setup scripts... anybody remember the win9x install where you had to sit in front of the machine until the install was done, because you'd have to answer silly questions multiple times during the install process?)

> *Indeed, you're an "old hand", as they say, who had a C64, right?*  
>

Well, back in the C=64 (and afterwards, amiga) days I was merely a little kiddo gamer... but I'm old enough to have experienced these wonderful machines, yes. I'm but a snotty-nosed 22-year-old, though, and didn't start programming until I got my hands on a PC.

> *and practically every game had its "loading picture", which would  
> slowly draw onto the screen while a tune played, as the game loaded*  
>

I remember the fastloaders from cracked games best, \*cough\*. Back in the C=64 days I didn't even know that you could *buy* games, you'd just get them from the other kids down the road, and there weren't really any places around in my town that sold games. In the amiga days I realized that you could actually *buy* software and had some vague idea that pirace was probably illegal, but hey...

- > *The Microsoft branded games actually aren't so bad at*
- > *all...although, of course, one needs to quote "Microsoft" here*
- > *because, in fact, \_other software companies\_ write them and it's*
- > *just released on the "Microsoft games" label ("contract" or*
- > *"freelance" work or something like that ;)...*

>  
Yup, that's why I wrote "microsoft brand" instead of "the microsoft game <xxx>" :). This thing can be both good and bad – really crappy when microsoft bought bungee (or whatever) and caused HALO not to be released on PC until years too late, but good if there's some smallish company that could need some financial backing...  
But well, I think microsoft should focus more attention on windows, and fire some of the sloppy programmers they got.

- > *but just what Microsoft's money can buy when handed to other people*
- > *who \_do\_, at least, know how to program a little...*

>  
Some of the microsoft programmers are actually very good – they just need to fire all those crap programmers, tidy up their ring3 business, plan a bit more ahead, and respect existing standards ^\_^

- >> *There's one console game (can't remember which), that has a very*
- >> *cute form of background loading... if you run too fast towards a*
- >> *new area, and the game sees it can't load fast enough, your*
- >> *character will stumble and trip, giving the engine that half or*
- >> *whole second that lets it load :)*

- >
- > *Well, full marks for the "novel" way of getting around the*
- > *technical problem...but, well, the player shouldn't really*
- > *suffer for what's the programmers' technical problem...I'd*
- > *wonder if the "map" could simply be designed in such a way as*
- > *it's just simply impossible to run to another section before it*
- > *could be loaded...you know, split up the "map" into smaller*
- > *chunks so that it can be loaded quicker...then place the "doors"*
- > *between rooms in certain places so that they can't \_physically\_*
- > *get into the other "room" before it's had a chance to load...*

>  
Consider things like a scratched CD, slightly dusty laser head, etc... or that the game designers got the timing just a \_bit\_ wrong. I don't think you'd experience this stumbling very often, just in those few situations where the game realized it couldn't load fast enough. And lightyears better than a load screen anyway :) (and it was one of those platform games with relatively large outdoor areas anyway, so "rooms" doesn't make much sense.)

Doors (or, in general, "portals") can be pretty useful in indoor games... but I guess it all comes down to the data structures you use to represent the level and visibility information.

- > *It would require careful planning and that kind of thing but I'm*
- > *sure, especially with today's PCs, it could be made to work, if*

> *someone really wanted it to work...*

>

The data structures used, the kind of AI you want, etc... I haven't really done any 3d programming except from looking at NeHe's GL tutorials :,) but I've got the impression that things like BSP (and the visibility data) used in the quake engine games is pretty static, and would/might be hard to split up like this. But that's just my "sorta technical end-user" view of things :). I've also got the impression that things are becoming a lot more dynamic with the next-generation engines, so it might be easier to split up levels?

>> *\*shrug\* – I remember the days of incompatible 3D standards (glide,*

>> *PowerVR, etc) and I certainly don't want to go back to this.*

>

> *Ah, sorry...I get annoyed at this kind of argument at*

> *times...because it can often lead to \_excuses\_ rather than*

> *\_reasons\_...*

>

(rest of stuff snipped). Well, we need some form of standard, and if the choices are GL or DX, and in the context of "primarily games, but also applications", I know which one I would choose. DX might not be the best possible API, but it certainly seems to be the better of the two. Features supported, ability to query hardware capabilities (more than just a list of vendor extensions), etc.

> *I mean, using "sockets" for IPC between threads \_on the same*

> *machine\_? Think of all that "overhead" when it strictly isn't*

> *needed...*

>

Yep, good thing *\*windows\** has a lot more flexible mechanisms than that... good thing we aren't stuck with posix...

> *No, I'm not saying "don't bother with portability"...but just*

> *pointing out that what was a "portability problem" one day \_CAN*

> *CHANGE\_...*

>

Well, bother with portability if you need it, or if there's some indication it can pay off. Or, rather, avoid portability if it means you have to stick with lowest common denominator code.

Also, the whole GLIDE thing shows that standards are, like, good... GLIDE probably wasn't a bad API at that time, but tying yourself to an API used by one single vendor (that doesn't have a monopoly :P ) is a bad thing. Also, APIs do change... first versions of DirectX sucked bigtime, and GLIDE (or S3's ViRGE stuff) wouldn't look anything similar today, since the hardware and programming paradigm has changed a lot.

Probably wouldn't even make much sense to expose the direct (GPU) hardware anymore – it would mean lots of incompatibilities,

most of "what needs to be done" happens at a relatively high level, and then there's the vertex and pixel shader programming being handled at an "abstracted assembly level"...

And if that isn't good enough, upgrade DirectX... the new features seem to get added a lot faster to both the GPUs, as well as DirectX, than games start utilizing them, anyway...

>> *You might argue that you could make a standard for how GPUs should work, but I'm afraid this would hinder innovation.*

>

> *Completely depends on the standard; For instance, if it's made highly "free form" then it could \_encourage\_ innovation instead...*

>

Yeah, I guess so... still think it's best to let software drivers handle abstraction rather than the hardware, to keep the cost of hardware down. But we do need standards, and we must make sure they don't get too loose, or it will be hard to actually make working programs. It's a very fine balance... I think the way DirectX has evolved isn't too bad... each version determines a set of features that must be supported, and you can keep a full interface around for each version. Doing this in hardware would probably bump up the transistor count a lot, but can be done in software relatively inexpensively (once an interface has become obsolete and needs a *\*lot\** of emulation, hardware should have improved a lot too :-))

But one shouldn't get carried away – we don't want to see "winmodem" behaviour in hardware graphics accelerators... Don't do things in software just because they can be done, one needs to look at both cost and performance.

> *I mean, it works both ways...if you just consider that you're saying "everyone should fall in line to a single software-based standard in DirectX" and then talk about "hindering innovation" > (!!)...*

>

Well, better to fall in line to DirectX, than a single hardware producers' API. At least there's some powerful hardware vendors that can push Microsoft developers in the right direction. Probably could be better, but at least DX has sorta quickly adapt vs/ps 1.x and 2.x and they're working on 3.x now. GL hasn't even standardized 2.x, and when that happens, it seems like 1.x won't be supported? Of course it could be better, but consider what would have happened if we had no standard, or if GL was the only standard (GL-only would be at least a bit better – it is a lot easier for a competing vendor to add some extension support, than making their card I/O compatible with a competitor.)

- > *Yeah, that's right...Microsoft could never be accused of being*
- > *behind some of the greatest hinderances in computer innovation*
- > *there's ever been, right?*

>

It's probably done a lot more good than bad in the case of graphics – DirectX has incorporated new technologies quickly (well, after the \*MISERABLE\* early years), and windows has been, on the PC scene anyway, a big pusher of hardware. MS has done lots and lots and lots of damage elsewhere though. Corrupting existing standards, buying other companies and discontinuing their products, and supporting the annoying AMD 64bit extensions for the x86 architecture, keeping this patchwork alive even longer :-)

Please do keep in mind I don't see MS as this big and good company that's only working for our greater well. They're greedy and ruthless, and have made a lot of mistakes. I'm trying to be a bit realistic and look at what other companies have to offer, though. And I have this idea that most other companies would have been just as ev0l if they had the chance... and the opensource movement seems to just be plagiarizing commercial ideas, and often not even doing it very well :/

- >> *DirectX doesn't really seem that bad, while the first versions*
- >> *sucked majorly, the more recent ones perform pretty well.*
- >
- > *It's not a question of "good" and "bad", in my opinion...in this*
- > *respect, there is only \_ONE\_ question to ask ourselves: "Can we*
- > *do \_BETTER\_?"...*

>

Well then, MS would have to listen to leading game and hardware developers (hm, don't they already do this?), and be careful not to listen too much to individuals (dunno if this is real or not, but I heard that part of the blame for the first horrible versions of D3D were that Mr. Carmack wanted oh-so-direct hardware access... then chickened out and went with OpenGL instead, \*shrug\*).

- > *I saw an interview with Pele (often considered the best soccer*
- > *player there's ever been...everyone will know him except maybe*
- > *Americans, as Americans don't follow soccer like the rest of the*
- > *world does)*

>

Soccer? What's soccer? ;-)

- >> *Humm, compare DX to GL and then talk about lowest common*
- >> *denominator :)*
- >
- > *Stands to reason and proves my point; DirectX only needs to*
- > *"standardise" across \_PC hardware using Windows...OpenGL is a*
- > *"standard" to cover \_all\_ platforms...*

>

Plus GL wasn't invented for games and multimedia... thing is, DX was invented for games and multimedia, and seems to do rather well in business applications as well. And there isn't really anything that stops it from being implemented on non-x86 machines. There's linux stuff working on top of nvidia GL, there's a MAC wrapper... and if it wasn't for opensource zeal, it could be implemented elsewhere, too (well, might be some MS patents hindering this, I dunno :-))

> Hence, the "lowest common denominator" for GL is going to  
> naturally be *\_lower\_* than the "lowest common denominator" of  
> DX...to expect anything else is to miscomprehend the very idea  
> of what we're talking about...

>  
There used to be more powerful hardware around for GL than for DX... I wonder how it is now, with the latest nvidia and ATi accelerators. Even SGI use ATi chips in their onyx4, and HP use P4 XEONs and nvidia quattro chips in their sv7 visualization cluster.

> Oh, probably; I said *\_properly\_* take advantage of...though, I've  
> not seen Thief 3 so perhaps that has made some good use of it  
> already...

>  
Well, it's the best use of bumpmapping I've seen yet – since you don't really think about it right away, the game just looks good. It's more impressive in doom3 I guess, though. Btw, as for thief, I really really really like the game and it's pretty immersive. The following strips point out some fun things about it though :p  
<http://www.ctrlaltdel-online.com/?t=archives&date=2004-06-16>  
<http://www.ctrlaltdel-online.com/?t=archives&date=2004-06-21>

> Actually, Counterstrike just a "mod" for Half-life and it's the  
> exact same engine running it, just different game files...and,  
> no, that's the surprising thing: Half-life is a modified *\_Quake*  
> *\_engine\_*...yes, the *\_first\_* one...

>  
Yup, know that CS is just a mod for hl (well, condition zero is somewhat more, but...) – I did think HL was Q2 rather than Q1. Looking back etc you're probably right, though... but I think the source that valve worked from was closer to Q2 than Q1. I mean, look at it – DLL files instead of progs.dat, hardware acceleration etc... they probably started from a codebase that was q1.5 or q1.66 rather than q1 or q2. I might be wrong, but it just seems a bit weird if valve would have implemented so many Q2 things in such a similar way :)

> now perhaps my observations about these cards being "Quake  
> accelerators" starts to make more sense: Two completely  
> different companies added in "hardware acceleration" and then  
> they both end up looking remarkably similar?

>

Not that fair though, there were a bunch of different game made even back in the very limited fixed-function days... and while the features were limited, the demoscene did show that you could do a hell lot more than quake-style graphics with that kind of acceleration!

The early DX games looked washed-out though, and the GL games were often dark. In my terrible memory, the GLIDE games looked a lot better than the early DX/GL games... but things have changed.

>> *But yes, that is still a very old engine.*

>

> *Yes; But, really, no mistake: It's the Quake I engine that was  
> the base of Half-life...basically, Valve's "modifications" were  
> more or less just a case of updating the engine to be able to  
> handle 3D acceleration cards and, you know, modifications to  
> little things here and there*

>

I think a lot of those things could have been done even with an unmodified quake1 engine with interpreted progs.dat style QuakeC. But halflife does show that, even without major changes, you can get very different gameplay with the same engine. I don't think the stuff done in halflife was "cheating" btw, it was getting things done in a realistic way. Waypoints don't require too much of the level designer, and make things a lot easier for the programmers.

Interesting note: valve claims that most things in HL2 will be driven by AI rather than scripts... ie, sequences that were scripted in HL1 would be scripted in HL2. This sounds pretty interesting, especially when you're going to complete the game a 2nd time (I think I went through HL about 3 times :-))

> *give-away...they are called "QBSP", "QVIS", "QRAD", etc....and,  
\*snip\**

> *So, really, it's the Quake \_one\_ engine, not Quake II...hence,  
> yes, it \_IS\_ an old engine...older than you think...*

>

Iirc RAD wasn't introduced until quake2, and quake1 used a much simpler lighting model, hinting that HL was at least somewhere between Q1 and Q2. But again, it has been quite some years, and I could very easily be wrong.

> *(you know, I'm sitting there watching TV and then jump up  
> shouting: "Wow! Those are the horns from Half-Life!"...*

>

Bwaha :D

Most of the times this happens to me, the sound in the movie or tv show is from either doom, XCOM or Magic Carpet... either those games have been ripped off bigtimes, or they used the same public

domain sound libraries as some movies :p

> *Actually, I must get around to that little bit of DOS code I was  
> going to post up here that demonstrated that the hardware is  
> actually capable of \_far more\_ than people often realise...the  
> actual problem being that everyone sees the BIOS interface and  
> then presumes that's about all you could really do with the  
> hardware...*

>

*\*snip\**

I played around with some of those tricks many years ago (no, most of the code wasn't written by me.) – graphical mouse cursor, copper bars, "graphic region in text mode", etc. It's all very cute, but some of it required so strict timing that it wouldn't be feasible to do in a multitasking OS. And that IS a thing to keep in mind – when not working on a very fixed hardware platform, you shouldn't make assumptions on the system, at least not outside demo apps :-/

> *hmmm, how do those C64 guys do the "bump mapping"?*

>

No idea, but probably lots of faking. Looked good, though.

> *Ah, okay...clarification: "input" covers the whole*

> *Mind you, if you're \_really\_ pedantic – and I've not actually  
> seen anyone take advantage of something like this – then one  
> should also account for the fact that the speed of light is much  
> faster than the speed of sound...things further distant \_should\_  
> have a very small amount of "latency" to actually be  
> realistic...*

>

I wonder what kind of range you would need before taking this into account, though... more than a couple hundred meters? You'd only see it in very few games today, then :) (we already have left/right (and even more funky 3D positional audio) separation and doppler effect.)

> *Yeah, but it's iD...and it's DOOM...an estimated \_15 million\_  
> installations of the shareware DOOM worldwide...and the engine  
> does have the edge, which is all you can be totally sure about  
> from screenshots...blah–blah–blah...*

>

I've seen the alpha in realtime... looked good, and totally spooge effect at that time. But unless they've add a *\*lot\** to it since then, it no longer impresses me much. Hope they've at least done some gameplay, otherwise it will lose majorly to HL2 (well, zealot fans aside, anyway.)

*\*phew\* :-)*