John Carmack Archive - .plan (1998)

http://www.team5150.com/~andrew/carmack

March 18, 2007

Contents

1 January **5**

1.1 Some of the things I have changed recently (Jan 01, 1998) . 5

1.2 Jan 02, 1998 6

1.3 New stuff fixed (Jan 03, 1998) 7

1.4 Version 3.10 patch is now out. (Jan 04, 1998) 8

1.5 Jan 09, 1998 9

1.6 I AM GOING OUT OF TOWN NEXT WEEK, DON'T SEND ME ANY MAIL! (Jan 11, 1998) 10

2 February **12**

2.1 Ok, I'm overdue for an update. (Feb 04, 1998) 12

2.2 Just got back from the Q2 wrap party in vegas that Activision threw for us. (Feb 09, 1998) 14

2.3 Feb 12, 1998 15

2.4 8 mb or 12 mb voodoo 2? (Feb 16, 1998) 19

2.5 I just read the Wired article about all the Doom spawn. (Feb 17, 1998) 20

2.6 Feb 22, 1998 21

3 March **22**

3.1 American McGee has been let go from Id. (Mar 12, 1998) . 22

3.2 The Old Plan (Mar 13, 1998) 22

3.3 Mar 20, 1998 25

3.4 I just shut down the last of the NEXTSTEP systems running at id. (Mar 21, 1998) 26

3.5 Mar 26, 1998 28

4 April **30**

4.1 Drag strip day! (Apr 02, 1998) 30

4.2 Things are progressing reasonably well on the Quake 3 engine. (Apr 08, 1998) 31

4.3 Apr 16, 1998 32

4.4 Yes, I bought an F50. No, I don't want a McLaren. (Apr 17, 1998) 32

4.5 F50 pros and cons vs F40 (Apr 22, 1998) 34

5 May **36**

5.1 May 02, 1998 36

5.2 May 04, 1998 36

5.3 Here is an example of some bad programming in quake (May 17, 1998) 39

5.4 May 19, 1998 39

5.5 May 22, 1998 41

6 June **42**

6.1 Jun 08, 1998 42

6.2 Jun 16, 1998 44

7 July 46

7.1 Here is the real story on the movement physics changes. (Jul 04, 1998) 46

7.2 Jul 05, 1998 47

7.3 I have spent the last two days working with Apple's Rhapsody DR2, and I like it a lot. (Jul 16, 1998) 48

7.4 My F50 took some twin turbo vitamins. (Jul 29, 1998) . . . 50

8 August 51

8.1 Aug 17, 1998 51

9 September 53

9.1 I just got a production TNT board installed in my Dolch today. (Sep 08, 1998) 53

9.2 Sep 10, 1998 55

10 October 58

10.1 Oct 14, 1998 58

11 November 61

11.1 Nov 03, 1998 61

11.2 Nov 04, 1998 64

12 December 65

12.1 Dec 29, 1998 65

12.2 Dec 30, 1998 66

Chapter 1

January

1.1 Some of the things I have changed recently (Jan 01, 1998)

* fixed the cinematics
* don’t clear config after dedicated server
* don’t reallocate sockets unless needed
* don’t process channel packets while connecting
* rate variable for modem bandwidth choking
* delta compress client usercmds
* fixed sound quality changing after intermissions
* fixed PVS problem when head was directly under solid in GL
* added r_drawflat and cl_testlights to cheats

There are a few problems that I am still trying to track down:

WSAEADDRNOTAVAIL errors
Map versions differ error
Sometimes connecting and seeing messages but not getting in
Decompression read overrun.

Of course, we don’t actually get any of those errors on any of our systems here, so I am having to work remotely with other users to try and fix them, which is a bit tougher.

My new years resolution is to improve my coding style by bracing all single line statements and consistantly using following caps on multi word variable names.

Actually, I am currently trying on the full sun coding style, but I'm not so sure about some of the comment conventions: don't use multiple lines of // comments, and don't use rows of seperating characters in comments. I'm not convinced those are good guidelines.

1.2 Jan 02, 1998

Wired magazine does something that almost no other print magazine we have dealt with does.

They check the statements they are going to print.

I just got a "fact check" questionaire email from wired about an upcoming article, and I recall that they did this last time the did an article about us.

Most of the time when we talk with the press, we try to get them to send us a proof of the article for fact checking. They usually roll their eyes, and grudgingly agree, then don't send us anything, or send it to us after it has gone to press.

Wired had a few errors in their statements, but it won't get printed that way because they checked with us.

How refreshing.

–

A small public announcement:

The Linux Expo is looking for:

1. People that develop games or game servers in *nix, and 2. People interested in learning how to develop games in *nix.

Either one should give a write to ddt@crack.com.

1.3 New stuff fixed (Jan 03, 1998)

* timeout based non-active packet streams
* FS_Read with CD off checks
* dedicated server not allocate client ports
* qport proxy checking stuff
* fixed mouse wheel control
* forced newlines on several Cbuf_AddText ()
* if no nextmap on a level, just stay on same one
* chat maximums to prevent user forced overflows
* limit stringcmds per frame to prevent malicious use
* helped jumping down slopes
* checksum client move messages to prevent proxy bots
* challenge / response connection process
* fixed rcon
* made muzzle flash lights single frame, rather than 0.1 sec

I still don't have an answer to the WAADRNOTAVAILABLE problem. I have made the packet stream as friendly as possible, but some computers are still choking.

I managed to get fixes for address translating routers done without costing any bandwidth from the server, just a couple bytes from the client, which isn't usually a critical path.

I have spent a fair amount of time trying to protect against "bad" users in this release. I'm sure there will be more things that come up, but I know I got a few of the ones that are currently being exploited.

We will address any attack that can make a server crash. Other attacks will have to have the damage and prevelence weighed against the cost of defending against it.

Client message overflows. The maximum number of commands that can be issued in a user packet has been limited. This prevents a client from doing enough "says" or "kills" to overflow the message buffers of other clients.

Challenge on connection. A connection request to a server is now a two stage process of requesting a challenge, then using it to connect. This prevents denial of service attacks where connection packets with forged IPs are flooded at a server, preventing any other users from connecting until they timeout.

Client packet checksumming. The packets are encoded in a way that will prevent proxies that muck with the packet contents, like the stoogebot, from working.

1.4 Version 3.10 patch is now out. (Jan 04, 1998)

`ftp://ftp.idsoftware.com/idstuff/quake2/q2-310.exe`

A few more minor fixes since yesterday:

* qhost support
* made qport more random
* fixed map reconnecting
* removed s_sounddir
* print out primary / secondary sound buffer status on init
* abort game after a single net error if not dedicated
* fixed sound loss when changing sound compatability
* removed redundant reliable overflow print on servers
* gl_lockpvs for map development checking
* made s_primary 0 the default

Christian will be updating the bug page tomorrow. So hold of on all reporting for 24 hours, then check the page to make sure the bug is not already known.

`http://www.idsoftware.com/cgi-win/bugs.exe`

All bug reports should go to Christian: xian@idsoftware.com.

I have had several cases of people with lockup problems and decompression overreads having their problems fixed after they mentioned that they were overclocking either their CPU, their system bus (to 75mhz), or their 3DFX.

It doesn't matter if ”it works for everything else”, it still may be the source of the problem.

I know that some people are still having problems with vanilla systems, though. I have tried everything I can think of remotely, but if someone from the Dallas area wants to bring a system by our office, I can try some more serious investigations.

Something that has shown to help with some 3dfx problems is to set ”cl_maxfps 31”, which will keep the console between level changes from rendering too fast, which has caused some cards to hang the system.

1.5 Jan 09, 1998

We got 70 people on a base100 server, and it died after it wedged at 100% utilization for a while. Tomorrow we will find exactly what overflowed, and do some profiling.

Base100 is really only good for 50 or so players without overcrowding, but we have another map being built that should hold 100 people reasonably well.

I will look into which will be the easier path to more server performance: scalar optimization of whatever is critical now, or splitting it off into some more threads to run on multiple processors. Neither one is trivial.

My goal is to be able to host stable 100 player games in a single map.

I just added a ”players” command that will dump the total number of players in the game, and as many frags/names as it can fit in a packet (around 50, I think).

1.6 I AM GOING OUT OF TOWN NEXT WEEK, DON'T SEND ME ANY MAIL! (Jan 11, 1998)

Odds are that I will get back and just flush the 500 messages in my mailbox.

No, I'm not taking a vacation. Quite the opposite, in fact.

I'm getting a hotel room in a state where I don't know anyone, so I can do a bunch of research with no distractions.

I bought a new computer specifically for this purpose - A Dolch portable pentium-II system. The significant thing is that it has full length PCI slots, so I was able to put an Evans & Sutherland OpenGL accelerator in it (not enough room for an intergraph Realizm, though), and still drive the internal LCD screen. It works out pretty well, but I'm sure there will be conventional laptops with good 3D acceleration available later this year.

This will be an interesting experiment for me. I have always wondered how much of my time that isn't at peak productivity is a necessary rest break, and how much of it is just wasted.

—

The client's IP address is now added to the userinfo before calling ClientConnect(), so any IP filtering / banning rules can now be implemented in the game dll. This will also give some of you crazy types the ability to sync up with multiple programs on the client computers outside of Q2 itself.

A new API entry point has been added to the game dll that gets called whenever an "sv" command is issued on the server console. This is to allow you to create commands for the server operator to type, as opposed to commands that a client would type (which are defined in g_cmds.c).

—

We did a bunch of profiling today, and finaly got the information I wanted. We weren't doing anything brain dead stupid in the server, and all of the time was pretty much where I expected it to be.

I did found two things we can pursue for optimization.

A moderately expensive catagorization function is called at both the beginning and end of client movement simulation. With some care, we should be able to avoid the first one most of the time. That alone should be good for a $>$ 10% server speedup.

The other major thing is that the client movement simulation accounted for 60% of the total execution time, and because it was already compartmentalized for client side prediction, it would not be much work to make it thread safe. Unfortunately, it would require MAJOR rework of the server code (and some of the game dll) to allow multiple client commands to run in parallel.

The potential is there to double the peak load that a server can carry if you have multiple processors. Note that you will definately get more players / system by just running multiple independent servers, rather than trying to get them all into a single large server.

We are not going to pursue either of these optimizations right now, but they will both be looked at again later.

All this optimizing of the single server is pushing the tail end of a paradigm. I expect trinity to be able to seamlessly hand off between clustered servers without the client even knowing it happened.

1.6. I AM GOING OUT OF TOWN NEXT WEEK, DON'T SEND ME ANY MAIL! (JAN 11, 1998)

Chapter 2

February

2.1 Ok, I'm overdue for an update. (Feb 04, 1998)

The research getaway went well. In the space of a week, I only left my hotel to buy diet coke. It seems to have spoiled me a bit, the little distractions in the office grate on me a bit more since. I will likely make week long research excursions a fairly regular thing during non- crunch time. Once a quarter sounds about right.

I'm not ready to talk specifically about what I am working on for trinity. Quake went through many false starts (beam trees, portals, etc) before settling down on its final architecture, so I know that the odds are good that what I am doing now won't actually be used in the final product, and I don't want to mention anything that could be taken as an implied "promise" by some people.

I'm very excited by all the prospects, though.

Many game developers are in it only for the final product, and the process is just what they have to go through to get there. I respect that, but my motivation is a bit different.

For me, while I do take a lot of pride in shipping a great product, the achievements along the way are more memorable. I don't remember any of our older product releases, but I remember the important insights all the way back to using CRTC wraparound for infinate smooth scrolling in Keen (actually, all the way back to understanding the virtues of structures over parallel arrays in apple II assembly language..). Knowledge builds on knowledge.

I wind up catagorizing periods of my life by how rich my learning experiences were at the time.

My basic skills built up during school on apple II computers, but lack of resources limited how far and fast I could go. The situation is so much better for programmers today - a cheap used PC, a linux CD, and an internet account, and you have all the tools and resources necessary to work your way to any level of programming skill you want to shoot for.

My first six months at Softdisk, working on the PC, was an incredible learning experience. For the first time, I was around a couple of programmers with more experience than I had (Romero and Lane Roath), there were a lot of books and materials available, and I could devote my full and undivided attention to programming. I had a great time.

The two years following, culminating in DOOM and the various video game console work I did, was a steady increase in skills and knowledge along several fronts - more graphics, networking, unix, compiler writing, cross development, risc architectures, etc.

The first year of Quake's development was awesome. I got to try so many new things, and I had Michael Abrash as my sounding board. It would probably surprise many classically trained graphics programmers how little I new about conventional 3D when I wrote DOOM - hell, I had problems properly clipping wall polygons (which is where all the polar coordinate nonsense came from). Quake forced me to learn things right, as well as find some new innovations.

The last six months of Quake's development was mostly pain and suffering trying to get the damn thing finished. It was all worth it in the end, but I don't look back at it all that fondly.

The development cycle of Quake 2 had some moderate learning experiences for me (glquake, quakeworld, radiosity, openGL tool programming, win32, etc), but it also gave my mind time to sift through a lot of things before getting ready to really push ahead.

I think that the upcoming development cycle for trinity is going to be at least as rewarding as Quake's was. I am reaching deep levels of understanding on some topics, and I am branching out into several completely new (non-graphics) areas for me, that should cross-polinate well with everything else I am doing.

There should also be a killer game at the end of it. :)

2.2 Just got back from the Q2 wrap party in vegas that Activision threw for us. (Feb 09, 1998)

Having a reasonable grounding in statistics and probability and no belief in luck, fate, karma, or god(s), the only casino game that interests me is blackjack.

Playing blackjack properly is a test of personal discipline. It takes a small amount of skill to know the right plays and count the cards, but the hard part is making yourself consistently behave like a robot, rather than succumbing to your "gut instincts".

I play a basic high/low count, but I scale my bets widely - up to 20 to 1 in some cases. Its not like I'm trying to make a living at it, so the chance of getting kicked out doesn't bother me too much.

I won $20,000 at the tables, which I am donating to the Free Software Foundation. I have been meaning to do something for the FSF for a long time. Quake was deployed on a dos port of FSF software, and both DOOM and Quake were developed on NEXTSTEP, which uses many FSF based tools. I don't subscribe to all the FSF dogma, but I have clearly benefited from their efforts.

2.3 Feb 12, 1998

I have been getting a lot of mail with questions about the intel i740 today, so here is a general update on the state of 3D cards as they relate to quake engine games.

ATI rage pro
—-
On paper, this chip looks like it should run almost decently - about the performance of a permedia II, but with per-pixel mip mapping and colored lighting. With the currently shipping MCD GL driver on NT, it just doesn't run well at all. The performance is well below acceptable, and there are some strange mip map selection errors. We have been hearing for quite some time that ATI is working on an OpenGL ICD for both '95 and NT, but we haven't seen it yet. The rage pro supposedly has multitexture capability, which would help out quite a bit if they implement the multitexture extension. If they do a very good driver, the rage pro may get up to the performance of the rendition cards. Supports up to 16MB, which would make it good for development work if the rest of it was up to par.

3DLabs permedia II

——

Good throughput, poor fillrate, fair quality, fair features.

No colored lighting blend mode, currently no mip mapping at all.

Supports up to 8MB.

The only currently shipping production full ICD for '95, but a little flaky.

If 3dlabs implemented per-polygon mip mapping, they would get both a quality and a slight fillrate boost.

Drivers available for WinNT on the DEC Alpha (but the alpha drivers are very flaky).

Power VR PCX2

—

Poor throughput, good fillrate, fair quality, poor features, low price.

No WinNT support.

Almost no blend modes at all, low alpha precision.

Even though the hardware doesn't support multitexture, they could implement the multi-texture extension just to save on polygon setup costs. That might get them a 10% to 15% performance boost.

They could implement the point parameters extension for a significant boost in the speed of particle rendering. That wouldn't affect benchmark scores very much, but it would help out in hectic deathmatches.

Their opengl minidriver is already a fairly heroic effort - the current PVR takes a lot of beating about the head to make it act like an OpenGL accelerator.

Rendition v2100 / v2200

———

Good throughput, good fillrate, very good quality, good features.

A good all around chip. Not quite voodoo1 performance, but close.

v2100 is simply better than everything else in the $99 price range.

Can render 24 bit color for the best possible quality, but their current drivers don't support it. Future ones probably will.

Can do 3D on the desktop.

Rendition should be shipping a full ICD OpenGL, which will make an 8mb v2200 a very good board for people doing 3D development work.

NVidia Riva 128

——

Very good throughput, very good fillrate, fair quality, fair features.

The fastest fill rate currently shipping, but it varies quite a bit based on texture size. On large textures it is slightly slower than voodoo, but on smaller textures it is over twice as fast.

On paper, their triangle throughput rate should be three times what voodoo gives, but in practice we are only seeing a slight advantage on very fast machines, and worse performance on pentium class machines. They probably have a lot of room to improve that in their drivers.

In general, it is fair to say that riva is somewhat faster than voodoo 1, but it has a few strikes against it.

The feature implementation is not complete. They have the blend mode for colored lighting, but they still don't have them all. That may hurt them in future games. Textures can only be 1 to 1 aspect ratio. In practice, that just means that non-square textures waste memory.

The rendering quality isn't quite as high as voodoo or rendition. It looks like some of their iterators don't have enough precision.

Nvidia is serious and committed to OpenGL. I am confident that their driver will continue to improve in both performance and robustness.

While they can do good 3D in a window, they are limited to a max of 4MB of framebuffer, which means that they can't run at a high enough resolution to do serious work.

3DFX Voodoo 1

——

The benchmark against which everything else is measured.

Good throughput, good fillrate, good quality, good features.

It has a couple faults, but damn few: max texture size limited to 256*256 and 8 to 1 aspect ratio. Slow texture swapping. No 24 bit rendering.

Because of the slow texture swapping, anyone buying a voodoo should get a six mb board (e.g. Canopus Pure3D). The extra ram prevents some sizable jerks when textures need to be swapped.

Highly tuned minidriver. They have a full ICD in alpha, but they are being slow about moving it into production. Because of the add-in board nature of the 3dfx, the ICD won't be useful for things like running level editors, but it would at least guarantee that any new features added to quake engine games won't require revving the minidriver to add new functionality.

3DFX Voodoo 2

——

Not shipping yet, but we were given permission to talk about the benchmarks on their preproduction boards.

Excellent throughput, excellent fillrate, good quality, excellent features.

The numbers were far and away the best ever recorded, and they are going to get significantly better. On quake 2, voodoo 2 is setup limited, not fill rate limited. Voodoo 2 can do triangle strip and fan setup in hardware, but their opengl can't take advantage of it until the next revision of glide. When that happens, the number of vertexes being sent to the card will drop by HALF. At 640*480, they will probably become fill rate bound again (unless you interleave two boards), but at 512*384, they will probably exceed 100 fps on a timedemo. In practice, that means that you will play the game at 60 fps with hardly ever a dropped frame.

The texture swapping rate is greatly improved, addressing the only significant problem with voodoo.

I expect that for games that heavily use multitexture (all quake engine games), voodoo 2 will remain the highest performer for all of '98. All you other chip companies, feel free to prove me wrong. :)

Lack of 24 bit rendering is the only visual negative.

As with any voodoo solution, you also give up the ability to run 3D applications on your desktop. For pure gamers, that isn't an issue, but for hobbyists that may be interested in using 3D tools it may have some weight.

Intel i740
—-
Good throughput, good fillrate, good quality, good features.

A very competent chip. I wish intel great success with the 740. I think that it firmly establishes the baseline that other companies (especially the ones that didn't even make this list) will be forced to come up to.

Voodoo rendering quality, better than voodoo1 performance, good 3D on a desktop integration, and all textures come from AGP memory so there is no texture swapping at all.

Lack of 24 bit rendering is the only negative of any kind I can think of.

Their current MCD OpenGL on NT runs quake 2 pretty well. I have seen their ICD driver on '95 running quake 2, and it seems to be progressing well. The chip has the potential to outperform voodoo 1 across the board, but 3DFX has more highly tuned drivers right now, giving it a performance edge. I expect intel will get the performance up before releasing the ICD.

It is worth mentioning that of all the drivers we have tested, intel's MCD was the only driver that did absolutely everything flawlessly. I hope that their ICD has a similar level of quality (it's a MUCH bigger job).

An 8mb i740 will be a very good setup for 3D development work.

2.4 8 mb or 12 mb voodoo 2? (Feb 16, 1998)

An 8mb v2 has 2 mb of texture memory on each TMU. That is not as general as the current 6mb v1 cards that have 4 mb of texture memory on a single TMU. To use the multitexture capability, textures are restricted to being on one or the other TMU (simplifying a bit here). There is some benefit over only having 2 mb of memory, but it isn't double. You will see more texture swapping in quake on an 8mb voodoo 2 than you would on a 6mb voodoo 1. However, the texture swapping is several times faster, so it isn't necessarily all that bad.

If you use the 8 bit palettized textures, there will probably not be any noticable speed improvement with a 12 mb voodoo 2 vs an 8 mb one. The situation that would most stress it would be an active deathmatch that had players using every skin. You might see a difference there.

A game that uses multitexture and 16 bit textures for everything will stress a 4/2/2 voodoo layout. Several of the Quake engine licensees are using full 16 bit textures, and should perform better on a 4/4/4 card.

The differences probably won't show as significant on timedemo numbers, but they will be felt as little one frame hitches here and there.

2.5 I just read the Wired article about all the Doom spawn. (Feb 17, 1998)

I was quoted as saying ”like I’m supposed to be scared of Monolith”, which is much more derogatory sounding than I would like.

I haven’t followed Monolith’s development, and I don’t know any of their technical credentials, so I am not in any position to evaluate them.

The topic of ”is microsoft going to crush you now that they are in the game biz”, made me a bit sarcastic.

I honestly wish the best to everyone pursuing new engine development.

2.6 Feb 22, 1998

Don’t send any bug reports on the 3.12 release to me, I just forward them over to jcash. He is going to be managing all future work on the Quake 2 codebase through the mission packs. I’m working on trinity.

3.12 answered the release question pretty decisively for me. We were in code freeze for over two weeks while the release was being professionally beta tested, and all it seemed to get us was a two week later release.

Future releases are going to be of the fast/multiple release type, but clearly labeled as a ”beta” release until it stabilizes. A dozen professional testers or fifty amature testers just can’t compare to the thousands of players who will download a beta on the first day.

I have spent a while thinking about the causes of the patches for Q2. Our original plan was to just have the contents of 3.12 as the first patch, but have it out a month earlier than we did.

The first several patches were forced due to security weaknesses. Lesson learned - we need to design more security conscious to try to protect against the assholes out there.

The cause for the upcoming 3.13 patch is the same thing that has caused us a fair amount of trouble through Q2's development - instability in the gamex86 code due to its decending from QC code in Q1. It turns out that there were lots of bugs in the original QC code, but because of its safe interpreted nature (specifically having a null entity reference the world) they never really bothered anyone. We basically just ported the QC code to regular C for Q2 (it shows in the code) and fixed crash bugs as they popped up. We should have taken the time to redesign more for C's strengths and weaknesses.

Chapter 3

March

3.1 American McGee has been let go from Id. (Mar 12, 1998)

His past contributions include work in three of the all time great games (DOOM 2, Quake, Quake 2), but we were not seeing what we wanted.

3.2 The Old Plan (Mar 13, 1998)

The rest of the team works on an aggressive Quake 2 expansion pack while Brian and I develop tools and code for the entirely new Trinity generation project to begin after the mission pack ships.

The New Plan:

Expand the mission pack into a complete game, and merge together a completely new graphics engine with the quake 2 game / client / server framework, giving us Quake 3.

”Trinity” is basically being broken up into two phases: graphics and everything else. Towards the end of Quake 1’s development I was thinking that we might have been better off splitting quake on those categories, but in reverse order. Doing client/server, the better modification framework, and qc, coupled with a spiced up DOOM engine (Duke style) for one game, then doing the full 3D renderer for the following game.

We have no reason to believe that the next generation development would somehow go faster than the previous, so there is a real chance that doing all of the Trinity technology at once might push game development time to a full two years for us, which might be a bit more than the pressure-cooker work atmosphere here could handle.

So, we are going to try an experiment.

The non-graphics things that I was planning for Trinity will be held off until the following project - much java integration with client downloadable code being one of the more significant aspects. I hope to get to some next generation sound work, but the graphics engine is the only thing I am committing to.

The graphics engine is going to be hardware accelerated ONLY. NO SOFTWARE RENDERER, and it won't work very well on a lot of current hardware. We understand fully that this is going to significantly cut into our potential customer base, but everyone was tired of working under the constraints of the software renderer. There are still going to be plenty of good quake derived games to play from other developers for people without appropriate hardware.

There are some specific things that the graphics technology is leveraging that may influence your choice of a 3D accelerator.

All source artwork is being created and delivered in 24 bit color. An accelerator that can perform all 3D work in 24 bit color will look substantially better than a 16 bit card. You will pay a speed cost for it, though.

Most of the textures are going to be higher resolution. Larger amounts of texture memory will make a bigger difference than it does on Quake 2.

Some key rendering effects require blending modes that some cards don't support.

The fill rate requirements will be about 50% more than Quake 2, on average. Cards that are fill rate limited will slow down unless you go to a lower resolution.

The triangle rate requirements will be at least double Quake 2, and scalable to much higher levels of detail on appropriate hardware.

Here are my current GUESSES about how existing cards will perform.

Voodoo 1 Performance will be a little slow, but it should look good and run acceptably. You will have to use somewhat condensed textures to avoid texture thrashing.

Voodoo 2 Should run great. Getting the 12 mb board is probably a good idea if you want to use the high resolution textures. The main rendering mode won't be able to take advantage of the dual TMU the same way quake 2 does, so the extra TMU will be used for slightly higher quality rendering modes instead of greater speed: trilinear / detail texturing, or some full color effects where others get a mono channel.

Permedia 2 Will be completely fill rate bound, so it will basically run 2/3 the speed that quake 2 does. Not very fast. It also doesn't have one of the needed blending modes, so it won't look very good, either. P2 does support 24 bit rendering, but it won't be fast enough to use it.

ATI Rage Pro It looks like the rage pro has all the required blending modes, but the jury is still out on the performance.

Intel I740 Should run good with all features, and because all of the textures come out of AGP memory, there will be no texture thrashing at all, even with the full resolution textures.

Rendition 2100/2200 The 2100 should run about the speed of a voodoo 1, and the 2200 should be faster. They support all the necessary features, and an 8 mb 2200 should be able to use the high res textures without a problem. The renditions are the only current boards that can do 24 bit rendering with all the features. It will be a bit slow in 24 bit mode, but it will look the best.

PVR PCX2 Probably won't run Quake 3. They don't have ANY of the necessary blending modes, so it can't look correct. Video Logic might decide to rev their minidriver to try to support it, but it is probably futile.

RIVA 128 Riva puts us in a bad position. They are very fast, but they don't support an important feature. We can crutch it up by performing some extra drawing passes, but there is a bit of a quality loss, and it will impact their speed. They will probably be a bit faster than voodoo 1, but not to the degree that they are in Quake 2.

Naturally, the best cards are yet to come (I won't comment on unreleased cards). The graphics engine is being designed to be scalable over the next few YEARS, so it might look like we are shooting a bit high for the first release, but by the time it actually ships, there will be a lot of people with brand new accelerators that won't be properly exploited by any other game.

3.3 Mar 20, 1998

Robert Duffy, the maintainer of Radiant QE4 is now "officially" in charge of further development of the editor codebase. He joins Zoid as a (part time) contractor for us.

A modified version of Radiant will be the level editor for Quake 3. The primary changes will be support for curved surfaces and more general surface shaders. All changes will be publicly released, either after Q3 ships or possibly at the release of Q3Test, depending on how things are going.

The other major effort is to get Radiant working properly on all of the 3D cards that are fielding full OpenGL ICDs. If you want to do level development, you should probably get an 8mb video card. Permedia II cards have been the mainstay for developers that can't afford intergraph systems, but 8mb rendition v2200 (thriller 3D) cards are probably a better bet as soon as their ICD gets all the bugs worked out.

3.4 I just shut down the last of the NEXTSTEP systems running at id. (Mar 21, 1998)

We hadn't really used them for much of anything in the past year, so it was just easier to turn them off than to continue to administer them.

Most of the intel systems had already been converted to NT or 95, and Onethumb gets all of our old black NeXT hardware, but we have four nice HP 712/80 workstations that can't be used for much of anything.

If someone can put these systems to good use (a dallas area unix hacker), you can have them for free. As soon as they are spoken for, I will update my .plan, so check immediately before sending me email.

You have to come by our office (in Mesquite) and do a fresh OS install here before you can take one. There may still be junk on the HD, and I can't spend the time to clean them myself. You can run either NEXTSTEP 3.3 or HP/UX. These are NOT intel machines, so you can't run dos or windows. I have NS CD's here, but I can't find the original HP/UX CDs. Bring your own if that's what you want.

I'm a bit nostalgic about the NeXT systems – the story in the Id Anthology is absolutely true: I walked through a mile of snow to the bank to pay for our first workstation. For several years, I considered it the best development environment around. It still has advantages today, but you can't do any accelerated 3D work on it.

I had high hopes for rhapsody, but even on a top of the line PPC, it felt painfully sluggish compared to the NT workstations I use normally, and apple doesn't have their 3D act together at all.

Its kind of funny, but even through all the D3D/OpenGL animosity, I think Windows NT is the best place to do 3D graphics development.

All gone!

Paul Magyar gets the last (slightly broken) one.

Bob Farmer gets the third.

Philip Kizer gets the second one.

Kyle Bousquet gets the first one.

3/21 pt 2

I haven't given up on rhapsody yet. I will certainly be experimenting with the release version when it ships, but I have had a number of discouraging things happen. Twice I was going to go do meetings at apple with all relevent people, but the people setting it up would get laid off before the meetings happened. Several times I would hear encouraging rumors about various things, but they never panned out. We had some biz discussions with apple about rhapsody, but they were so incredibly cautious about targeting rhapsody for consumer apps at the expense of macos that I doubted their resolve.

I WANT to help. Maybe post-E3 we can put something together.

The SGI/microsoft deal fucked up a lot of the 3D options. The codebase that everyone was using to develop OpenGL ICDs is now owned by microsoft, so it is unlikely any of them will ever be allowed to port to rhapsody (or linux, or BeOS).

That is one of the things I stress over – The Right Thing is clear, but its not going to happen because of biz moves. It would be great if ATI, which has video drivers for win, rhapsody, linux, and BeOS, could run the same ICD on all those platforms.

3.5 Mar 26, 1998

I haven't even seen the "BeOS port of Quake". Stop emailing me about aproving it. I told one of the Lion developers he could port it to BeOS in his spare time, but I haven't seen any results from it.

-

There is a public discussion / compilation going on at openquake for suggestions to improve technical aspects of quake 3:

http://www.openquake.org/q3suggest/

This is sooo much better than just dropping me an email when a thought hits you. There are many, many thousands of you out there, and there needs to be some filtering process so we can get the information efficiently.

We will read and evaluate everything that makes it through the discussion process. There are two possible reasons why features don't make it into our games - either we decide that the effort is better spent elsewhere, or we just don't think about it. Sometimes the great ideas are completely obvious when suggested, but were just missed. That is what I most hope to see.

When the suggestions involve engineering tradeoffs and we have to consider the implementation effort of a feature vs its benefits, the best way to convince us to pursue it is to specify EXACTLY what benefits would be gained by undertaking the work, and specifying a clean interface to the feature from the file system data and the gamex86 code.

We hack where necessary, but I am much more willing to spend my time on an elegant extension that has multiple uses, rather than adding api bulk for specific features. Point out things that are clunky and inelegant in the current implementation. Even if it doesn't make any user visible difference, restructuring api for cleanliness is still a worthwhile goal.

We have our own ideas about game play features, so we may just disagree with you. Even if you-and-all-your-friends are SURE that your suggestions will make the game a ton better, we may not think it fits with our overall direction. We aren't going to be all things to all people, and we don't design by committee.

Chapter 4

April

4.1 Drag strip day! (Apr 02, 1998)

Most of the id guys, John Romero from ION, and George and Alan from 3drealms headed to the Ennis dragstrip today.

Nobody broke down, and some good times were posted.

11.9 @ 122 John Carmack F40
12.2 @ 122 George Broussard custom turbo 911
12.4 @ 116 Brian Hook Viper GTS
13.4 @ 106 John Romero custom turbo testarossa
13.6 @ 106 Todd Hollenshead 'vette
13.9 @ 100 Paul Steed 911
14.0 @ 99 Tim Willits 911
14.3 @ 101 Bear Turbo Supra
14.4 @ 98 Alan Blum turbo rx-7
14.7 @ 92 Brandon James M3
15.3 @ 92 Christian Boxster
15.5 @ 93 Jen (Hook's Chick) Turbo Volvo
16.1 @ 89 Ms. Donna Mustang GT
17.4 @ 82 Anna (Carmack's Chick) Honda Accord
18.1 @ 75 Jennifer (Jim Molinets' Chick) Saturn

We had three significant no-shows for various reasons: my TR, Adrian's viper, and Cash's supercharged M3 were all in the shop.

4.2 Things are progressing reasonably well on the Quake 3 engine. (Apr 08, 1998)

Not being limited to supporting a 320*240 256 color screen is very, very nice, and will make everyone's lives a lot easier.

All of our new source artwork is being done in 24 bit TGA files, but the engine will continue to load .wal files and .pcx files for developer's convenience. Each pcx can have its own palette now though, because it is just converted to 24 bit at load time.

Q3 is going to have a fixed virtual screen coordinate system, independant of resolution. I tried that back in the original glquake, but the fixed coordinate system was only 320*200, which was excessively low. Q2 went with a dynamic layout at different resolutions, which was a pain, and won't scale to the high resolutions that very fast cards will be capable of running at next year.

All screen drawing is now done assuming the screen is 640*480, and everything is just scaled as you go higher or lower. This makes laying out status bars and HUDs a ton easier, and will let us do a lot cooler looking screens.

There will be an interface to let game dlls draw whatever they want on the screen, precisely where they want it. You can suck up a lot of network bandwidth doing that though, so some care will be needed.

-

Going to the completely opposite end of the hardware spectrum from quake 3..

I have been very pleased with the fallout from the release of the DOOM source code.

At any given spot in design space, there are different paths you can take to move forward. I have usually chosen to try to make a large step to a completely new area, but the temptation is there to just clean up and improve in the same area, continuously polishing the same program.

I am enjoying seeing several groups pouring over DOOM, fixing it up and enhancing it. Cleaning up long standing bugs. Removing internal limitations. Orthogonalizing feature sets. Etc.

The two that I have been following closest are Team TNT's BOOM engine project, which is a clear headed, well engineered improvement on the basic DOOM technical decisions, and Bruce Lewis' glDoom project.

Any quakers feeling nostalgic should browse around:

`http://www.doomworld.com/`

4.3 Apr 16, 1998

F40 + $465,000 = F50

4.4 Yes, I bought an F50. No, I don't want a McLaren. (Apr 17, 1998)

We will be going back to the dragstrip in a couple weeks, and I will be exercising both the F50 and the TR there. Cash's supercharged M3 will probably show some of the porsches a thing or two, as well.

I'll probably rent a road coarse sometime soon, but I'm not in too much of a hurry to run the F50 into the weeds.

My TR finally got put back together after a terrific nitrous explosion just before the last dragstrip. It now makes 1000.0 hp at the rear wheels. Contrast that with the 415 rear wheel hp that the F40 made. Of course, a loaded testarossa does weigh about 4000 lbs..

My project car is somewhat nearing completion. My mechanic says it will be running in six weeks, but mechanics can be even more optimistic than software developers. :) I'm betting on fall. It should really be something when completed: a carbon fiber bodied ferrari GTO with a custom, one-of-a kind billet alluminum 4 valve DOHC 5.2L V12 with twin turbos running around 30 lbs of boost. It should be good for quite a bit more hp than my TR, and the entire car will only weigh 2400 lbs.

—

The distance between a cool demo and production code is vast. Two months ago, I had some functional demos of several pieces of the Quake 3 rendering tech, but today it still isn't usable as a full replacement for ref_gl yet.

Writing a modern game engine is a lot of work.

The new architecture is turning out very elegent. Not having to support software rendering or color index images is helping a lot, but it is also nice to reflect on just how much I have learned in the couple years since the original Quake renderer was written.

My C coding style has changed for Quake 3, which is going to give me a nice way of telling at a glance which code I have or haven't touched since Quake 2. In fact, there have been enough evolutions in my style that you can usually tell what year I wrote a piece of code by just looking at a single function:

```
/*
=============
=
= Function headers like this are DOOM or earlier
=
=============
*/

/*
=============
Function Headers like this are Quake or later
=============
```

4.4. YES, I BOUGHT AN F50. NO, I DON'T WANT A MCLAREN. (APR 17, 1998)

```
*/

{
// comments not indented were written on NEXTSTEP
// (quake 1)

    // indented comments were written on
    // Visual C++ (glquake / quakeworld, quake2)
}

for (testnum=0 ; testnum<4 ; testnum++)
{   // older coding style
}

for (testNumber = 0 ; testNumber < 4 ; testNumber++) {
    // quake 3 coding style
}
```

4.5 F50 pros and cons vs F40 (Apr 22, 1998)

The front and rear views are definately cooler on the F50, but I think I like the F40 side view better. I haven't taken the top off the F50 yet, though (its supposed to be a 40 minute job..).

Adjustable front suspension. Press a button and it raises two inches, which means you can actually drive it up into strip malls. The F40 had to be driven into my garage at an angle to keep the front from rubbing. This makes the car actually fairly practical for daily driving.

Drastically better off idle torque. You have to rev the F40 a fair amount to even get it moving, and if you are moving at 2000 rpm in first gear, a honda can pull away from you until it starts making boost at 3500 rpm. The f50 has enough torque that you don't even need to rev to get moving, and it goes quite well by just flooring it after you are moving. No need to wreck a clutch by slipping it out from 4000 rpm.

Much nicer clutch. The F40 clutch was a very low-tech single disk clutch that required more effort than on my crazy TR with over twice the torque.

Better rearward visibility. The F40's lexan fastback made everything to your rear a blur.

Better shifting. A much smoother six speed than the F40's five speed.

Better suspension. Some bumps that would upset the F40 badly are handled without any problems.

Better aerodynamics. A flat underbody with tunnels is a good thing if you are going to be moving at very high speeds.

I beleive the F50 could probably lap a road coarse faster than the F40, but in a straight line, the F40 is faster. The F50 felt a fair amount slower, but I was chalking that up to the lack of non-linear turbo rush. Today I drove it down to the dyno and we got real numbers.

It only made 385 hp at the rear wheels, which is maybe 450 at the crank if you are being generous. The F40 made 415, but that was with the boost cranked up a bit over stock.

We're going to have to do something about that.

I'm thinking that a mild twin-turbo job will do the trick. Six pounds of boost should get it up to a health 500 hp at the rear wheels, which will keep me happy. I don't want to turn it into a science project like my TR, I just want to make sure it is well out of the range of any normal cars.

I may put that in line after my GTO gets finished.

Chapter 5

May

5.1 May 02, 1998

The rcon backdoor was added to help the development of QuakeWorld (It is not present in Quake 1). At the time, attacking Quake servers with spoofed packets was not the popular sport it seems to have become with Quake 2, so I didn't think much about the potential for exploitation.

The many forced releases of Quake 2 due to hacker attacks has certainly taught me to be a lot more wary.

It was a convenient feature for us, but it turned out to be irresponsible. Sorry.

There will be new releases of QuakeWorld and Quake 2 soon.

5.2 May 04, 1998

Here are some notes on a few of the technologies that I researched in preparing for the Quake3/trinity engine. I got a couple months of pretty much wide open research done at the start, but it turned out that none of the early research actually had any bearing on the directions I finally decided on. Ah well, I learned a lot, and it will probably pay off at some later time.

I spent a little while doing some basic research with lummigraphs, which are sort of a digital hologram. The space requirements are IMMENSE, on the order of several gigs uncompressed for even a single full sized room. I was considering the possibility of using very small lumigraph fragments (I called them ”lumigraphlets”) as imposters for large clusters of areas, similar to aproximating an area with a texture map, but it would effectively be a view dependent texture.

The results were interesting, but transitioning seamlessly would be difficult, the memory was still large, and it has all the same caching issues that any impostor scheme has.

Another aproach I worked on was basically extending the sky box code style of rendering from quake 2 into a complete rendering system. Take a large number of environment map snapshots, and render a view by interpolating between up to four maps (if in a tetrahedral arangement) based on the view position.

A simple image based interpolating doesn't convey a sense of motion, because it basically just ghosts between seperate points unless the maps are VERY close together reletive to the nearest point visible in the images.

If the images that make up the environment map cube also contain depth values at some (generally lower) resolution, instead of rendering the environment map as six big flat squares at infinity, you can render it as a lot of little triangles at the proper world coordinates for the individual texture points. A single environment map like this can be walked around in and gives a sense of motion. If you have multiple maps from nearby locations, they can be easily blended together. Some effort should be made to nudge the mesh samples so that as many points are common between the maps as possible, but even a regular grid works ok.

You get texture smearing when occluded detail should be revealed, and if you move too far from the original camera point the textures blur out a lot, but it is still a very good effect, is completely complexity insensitive, and is aliasing free except when the view position causes a silhouette crease in the depth data.

Even with low res environment maps like in Quake2, each snapshot would consume 700k, so taking several hundred environment images throughout a level would generate too much data. Obviously there is a great deal of redundancy - you will have several environment maps that contain the same wall image, for instance. I had an interesting idea for compressing it all. If you ignore specular lighting and atmospheric effects, any surface that is visible in multiple environment maps can be represented by a single copy of it and perspective transformation of that image. Single image, transformations, sounds like.. fractal compression. Normal fractal compression only deals with affine maps, but the extension to projective maps seems logical.

I think that a certain type of game could be done with a technology like that, but in the end, I didn't think it was the right direction for a first person shooter.

There is a tie in between lummigraphs, multiple environment maps, specularity, convolution, and dynamic indirect lighting. Its nagging at me, but it hasn't come completely clear.

Other topics for when I get the time to write more:

Micro environment map based model lighting. Convolutions of environment maps by phong exponent, exponent of one with normal vector is diffuse lighting.

Full surface texture representation. Interior antaliasing with edge matched texels.

Octree represented surface voxels. Drawing and tracing.

Bump mapping, and why most of the aproaches being suggested for hardware are bogus.

Parametric patches vs implicit functions vs subdivision surfaces.

Why all analytical boundary representations basically suck.

Finite element radiosity vs photon tracing.

etc.

5.3 Here is an example of some bad programming in quake (May 17, 1998)

There are three places where text input is handled in the game: the console, the chat line, and the menu fields. They all used completely different code to manage the input line and display the output. Some allowed pasting from the system clipboard, some allowed scrolling, some accepted unix control character commands, etc. A big mess.

Quake 3 will finally have full support for international keyboards and character sets. This turned out to be a bit more trouble than expected because of the way Quake treated keys and characters, and it led to a rewrite of a lot of the keyboard handling, including the full cleanup and improvement of text fields.

A similar cleanup of the text printing hapened when Cash implemented general colored text: we had at least a half dozen different little loops to print strings with slightly different attributes, but now we have a generalized one that handles embedded color commands or force-to-color printing.

Amidst all the high end graphics work, sometimes it is nice to just fix up something elementary.

5.4 May 19, 1998

A 94 degree day at the dragstrip today. Several 3drealms and Norwood Autocraft folk also showed up to run. We got to weigh most of the cars on the track scales, which gives us a few more data points.

11.6 @ 125 Bob Norwood's ferrari P4 race car (2200 lbs)
11.9 @ 139 John Carmack's twin turbo testarossa (3815 lbs)
11.9 @ 117 Paul Steed's YZF600R bike
12.1 @ 122 John Carmack's F50 (3205 lbs)
12.3 @ 117 Brian's Viper GTS (3560 lbs)
13.7 @ 103 John Cash's supercharged M3

14.0 @ 96 Scott Miller's lexus GS400
15.0 @ ??? Someone's volkswagon GTI
15.1 @ ??? Christian's boxter (with Tim driving)

Weight is the key for good ETs. The TR has considerably better power to weight ratio than the P4, but it can't effectively use most of the power until it gets into third gear. The viper is actually making more power than the F50, (Brian got a big kick out of that after his dyno run) but 350 lbs more than compensated for it.

I wanted to hit 140 in the TR, but the clutch started slipping on the last run and I called it a day.

I was actually surprised the F50 ran 122 mph, which is the same the F40 did on a 25 degree cooler day. I was running with the top off, so it might even be capable of going a bit faster with it on.

The F50 and the viper were both very consistant performers, but the TR and the supercharged M3 were all over the place with their runs.

Brian nocked over a tenth off of his times even in spite of the heat, due to launch practice and some inlet modifications. He also power shifted on his best run.

It was pretty funny watching the little volkswagon consistantly beat up on a tire shredding trans-am.

George Broussard had his newly hopped up 911 turbo, but it broke the trans on its very first run. We were expecting him to be in the 11's.

We probably won't run again until either I get the F50 souped up, or my GTO gets finished.

5.5 May 22, 1998

Congratulations to Epic, Unreal looks very good.

Chapter 6

June

6.1 Jun 08, 1998

I spent quite a while investigating the limits of input under windows recently. I foudn out a few interesting things:

Mouse sampling on win95 only happens every 25ms. It doesn't matter if you check the cursor or use DirectInput, the values will only change 40 times a second.

This means that with normal checking, the mouse control will feel slightly stuttery whenever the framerate is over 20 fps, because on some frames you will be getting one input sample, and on other frames you will be getting two. The difference between two samples and three isn't very noticable, so it isn't much of an issue below 20 fps. Above 40 fps it is a HUGE issue, because the frames will be bobbing between one sample and zero samples.

I knew there were some sampling quantization issues early on, so I added the "m_filter 1" variable, but it really wasn't an optimal solution. It averaged together the samples collected at the last two frames, which worked out ok if the framerate stayed consistantly high and you were only averaging together one to three samples, but when the framerate dropped to 10 fps or so, you wound up averaging together a dozen more samples than were really needed, giving the ”rubber stick” feel to the mouse control.

I now have three modes of mouse control:

in_mouse 1: Mouse control with standard win-32 cursor calls, just like Quake 2.

in_mouse 2: Mouse control using DirectInput to sample the mouse relative counters each frame. This behaves like winquake with -dinput. There isn’t a lot of difference between this and 1, but you get a little more precision, and you never run into window clamping issues. If at some point in the future microsoft changes the implementation of DirectInput so that it processes all pending mouse events exactly when the getState call happens, this will be the ideal input mode.

in_mouse 3: Processes DirectInput mouse movement events, and filters the amount of movement over the next 25 milliseconds. This effectively adds about 12 ms of latency to the mouse control, but the movement is smooth and consistant at any variable frame rate. This will be the default for Quake 3, but some people may want the 12ms faster (but rougher) response time of mode 2.

It takes a pretty intense player to even notice the difference in most cases, but if you have a setup that can run a very consistant 30 fps you will probably apreciate the smoothness. At 60 fps, anyone can tell the difference, but rendering speeds will tend to cause a fair amount of jitter at those rates no matter what the mouse is doing.

DirectInput on WindowsNT does not log mouse events as they happen, but seems to just do a poll when called, so they can’t be filtered properly.

Keyboard sampling appears to be millisecond precise on both OS, though.

In doing this testing, it has become a little bit more tempting to try to put in more leveling optimizations to allow 60 hz framerates on the highest end hardware, but I have always shied away from targeting very high framerates as a goal, because when you miss by a tiny little bit, the drop from 60 to 30 (1 to 2 vertical retraces) fps is extremely noticable.

-

I have also concluded that the networking architecture for Quake 2 was just not the right thing. The interpolating 10 hz server made a lot of animation easier, which fit with the single player focus, but it just wasn't a good thing for internet play.

Quake 3 will have an all new entity communication mechanism that should be solidly better than any previous system. I have some new ideas that go well beyond the previous work that I did on QuakeWorld.

Its tempting to try to roll the new changes back into Quake 2, but a lot of them are pretty fundamental, and I'm sure we would bust a lot of important single player stuff while gutting the network code.

(Yes, we made some direction changes in Quake 3 since the original announcement when it was to be based on the Quake 2 game and networking with just a new graphics engine)

6.2 Jun 16, 1998

My last two .plan updates have described efforts that were not in our original plan for quake 3, which was "quake 2 game and network technology with a new graphics engine".

We changed our minds.

The new product is going to be called "Quake Arena", and will consist exclusively of deathmatch style gaming (including CTF and other derivatives). The single player game will just be a progression through a ranking ladder against bot AIs. We think that can still be made an enjoyable game, but it is definately a gamble.

In the past, we have always been designing two games at once, the single player game and the multi player game, and they often had conflicting goals. For instance, the client-server communications channel discouraged massive quantities of moving entities that would have been interesting in single player, while the maps and weapons designed for single player were not ideal for multiplayer. The largest conflict was just raw development time. Time spent on monsters is time not spent on player movement. Time spent on unit goals is time not spent on game rules.

There are many wonderful gaming experiences in single player FPS, but we are choosing to leave them behind to give us a purity of focus that will let us make significant advances in the multiplayer experience.

The emphasis will be on making every aspect as robust and high quality as possible, rather than trying to add every conceivable option anyone could want. We will not be trying to take the place of every mod ever produced, but we hope to satisfy a large part of the network gaming audience with the out of box experience.

There is a definite effect on graphics technology decisions. Much of the positive feedback in a single player FPS is the presentation of rich visual scenes, which are often at the expense of framerate. A multiplayer level still needs to make a good first impression, but after you have seen it a hundred times, the speed of the game is more important. This means that there are many aggressive graphics technologies that I will not pursue because they are not apropriate to the type of game we are creating.

The graphics engine will still be OpenGL only, with significant new features not seen anywhere before, but it will also have fallback modes to render at roughly Quake-2 quality and speed.

Chapter 7

July

7.1 Here is the real story on the movement physics changes. (Jul 04, 1998)

Zoid changed the movement code in a way that he felt improved gameplay in the 3.15 release.

We don't directly supervise most of the work Zoid does. One of the main reasons we work with him is that I respect his judgment, and I feel that his work benefits the community quite a bit with almost no effort on my part. If I had to code review every change he made, it wouldn't be worth the effort.

Zoid has "ownership" of the Quake, Glquake, and QuakeWorld codebases. We don't intend to do any more modifications at Id on those sources, so he has pretty free reign within his discretion.

We passed over the Quake 2 codebase to him for the addition of new features like auto download, but it might have been a bit premature, because official mission packs were still in development, and unlike glquake and quakeworld, Q2 is a product that must remain official and supported, so the scope of his freedoms should have been spelled out a little more clearly.

The air movement code wasn't a good thing to change in Quake 2, because the codebase still had to support all the commercial single player levels, and subtle physics changes can have lots of unintended effects.

QuakeWorld didn't support single player maps, so it was a fine place to experiment with physics changes.

QuakeArena is starting with fresh new data, so it is also a good place to experiment with physics changes.

Quake 2 cannot be allowed to evolve in a way that detracts from the commercial single player levels.

The old style movement should not be referred to as "real world physics". None of the quake physics are remotely close to real world physics, so I don't think one way is significantly more "real" than the other. In Q2, you accelerate from 0 to 27 mph in 1/30 of a second, which just as unrealistic as being able to accelerate in midair..

7.2 Jul 05, 1998

I am not opposed to adding a flag to control the movement styles. I was rather expecting it to be made optional in 3.17, but I haven't been directly involved in the last few releases.

The way this played out in public is a bit unfortunate. Everyone at Id is busy full time with the new product, so we just weren't paying enough attention to the Quake 2 modifications. Some people managed to read into my last update that we were blaming Zoid for things. Uh, no. I think he was acting within his charter (catering to the community) very well, it just interfered with an aspect of the game that shouldn't have been modified. We just never made it explicitly clear that it shouldn't have been modified.

It is a bit amusing how after the QuakeArena anouncement, I got flamed by lots of people for abandoning single player play (even though we aren't, really) but after I say that Quake 2 can't forget that it is a single player game, I get flamed by a different set of people who think it is stupid to care about single player anymore when all ”everyone” plays is multiplayer. The joy of having a wide audience that knows your email address.

7.3 I have spent the last two days working with Apple’s Rhapsody DR2, and I like it a lot. (Jul 16, 1998)

I was dissapointed with the original DR1 release. It was very slow and seemed to have added the worst elements of the mac experience (who the hell came up with that windowshade minimizing?) while taking away some of the strengths of NEXTSTEP.

Things are a whole lot better in the latest release. General speed is up, memory consumption is down, and the UI feels consistant and productive.

Its still not as fast as windows, and probably never will be, but I think the tradeoffs are valid.

There are so many things that are just fundamentally better in the rhapsody design than in windows: frameworks, the yellow box apis, fat binaries, buffered windows, strong multi user support, strong system / local seperation, netinfo, etc.

Right now, I think WindowsNT is the best place to do graphics development work, but if the 3D acceleration issue was properly addressed on rhapsody, I think that I could be happy using it as my primary development platform.

I ported the current Quake codebase to rhapsody to test out conix’s beta OpenGL. The game isn’t really playable with the software emulated OpenGL, but it functions properly, and it makes a fine dedicated server.

We are going to try to stay on top of the portability a little better for QA. Quake 2 slid a bit because we did the development on NT instead of NEXTSTEP, and that made the irix port a lot more of a hassle than the original glquake port.

I plan on using the rhapsody system as a dedicated server during development, and Brian will be using an Alpha-NT system for a lot of testing, which should give us pretty good coverage of the portability issues.

I'm supposed to go out and have a bunch of meetings at apple next month to cover games, graphics, and hardware. Various parts of apple have scheduled meetings with me on three seperate occasions over the past couple years, but they have always been canceled for one reason or another (they laid off the people I was going to meet with once..).

I have said some negative things about MacOs before, but my knowledge of the mac is five years old. There was certainly the possibility that things had improved since then, so I spent some time browsing mac documentation recently. I was pretty amused. A stack sniffer. Patching trap vectors. Cooperative multitasking. Application memory partitions. Heh.

I'm scared of MacOS X. As far as I can tell, The basic plan is to take rhapsody and bolt all the MacOS APIs into the kernel. I understand that that may well be a sensible biz direction, but I fear it.

In other operating system news, Be has glquake running hardware accelerated on their upcoming OpenGL driver architecture. I gave them access to the glquake and quake 2 codebases for development purposes, and I expect we will work out an agreement for distribution of the ports.

Any X server vendors working on hardware accelerated OpenGL should get in touch with Zoid about interfacing and tuning with the Id OpenGL games on linux.

7.4 My F50 took some twin turbo vitamins. (Jul 29, 1998)

Rear wheel numbers: 602 HP @ 8200 rpm 418 ft-lb @ 7200 rpm

This is very low boost, but I got the 50% power increase I was looking for, and hopefully it won't be making any contributions to my piston graveyard.

There will be an article in Turbo magazine about it, and several other car magazines want to test it out. They usually start out with ”He did WHAT to an F50???” :)

Brian is getting a nitrous kit installed in his viper, and Cash just got his suspension beefed up, so we will be off to the dragstrip again next month to sort everything out again.

Chapter 8

August

8.1 Aug 17, 1998

I added support for HDTV style wide screen displays in QuakeArena, so 24” and 28” monitors can now cover the entire screen with game graphics.

On a normal 4:3 aspect ratio screen, a 90 degree horizontal field of view gives a 75 degree vertical field of view. If you keep the vertical fov constant and run on a wide screen, you get a 106 degree horizontal fov.

Because we specify fov with the horizontal measurement, you need to change fov when going into or out of a wide screen mode. I am considering changing fov to be the vertical measurement, but it would probably cause a lot of confusion if ”fov 90” becomes a big fisheye.

Many video card drivers are supporting the ultra high res settings like 1920 * 1080, but hopefully they will also add support for lower settings that can be good for games, like 856 * 480.

I spent a day out at apple last week going over technical issues.

I’m feeling a lot better about MacOS X. Almost everything I like about rhapsody will be there, plus some solid additions.

I presented the OpenGL case directly to Steve Jobs as strongly as possible.

If Apple embraces OpenGL, I will be strongly behind them. I like OpenGL more than I dislike MacOS. :)

-

Last friday I got a phone call: "want to make some exhibition runs at the import / domestic drag wars this sunday?". It wasn't particularly good timing, because the TR had a slipping clutch and the F50 still hasn't gotten its computer mapping sorted out, but we got everything functional in time.

The tech inspector said that my cars weren't allowed to run in the 11s at the event because they didn't have roll cages, so I was supposed to go easy.

The TR wasn't running its best, only doing low 130 mph runs. The F50 was making its first sorting out passes at the event, but it was doing ok. My last pass was an 11.8(oops) @ 128, but we still have a ways to go to get the best times out of it.

I'm getting some racing tires on the F50 before I go back. It sucked watching a tiny honda race car jump ahead of me off the line. :)

I think ESPN took some footage at the event.

Chapter 9

September

9.1 I just got a production TNT board installed in my Dolch today. (Sep 08, 1998)

The riva-128 was a troublesome part. It scored well on benchmarks, but it had some pretty broken aspects to it, and I never recommended it (you are better off with an intel I740).

There aren't any troublesome aspects to TNT. Its just great. Good work, Nvidia.

In terms of raw speed, a 16 bit color multitexture app (like quake / quake 2) should still run a bit faster on a voodoo2, and an SLI voodoo2 should be faster for all 16 bit color rendering, but TNT has a lot of other things going for it:

32 bit color and 24 bit z buffers. They cost speed, but it is usually a better quality tradeoff to go one resolution lower but with twice the color depth.

More flexible multitexture combine modes. Voodoo can use its multitexture for diffuse lightmaps, but not for the specular lightmaps we offer in QuakeArena. If you want shiny surfaces, voodoo winds up leaving half of its texturing power unused (you can still run with diffuse lightmaps for max speed).

Stencil buffers. There aren't any apps that use it yet, but stencil allows you to do a lot of neat tricks.

More texture memory. Even more than it seems (16 vs 8 or 12), because all of the TNT's memory can be used without restrictions. Texture swapping is the voodoo's biggest problem.

3D in desktop applications. There is enough memory that you don't have to worry about window and desktop size limits, even at 1280*1024 true color resolution.

Better OpenGL ICD. 3dfx will probably do something about that, though.

This is the shape of 3D boards to come. Professional graphics level rendering quality with great performance at a consumer price.

We will be releasing preliminary QuakeArena benchmarks on all the new boards in a few weeks. Quake 2 is still a very good benchmark for moderate polygon counts, so our test scenes for QA involve very high polygon counts, which stresses driver quality a lot more. There are a few surprises in the current timings..

-

A few of us took a couple days off in vegas this weekend. After about ten hours at the tables over friday and saturday, I got a tap on the shoulder..

Three men in dark suits introduced themselves and explained that I was welcome to play any other game in the casino, but I am not allowed to play blackjack anymore.

Ah well, I guess my blackjack days are over. I was actually down a bit for the day when they booted me, but I made +$32k over five trips to vegas in the past two years or so.

I knew I would get kicked out sooner or later, because I don't play "safely". I sit at the same table for several hours, and I range my bets around 10 to 1.

9.1. I JUST GOT A PRODUCTION TNT BOARD INSTALLED IN MY DOLCH TODAY. (SEP 08, 1998)

9.2 Sep 10, 1998

I recently set out to start implementing the dual-processor acceleration for QA, which I have been planning for a while. The idea is to have one processor doing all the game processing, database traversal, and lighting, while the other processor does absolutely nothing but issue OpenGL calls.

This effectively treats the second processor as a dedicated geometry accelerator for the 3D card. This can only improve performance if the card isn't the bottleneck, but voodoo2 and TNT cards aren't hitting their limits at 640*480 on even very fast processors right now.

For single player games where there is a lot of cpu time spent running the server, there could conceivably be up to an 80% speed improvement, but for network games and timedemos a more realistic goal is a 40% or so speed increase. I will be very satisfied if I can makes a dual pentium-pro 200 system perform like a pII-300.

I started on the specialized code in the renderer, but it struck me that it might be possible to implement SMP acceleration with a generic OpenGL driver, which would allow Quake2 / sin / halflife to take advantage of it well before QuakeArena ships.

It took a day of hacking to get the basic framework set up: an smpgl.dll that spawns another thread that loads the original oepngl32.dll or 3dfxgl.dll, and watches a work que for all the functions to call.

I get it basically working, then start doing some timings. Its 20% slower than the single processor version.

I go in and optimize all the queing and working functions, tune the communications facilities, check for SMP cache collisions, etc.

After a day of optimizing, I finally squeak out some performance gains on my tests, but they aren't very impressive: 3% to 15% on one test scene, but still slower on the another one.

This was fairly depressing. I had always been able to get pretty much linear speedups out of the multithreaded utilities I wrote, even up to sixteen processors. The difference is that the utilities just split up the work ahead of time, then don't talk to each other until they are done, while here the two threads work in a high bandwidth producer / consumer relationship.

I finally got around to timing the actual communication overhead, and I was appalled: it was taking 12 msec to fill the que, and 17 msec to read it out on a single frame, even with nothing else going on. I'm surprised things got faster at all with that much overhead.

The test scene I was using created about 1.5 megs of data to relay all the function calls and vertex data for a frame. That data had to go to main memory from one processor, then back out of main memory to the other. Admitedly, it is a bitch of a scene, but that is where you want the acceleration..

The write times could be made over twice as fast if I could turn on the PII's write combining feature on a range of memory, but the reads (which were the gating factor) can't really be helped much.

Streaming large amounts of data to and from main memory can be really grim. The next write may force a cache writeback to make room for it, then the read from memory to fill the cacheline (even if you are going to write over the entire thing), then eventually the writeback from the cache to main memory where you wanted it in the first place. You also tend to eat one more read when your program wants to use the original data that got evicted at the start.

What is really needed for this type of interface is a streaming read cache protocol that performs similarly to the write combining: three dedicated cachelines that let you read or write from a range without evicting other things from the cache, and automatically prefetching the next cacheline as you read.

Intel's write combining modes work great, but they can't be set directly from user mode. All drivers that fill DMA buffers (like OpenGL ICDs..) should definately be using them, though.

Prefetch instructions can help with the stalls, but they still don't prevent all the wasted cache evictions.

It might be possible to avoid main memory alltogether by arranging things so that the sending processor ping-pongs between buffers that fit in L2, but I'm not sure if a cache coherent read on PIIs just goes from one L2 to the other, or if it becomes a forced memory transaction (or worse, two memory transactions). It would also limit the maximum amount of overlap in some situations. You would also get cache invalidation bus traffic.

I could probably trim 30% of my data by going to a byte level encoding of all the function calls, instead of the explicit function pointer / parameter count / all-parms-are-32-bits that I have now, but half of the data is just raw vertex data, which isn't going to shrink unless I did evil things like quantize floats to shorts.

Too much effort for what looks like a reletively minor speedup. I'm giving up on this aproach, and going back to explicit threading in the renderer so I can make most of the communicated data implicit.

Oh well. It was amusing work, and I learned a few things along the way.

Chapter 10

October

10.1 Oct 14, 1998

It has been difficult to write .plan updates lately. Every time I start writing something, I realize that I'm not going to be able to cover it satisfactorily in the time I can spend on it. I have found that terse little comments either get misinterpreted, or I get deluged by email from people wanting me to expand upon it.

I wanted to do a .plan about my evolving thoughts on code quality and lessons learned through quake and quake 2, but in the interest of actually completing an update, I decided to focus on one change that was intended to just clean things up, but had a surprising number of positive side effects.

Since DOOM, our games have been defined with portability in mind. Porting to a new platform involves having a way to display output, and having the platform tell you about the various relevant inputs. There are four principle inputs to a game: keystrokes, mouse moves, network packets, and time. (If you don't consider time an input value, think about it until you do - it is an important concept)

These inputs were taken in separate places, as seemed logical at the time. A function named Sys_SendKeyEvents() was called once a frame that would rummage through whatever it needed to on a system level, and call back into game functions like Key_Event(key, down) and IN_MouseMoved(dx, dy). The network system dropped into system specific code to check for the arrival of packets. Calls to Sys_Milliseconds() were littered all over the code for various reasons.

I felt that I had slipped a bit on the portability front with Q2 because I had been developing natively on windows NT instead of cross developing from NEXTSTEP, so I was reevaluating all of the system interfaces for Q3.

I settled on combining all forms of input into a single system event queue, similar to the windows message queue. My original intention was to just rigorously define where certain functions were called and cut down the number of required system entry points, but it turned out to have much stronger benefits.

With all events coming through one point (The return values from system calls, including the filesystem contents, are "hidden" inputs that I make no attempt at capturing,), it was easy to set up a journalling system that recorded everything the game received. This is very different than demo recording, which just simulates a network level connection and lets time move at its own rate. Realtime applications have a number of unique development difficulties because of the interaction of time with inputs and outputs.

Transient flaw debugging. If a bug can be reproduced, it can be fixed. The nasty bugs are the ones that only happen every once in a while after playing randomly, like occasionally getting stuck on a corner. Often when you break in and investigate it, you find that something important happened the frame before the event, and you have no way of backing up. Even worse are realtime smoothness issues - was that jerk of his arm a bad animation frame, a network interpolation error, or my imagination?

Accurate profiling. Using an intrusive profiler on Q2 doesn't give accurate results because of the realtime nature of the simulation. If the program is running half as fast as normal due to the instrumentation, it has to do twice as much server simulation as it would if it wasn't instrumented, which also goes slower, which compounds the problem. Aggressive instrumentation can slow it down to the point of being completely unplayable.

Realistic bounds checker runs. Bounds checker is a great tool, but you just can't interact with a game built for final checking, its just waaaaay too slow. You can let a demo loop play back overnight, but that doesn't exercise any of the server or networking code.

The key point: Journaling of time along with other inputs turns a real-time application into a batch process, with all the attendant benefits for quality control and debugging. These problems, and many more, just go away. With a full input trace, you can accurately restart the session and play back to any point (conditional breakpoint on a frame number), or let a session play back at an arbitrarily degraded speed, but cover exactly the same code paths..

I'm sure lots of people realize that immediately, but it only truly sunk in for me recently. In thinking back over the years, I can see myself feeling around the problem, implementing partial journaling of network packets, and included the "fixedtime" cvar to eliminate most timing reproducibility issues, but I never hit on the proper global solution. I had always associated journaling with turning an interactive application into a batch application, but I never considered the small modification necessary to make it applicable to a realtime application.

In fact, I was probably blinded to the obvious because of one of my very first successes: one of the important technical achievements of Commander Keen 1 was that, unlike most games of the day, it adapted its play rate based on the frame speed (remember all those old games that got unplayable when you got a faster computer?). I had just resigned myself to the non-deterministic timing of frames that resulted from adaptive simulation rates, and that probably influenced my perspective on it all the way until this project.

Its nice to see a problem clearly in its entirety for the first time, and know exactly how to address it.

Chapter 11

November

11.1 Nov 03, 1998

This was the most significant thing I talked about at The Frag, so here it is for everyone else.

The way the QA game architecture has been developed so far has been as two seperate binary dll's: one for the server side game logic, and one for the client side presentation logic.

While it was easiest to begin development like that, there are two crucial problems with shipping the game that way: security and portability.

It's one thing to ask the people who run dedicated servers to make informed decisions about the safety of a given mod, but its a completely different matter to auto-download a binary image to a first time user connecting to a server they found.

The quake 2 server crashing attacks have certainly proven that there are hackers that enjoy attacking games, and shipping around binary code would be a very tempting opening for them to do some very nasty things.

With quake and Quake 2, all game modifications were strictly server side, so any port of the game could connect to any server without problems. With Quake 2's binary server dll's not all ports could necessarily run a server, but they could all play.

With significant chunks of code now running on the client side, if we stuck with binary dll's then the less popular systems would find that they could not connect to new servers because the mod code hadn't been ported. I considered having things set up in such a way that client game dll's could be sort of forwards-compatable, where they could always connect and play, but new commands and entity types just might now show up. We could also GPL the game code to force mod authors to release source with the binaries, but that would still be inconvenient to deal with all the porting.

Related both issues is client side cheating. Certain cheats are easy to do if you can hack the code, so the server will need to verify which code the client is running. With multiple ported versions, it wouldn't be possible to do any binary verification.

If we were willing to wed ourselves completely to the windows platform, we might have pushed ahead with some attempt at binary verification of dlls, but I ruled that option out. I want QuakeArena running on every platform that has hardware accelerated OpenGL and an internet connection.

The only real solution to these problems is to use an interpreted language like Quake 1 did. I have reached the conclusion that the benefits of a standard language outweigh the benefits of a custom language for our purposes. I would not go back and extend QC, because that stretches the effort from simply system and interpreter design to include language design, and there is already plenty to do.

I had been working under the assumption that Java was the right way to go, but recently I reached a better conclusion.

The programming language for QuakeArena mods is interpreted ANSI C. (well, I am dropping the double data type, but otherwise it should be pretty conformant)

The game will have an interpreter for a virtual RISC-like CPU. This should have a minor speed benefit over a byte-coded, stack based java interpreter. Loads and stores are confined to a preset block of memory, and access to all external system facilities is done with system traps to the main game code, so it is completely secure.

The tools necessary for building mods will all be freely available: a modified version of LCC and a new program called q3asm. LCC is a wonderful project - a cross platform, cross compiling ANSI C compiler done in under 20K lines of code. Anyone interested in compilers should pick up a copy of ”A retargetable C compiler: design and implementation” by Fraser and Hanson.

You can’t link against any libraries, so every function must be resolved. Things like strcmp, memcpy, rand, etc. must all be implemented directly. I have code for all the ones I use, but some people may have to modify their coding styles or provide implementations for other functions.

It is a fair amount of work to restructure all the interfaces to not share pointers between the system and the games, but it is a whole lot easier than porting everything to a new language. The client game code is about 10k lines, and the server game code is about 20k lines.

The drawback is performance. It will probably perform somewhat like QC. Most of the heavy lifting is still done in the builtin functions for path tracing and world sampling, but you could still hurt yourself by looping over tons of objects every frame. Yes, this does mean more load on servers, but I am making some improvements in other parts that I hope will balance things to about the way Q2 was on previous generation hardware.

There is also the amusing avenue of writing hand tuned virtual assembly assembly language for critical functions..

I think this is The Right Thing.

11.2 Nov 04, 1998

More extensive comments on the interpreted-C decision later, but a quick note: the plan is to still allow binary dll loading so debuggers can be used, but it should be interchangable with the interpreted code. Client modules can only be debugged if the server is set to allow cheating, but it would be possible to just use the binary interface for server modules if you wanted to sacrifice portability. Most mods will be able to be implemented with just the interpreter, but some mods that want to do extensive file access or out of band network communications could still be implemented just as they are in Q2. I will not endorse any use of binary client modules, though.

Chapter 12

December

12.1 Dec 29, 1998

I am considering taking a shortcut with my virtual machine implementation that would make the integration a bit easier, but I'm not sure that it doesn't compromise the integrity of the base system.

I am considering allowing the interpreted code to live in the global address space, instead of a private 0 based address space of its own. Store instructions from the VM would be confined to the interpreter's address space, but loads could access any structures.

On the positive side:

This would allow full speed (well, full interpreted speed) access to variables shared between the main code and the interpreted modules. This allows system calls to return pointers, instead of filling in allocated space in the interpreter's address space.

For most things, this is just a convenience that will cut some development time. Most of the shared accesses could be recast as "get" system calls, and it is certainly arguable that that would be a more robust programming style.

The most prevelent change this would prevent is all the cvar_t uses. Things could stay in the same style as Q2, where cvar accesses are free and transparantly updated. If the interpreter lives only in its own address space, then cvar access would have to be like Q1, where looking up a variable is a potentially time consuming operation, and you wind up adding lots of little cvar caches that are updated every from or restart.

On the negative side:

A client game module with a bug could cause a bus error, which would not be possible with a pure local address space interpreter.

I can't think of any exploitable security problems that read only access to the entire address space opens, but if anyone thinks of something, let me know.

12.2 Dec 30, 1998

I got several vague comments about being able to read "stuff" from shared memory, but no concrete examples of security problems.

However, Gregory Maxwell pointed out that it wouldn't work cross platform with 64 bit pointer environments like linux alpha. That is a killer, so I will be forced to do everything the hard way. Its probably for the best, from a design standpoint anyway, but it will take a little more effort.