John Carmack Archive - .plan (1999)

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

March 18, 2007

Contents

1 January **6**

1.1 Jan 10, 1999 6

1.2 Jan 29, 1999 11

2 March **14**

2.1 Mar 03, 1999 14

2.2 First impressions of the SGI visual workstation 320 (Mar 17, 1999) 15

3 April **18**

3.1 We are finally closing in on the first release of Q3test. (Apr 24, 1999) 18

3.2 Apr 25, 1999 20

3.3 Interpreting the lagometer (the graph in the lower right corner) (Apr 26, 1999) 23

3.4 Apr 27, 1999 25

3.5 Apr 28, 1999 26

3.6 Apr 29, 1999 26

3.7 Apr 30, 1999 27

4 May **28**

4.1 May 04, 1999 28

4.2 May 05, 1999 29

4.3 May 07, 1999 29

4.4 May 08, 1999 29

4.5 May 09, 1999 30

4.6 May 10, 1999 31

4.7 May 11, 1999 32

4.8 May 12, 1999 35

4.9 Now that all the E3 stuff is done with, I can get back to work... (May 19, 1999) 37

4.10 May 22, 1999 38

4.11 May 26, 1999 40

4.12 May 27, 1999 41

4.13 May 30, 1999 41

5 June **45**

5.1 Whee! Lots of hate mail from strafe-jupers! (Jun 03, 1999) . 45

5.2 Jun 27, 1999 47

6 July **52**

6.1 AMD K7 cpus are very fast. (Jul 03, 1999) 52

6.2 Jul 24, 1999 54

6.3 Jul 29, 1999 56

6.4 I apologize for not getting the Mac version released this last time. (Jul 30, 1999) 57

7 August **58**

7.1 Aug 01, 1999 58

7.2 Aug 16, 1999 58

7.3 Aug 26, 1999 63

7.4 Aug 28, 1999 66

8 September **68**

8.1 Sep 01, 1999 68

8.2 I have been getting a lot of requests for commentary on two subjects lately (Sep 02, 1999) 70

8.3 Sep 06, 1999 71

8.4 Ok, obviously we didn't get a release out in the middle of the month... (Sep 28, 1999) 72

8.5 Sep 29, 1999 74

9 October **76**

9.1 Oct 01, 1999 76

9.2 Oct 04, 1999 77

9.3 Oct 05, 1999 78

9.4 Oct 07, 1999 78

9.5 Oct 11, 1999 79

9.6 Oct 14, 1999 79

9.7 Oct 17, 1999 80

9.8 An announcement (Oct 23, 1999) 82

10 November 84

10.1 Nov 01, 1999 84

10.2 Nov 02, 1999 85

10.3 Nov 03, 1999 85

10.4 Nov 05, 1999 86

10.5 Nov 09, 1999 87

10.6 Nov 11, 1999 88

10.7 Nov 13, 1999 88

10.8 The demo test is built. It should go up in a couple hours if nothing explodes. (Nov 14, 1999) 89

10.9 Nov 15, 1999 89

10.10The demo servers have general purpose flood-protection that has caused some confusion. (Nov 16, 1999) 91

10.11The mac version is out. Go to www.quake3arena.com for links. (Nov 18, 1999) 92

10.12Nov 19, 1999 95

10.13Nov 21, 1999 95

10.14d:> mkdir research (Nov 22, 1999) 95

11 December 97

11.1 WANTED: Independant OpenGL conformance nazi (Dec 12, 1999) 97

11.2 Dec 15, 1999 99

11.3 Dec 21, 1999 99

11.4 Dec 25, 1999 100
11.5 I have been playing a lot of Q3 on a 28.8 modem for the last several days. (Dec 29, 1999) 101
11.6 Dec 30, 1999 103

Chapter 1

January

1.1 Jan 10, 1999

Ok, many of you have probably heard that I spoke at the macworld keynote on tuesday. Some information is probably going to get distorted in the spinning and retelling, so here is an info dump straight from me:

Q3test, and later the full commercial Quake3: Arena, will be simultaniously released on windows, mac, and linux platforms.

I think Apple is doing a lot of things right. A lot of what they are doing now is catch-up to wintel, but if they can keep it up for the next year, they may start making a really significant impact.

I still can't give the mac an enthusiastic recommendation for sophisticated users right now because of the operating system issues, but they are working towards correcting that with MacOS X.

The scoop on the new G3 mac hardware:

Basically, its a great system, but Apple has oversold its performance reletive to intel systems. In terms of timedemo scores, the new G3 systems should be near the head of the pack, but there will be intel systems outperforming them to some degree. The mac has not instantly become a "better" platform for games than wintel, it has just made a giant leap from the back of the pack to near the front.

I wish Apple would stop quoting ”Bytemarks”. I need to actually look at the contents of that benchmark and see how it can be so misleading. It is pretty funny listening to mac evangelist types try to say that an iMac is faster than a pentium II-400. Nope. Not even close.

From all of my tests and experiments, the new mac systems are basically as fast as the latest pentium II systems for general cpu and memory performance. This is plenty good, but it doesn't make the intel processors look like slugs.

Sure, an in-cache, single precision, multiply-accumulate loop could run twice as fast as a pentium II of the same clock rate, but conversly, a double precision add loop would run twice as fast on the pentium II.

Spec95 is a set of valid benchmarks in my opinion, and I doubt the PPC systems significantly (if at all) outperform the intel systems.

The IO system gets mixed marks. The 66 mhz video slot is a good step up from 33 mhz pci in previous products, but that's still half the bandwidth of AGP 2X, and it can't texture from main memory. This will have a small effect on 3D gaming, but not enough to push it out of its class.

The 64 bit pci slots are a good thing for network and storage cards, but the memory controller doesn't come close to getting peak utilization out of it. Better than normal pci, though.

The video card is almost exactly what you will be able to get on the pc side: a 16 mb rage-128. Running on a 66mhz pci bus, it's theoretical peak performance will be midway between the pci and agp models on pc systems for command traffic limited scenes. Note that current games are not actually command traffic limited, so the effect will be significantly smaller. The fill rates will be identical.

The early systems are running the card at 75 mhz, which does put it at a slight disadvantage to the TNT, but faster versions are expected later. As far as I can tell, the rage-128 is as perfect as the TNT feature-wise. The 32 mb option is a feature ATI can hold over TNT.

Firewire is cool.

Its a simple thing, but the aspect of the new G3 systems that struck me the most was the new case design. Not the flashy plastic exterior, but the functional structure of it. The side of the system just pops open, even with the power on, and lays the motherboard and cards down flat while the disks and power supply stay in the encloser. It really is a great design, and the benefits were driven home yesterday when I had to scavenge some ram out of old wintel systems yesterday - most case designs suck really bad.

-

I could gripe a bit about the story of our (lack of) involvement with Apple over the last four years or so, but I'm calling that water under the bridge now.

After all the past fiascos, I had been basically planning on ignoring Apple until MacOS X (rhapsody) shipped, which would then turn it into a platform that I was actually interested in.

Recently, Apple made a strategic corporate decision that games were a fundamental part of a consumer oriented product line (duh). To help that out, Apple began an evaluation of what it needed to do to help game developers.

My first thought was "throw out MacOS", but they are already in the process of doing that, and its just not going to be completed overnight.

Apple has decent APIs for 2D graphics, input, sound, and networking, but they didn't have a solid 3D graphics strategy.

Rave was sort of ok. Underspecified and with no growth path, but sort of ok. Pursuing a proprietary api that wasn't competetive with other offerings would have been a Very Bad Idea. They could have tried to make it better, or even invent a brand new api, but Apple doesn't have much credebility in 3D programming.

For a while, it was looking like Apple might do something stupid, like license DirectX from microsoft and be put into a guaranteed trailing edge position behind wintel.

OpenGL was an obvious direction, but there were significant issues with the licensing and implementation that would need to be resolved.

I spent a day at apple with the various engineering teams and executives, laying out all the issues.

The first meeting didn't seem like it went all that well, and there wasn't a clear direction visible for the next two months. Finally, I got the all clear signal that OpenGL was a go and that apple would be getting both the sgi codebase and the conix codebease and team (the best possible arrangement).

So, I got a mac and started developing on it. My first weekend of effort had QuakeArena limping along while held together with duct tape, but weekend number two had it properly playable, and weekend number three had it brought up to full feature compatability. I still need to do some platform specific things with odd configurations like multi monitor and addon controlers, but basically now its just a matter of compiling on the mac to bring it up to date.

This was important to me, because I felt that Quake2 had slipped a bit in portability because it had been natively developed on windows. I like the discipline of simultanious portable development.

After 150 hours or so of concentrated mac development, I learned a lot about the platform.

CodeWarrior is pretty good. I was comfortable devloping there almost immediately. I would definately say VC++ 6.0 is a more powerful overall tool, but CW has some nice little aspects that I like. I am definately looking forward to CW for linux. Unix purists may be aghast, but I have allways liked gui dev environments more than a bunch of xterms running vi and gdb.

The hardware (even the previous generation stuff) is pretty good.

The OpenGL performance is pretty good. There is a lot of work underway to bring the OpenGL performance to the head of the pack, but the existing stuff works fine for development.

The low level operating systems SUCKS SO BAD it is hard to believe.

The first order problem is lack of memory management / protection.

It took me a while to figure out that the zen of mac development is ”be at peace while rebooting”. I rebooted my mac system more times the first weekend than I have rebooted all the WinNT systems I have ever owned. True, it has gotten better now that I know my way around a bit more, and the codebase is fully stable, but there is just no excuse for an operating system in this day and age to act like it doesn't have access to memory protection.

The first thing that bit me was the static memory allocation for the program. Modern operating systems just figure out how much memory you need, but because the mac was originally designed for systems without memory management, significant things have to be laid out ahead of time.

Porting a win32 game to the mac will probably involve more work dealing with memory than any other aspect. Graphics, sound, and networking have reasonable analogues, but you just can't rely on being able to malloc() whatever you want on the mac.

Sure, game developers can manage their own memory, but an operating system that has proper virtual memory will let you develop a lot faster.

The lack of memory protection is the worst aspect of mac development. You can just merrily write all over other programs, the development environment, and the operating system from any application.

I remember that. From dos 3.3 in 1990.

Guard pages will help catch simple overruns, but it won't do anything for all sorts of other problems.

The second order problem is lack of preemptive multitasking.

The general responsiveness while working with multiple apps is significantly worse than windows, and you often run into completely modal dialogs that don't let you do anything else at all.

A third order problem is that a lot of the interfaces are fairly clunky.

There are still many aspects of the mac that clearly show design decisions based on a 128k 68000 based machine. Wintel has grown a lot more than the mac platform did. It may have been because the intel architecture didn't evolve gracefully and that forced the software to reevaluate itself more fully, or it may just be that microsoft pushed harder.

Carbon sanitizes the worst of the crap, but it doesn't turn it into anything particularly good looking.

MacOS X nails all these problems, but thats still a ways away.

I did figure one thing out - I was always a little curious why the early BeOS advocates were so enthusiastic. Coming from a NEXTSTEP background, BeOS looked to me like a fairly interesting little system, but nothing special. To a mac developer, it must have looked like the promised land..

1.2 Jan 29, 1999

The issue of the 8192 unit map dimension limit had been nagging at me for a long time now. The reason for the limit is that coordinates are communicated over the network in 16 bit shorts divided as a sign bit, twelve unit bits, and three fractional bits. There was also another side to the representation problem, but it was rarely visible: if you timescale down, you can actually tell the fractional bit granularity in position and velocity.

The rest of the system (rendering and gameplay) has never had any issues with larger maps, even in quake 1. There are some single precision floating point issues that begin to creep in if things get really huge, but maps should be able to cover many times the current limit without any other changes.

A while ago I had changed Q3 so that the number of fractional bits was a compile time option, which allowed you to trade off fine grain precision for larger size. I was considering automatically optimizing this for each level based on its size, but it still didn't feel like a great solution.

Another aspect of the problem that wasn't visible to the public was that the the fractional quantization of position could cause the position to actually be inside a nearby solid when used for client side prediction. The code had to check for this and try to correct the situation by jittering the position in each of the possible directions it might have been truncated from. This is a potential issue whenever there is any loss of precision whatsoever in the server to client communication.

The obvious solution is to just send the full floating point value for positions, but I resisted that because the majority of our network traffic is positional updates, and I didn't want to bloat it. There have been other bandwidth savings in Q3, and LANs and LPB connections are also relevent, so I was constantly evaluating the tradeoff.

Dealing with four or five players in view isn't a real problem. The big bandwidth issues arrive when multiple players start unloading with rapid fire weapons. (as an aside, I tried making 5hz fire weapons for Q3 to save bandwidth, but no matter how much damage they did, 5hz fire rates just seemed to feel slow and weak...)

I finally moved to a bit-level stream encoding to save some more bandwidth and give me some more representational flexibility, and this got me thinking about the characteristics of the data that bother us.

In general, the floating point coordinates have significant bits all through the mantissa. Any movement along an angle will more or less randomize the low order bits.

My small little insight was that because missiles are evaluated parameterically instead of itteretively in Q3, a one-time snapping of the coordinates can be performed at their generation time, giving them fixed values with less significant bits for their lifetime without any effort outside their spawning function. It also works for doors and plats, which are also parametrically represented now. Most events will also have integral coordinates.

The float encoder can check for an integral value in a certain range and send that as a smaller number of bits, say 13 or so. If the value isn't integral, it will be transmitted as a full 32 bit float.

The other thing I am investigating is sub-byte delta encoding of floating point values. Even with arbitrary precision movement deltas, the sign and exponent bits change with very low frequency except when you are very near the origin. At the minimum, I should be able to cut the standard player coordinate delta reps to three bytes from four.

So, the bottom line is that the bandwidth won't move much (it might even go down if I cut the integral bits below 15), the maps become unbounded in size to the point of single precision roundoff, and the client doesn't have to care about position jittering (which was visible in Q3 code that will be released).

Chapter 2

March

2.1 Mar 03, 1999

On the issue of railgun firing rates – we played with it for a while at the slower speed, but it has been put back to exactly Q2's rate of fire.

I do agree with Thresh that the way we had it initially (faster than Q2, but with the same damage) made it an overpowered weapon in the hands of highly skilled players, which is exactly what we should try to avoid.

An ideal game should give scores as close to directly proportional to the players reletive skills as possible. The better player should win in almost all cases, but the game will be more entertaining if the inferior players are not completely dominated.

Quake 1 had really bad characteristics that way – Thresh can play extremely talented players and often prevent them from scoring a single point. We wouldn't put up with a conventional sport that commonly game scores of 20 to 1 in championship matches, and I don't think we should encourage it in our games.

Eliminating health items is probably the clearest way to prevent blowout games, but that has never been popular. Still, we should try to avoid weapon decisions that allow the hyper-skilled to pull even farther away from the rest of the crowd. They will still win, no matter what the weapons are, just not by as wide a margin.

2.2 First impressions of the SGI visual workstation 320 (Mar 17, 1999)

I placed an order for a loaded system ($11k) from their web site two months ago. It still hasn't arrived (bad impression), but SGI did bring a loaner system by for us to work with.

The system tower is better than standard pc fare, but I still think Apple's new G3 has the best designed computer case.

The wide aspect LCD panel is very nice. A while ago I had been using a dual monitor LCD setup on a previous intergraph, but the analog syncing LCD screens had some fringing problems, and the gamma range was fairly different from a CRT. The SGI display is perfectly crisp (digital interface), and has great color. The lighting is a little bit uneven top to bottom on this unit – I am interested to see how the next unit looks for comparison.

Unfortunately, the card that they bundle with the LCD if you buy it separately is NOT a good 3D accelerator, so if you care about 3D and want this LCD screen, you need to buy an sgi visual workstation. Several of the next generation consumer cards are going to support digital flat panel outputs, so hopefully soon you will be able to run a TNT2 or something out to one of these.

The super memory system does not appear to have provided ANY benefit to the CPU. My memory benchmarking tests showed it running about the same as a standard intel design.

Our first graphics testing looked very grim – Quake3 didn't draw the world at all. I spent a while trying to coax some output by disabling various things, but to no avail. We reported it to SGI, and they got us a fix the next day. Some bug with depthRange(). Even with the fix, 16 bit rendering doesn't seem to work. I expect they will address this.

Other than that, there haven't been any driver anomolies, and both the game and editor run flawlessly.

For single pass, top quality rendering (32 bit framebuffer, 32 bit depth buffer, 32 bit trilinear textures, high res screen), the SGI has a higher fill rate than any other card we have ever tested on a pc, but not by too wide of a margin.

If your application can take advantage of multitexture, a TNT or rage128 will deliver slightly greater fill performance. It is likely that the next speed bump of both chips will be just plain faster than the SGI on all fill modes.

A serious flaw is that the LCD display can't support ANY other resolutions except the native 1600*1024. The game chunks along at ten to fifteen fps at that resolution (but it looks cool!). They desperately need to support a pixel doubled 800*512 mode to make any kind of full screen work possible. I expect they will address this.

Vsync disable is implemented wrong. Disabling sync causes it to continue rendering, but the flip still doesn't happen until the next frame. This gives repeatable (and faster) benchmark numbers, but with a flashing screen that is unusable. The right way is to just cause the flip to happen on the next scan line, like several consumer cards do, or blit. It gives tearing artifacts, but it is still completely usable, and avoids temporal nyquist issues between 30 and 60 hz. I expect they will address this.

Total throughput for games is only fair, about like an intergraph. Any of the fast consumer cards will run a quake engine game faster than the sgi in its current form. I'm sure some effort will be made to improve this, but I doubt it will be a serious focus, unless some SGI engineers develop unhealthy quake addictions. :-)

The unified memory system allows nearly a gig of textures, and individual textures can by up to 4k by 4k. AGP texturing provides some of this benefit for consumer cards, but not to the same degree or level of performance.

The video stream support looks good, but I haven't tried using it yet.

Very high interpolater accuracy. All the consumer cards start to break up a bit with high magnification, weird aspects, or long edges. The professional cards (intergraph, glint, E&S, SGI) still do a better job.

SGI exports quite a few more useful OpenGL extensions than intergraph does, but multisample antialiasing (as claimed in their literature) doesn't seem to be one of them.

Overall, it looks pretty good, and I am probably going to move over to using the SGI workstation full time when my real system arrives.

I was very happy with the previous two generations of intergraph workstations, but this last batch (GT1) has been a bunch of lemons, and the wildcat graphics has been delayed too long. The current realizm-II systems just don't have enough fill rate for high end development.

For developers that don't have tons of money, the decision right now is an absolute no-brainer – buy a TNT and stick it in a cheap system. Its a better ”professional” 3D accelerator than you could buy at any price not too long ago.

Chapter 3

April

3.1 We are finally closing in on the first release of Q3test. (Apr 24, 1999)

As you have probably heard by now, the first release in going to be the mac version, probably followed by the linux version, and only then the windows version.

Some of you are busy getting all bent out of shape about this.

We want to get a lot of people playing with the test to find bugs and offer well thought out suggestions, but there are classes of bugs and suggestions that emerge in each order of magnitude of exposed testers.

If a given bug is going to show up when a thousand people have looked at it, but we had released it to a hundred thousand people, then we are going to have a lot of duplication to wade through.

The mac testers will find some obvious problems. We will fix them. The later releases will be better.

Even if we had the windows distribution ready to go right now, I would still seriously consider releasing one of the mac or linux versions first because it will make our tracking a lot easier.

The holdup on the windows side are the issues with updated driver distribution. The game itself doesn't have any holdups.

We could do a windows release for just, say, voodoo2 owners and get some of the benefit of a controlled release, but it wouldn't really work out, because everyone would figure out that it can be made to (almost) work on lots of other cards by tweaking some values. That type of feedback would not be useful, because we KNOW that there are problems with most of the current drivers. We have been working with all of the vendors for the past year to get them all straightened out.

Glsetup is going to be slick – just run it and it will Do The Right Thing for your video configuration.

We hope it will be done soon, but there are factors out of our direct control involved.

Don't be spiteful. This is just the beginning of the testing and release process.

One conspiracy theory suggests that Apple is somehow getting us to do this.

What we have "gotten" from Apple is a few development machines. No cash payoff. No bundling deal. No marketing contract.

I am looking at this long term. I want to see OS X become a top notch platform for graphics development. I think highly of the NEXTSTEP heritage and I might move my development from NT if it turns out well. There is a lot of groundwork that needs to be laid with apple for this to happen, and my working on the mac right now is part of that. Plus a lot of complaining to various apple engineers and executives. :-)

To be clear:

At this time, there is no mac that is as fast for gaming (or just about anything, actually) as a pentium III with a top of the line 3D card. Period. I have been misquoted by some mac evangelists as saying otherwise.

The new (blue and white) G3 systems are very good systems in many ways, and make a perfectly good gaming platform. However, a high end wintel machine just has more horsepower on both the CPU and the 3D card.

A 400 mhz G3 performs about the same as a 400 mhz PII if they aren't fill rate limited, where the faster cards on the PC will give about a 25% advantage. A 500 mhz PIII with an appropriate card in 30% faster than the best mac you can buy.

The multi colored iMacs, old G3 desktops, and powerbooks can play Quake3, but the RagePro 3D acceleration defines the absolute bottom end of our supported platforms. A serious gamer will not be satisfied with it.

Voodoo cards are not currently supported by the OpenGL driver, which is very unfortunate because many serious mac gamers own voodoo cards. I hope Apple and 3dfx will be able to correct this soon, but I certainly understand Apple's prioritization – obviously, good support for the OEM video options is of primary importance.

The voodoo performance will still lag the windows platform by some amount, but some strides have been made in that area recently, so I expect good performance,

Gaming is not a reason to buy a mac, but Apple is taking steps so that it may not be a reason to avoid a mac if you have other reasons for wanting one.

MacOS still sucks.

3.2 Apr 25, 1999

Some people seem to think that I just make up these performance comparison numbers. I don't. I measure things, and I understand control conditions.

In this discussion, assume "wintel" is a 500 mhz PIII with either a agp rage128, or an agp TNT card, and "macos" is a 400 mhz G3 with the pci rage128.

At the highest level, you can make application class comparisons between platforms. For instance, CodeWarrior on the mac compiles faster than VC++ on wintel, but stuffit is way slower than winzip. This is useful data, but says more about the application design than the relative merits of the platforms. CW uses a single object file repository, for instance.

A better comparison is an identical app on both platforms.

Photoshop is often faster on macos than wintel. There is certainly a lot of common code, but individual filters are optimized for each platform. Some of these hand optimized operations are significantly faster on the mac.

Quake1 was the counterpoint to that. Quake1 had significant amounts of hand tuned asm code for intel, and the PPC version never got as much attention. The PPC version was noticeably slower (you would have to time at 640*480 to avoid unfairly penalizing the mac for lack of low res modes).

So, clearly, hand tuned asm code can make either platform pull ahead. It also shows that the two platforms are at least fairly close in performance. I never said macs were SLOW, just not quite as fast as the best intel systems.

Quake3 doesn't software rasterize, so there isn't any great place for lots of asm code (the great place is in the OpenGL driver). The code is essentially identical on all platforms.

Q3 is definitely faster on a wintel system than a macos system. When the wintel version is released, everyone will be independantly repeating that measurement.

Even this measurement isn't exactly an apples to apples comparison, because the OpenGL driver and 3D card are still a significant variance. The two can be broken out farther: Q3 can be run without 3D output to test just the identical compiled code. Wintel is still faster, although somewhat less so. The OpenGL + 3D card setup can be benchmarked separately on the axis of throughput and fill rate, which show the intel system being significantly faster. I can't break that apart into the two separate components, but I will guess that the OpenGL driver is probably as efficient as the wintel drivers and the performance delta is due to the system interface and the video card. The current mac rage128 cards run at 75 mhz, which is a third slower than the PC cards. AGP is also more than just a faster PCI, it can influence the structure of communication with the card.

It has been my observation in the past that most of my code tracks just about midway between specint and specfp for performance comparisons. There is a lot of floating point, but it is all short vectors, rather than the massive vectors of scientific computing. If we discount the graphics subsystem, Q3 follows this reasonably well. The intel system does slightly better than projected.

"Sucks" is a subjective description that can be dismissed as opinion. Note that I have NEVER said that the hardware sucks, or the user interface sucks, just that the mac OPERATING SYSTEM sucks.

"Faster", when qualified with testing conditions, is objective, and all the wishing in the world doesn't change it.

Objectivity and quantification are the paths to improvement.

I will be very happy if Apple can produce a desktop system that is faster than anything else you can get. I respect good engineering from any source. Altivec should be better than the PIII extensions (trinary ops – yeah!). The upcoming system architectures look good. They have a shot at it, but they won't make it if they complacently think "oh, we are already faster than any pc system".

My twin turbo F50 can still be outrun at the dragstrip by much cheaper race cars. Many ferrari owners would not dare set foot at a drag strip, because they fear objective measurements that may not show their important possession in the best light. I would rather have the facts, so I can base future decisions on logical grounds.

3.3 Interpreting the lagometer (the graph in the lower right corner) (Apr 26, 1999)

The upper graph (blue/yellow) slides one pixel for every rendered frame. Blue lines below the baseline mean that the frame is interpolating between two valid snapshots. Yellow lines above the baseline mean the frame is extrapolating beyond the latest valid time. The length of the line is proportional to the time.

The lower graph (green/yellow/red) slides one pixel for every received snapshot. By default, snapshots come 20 times a second, so if you are running > 20 fps, the top graph will move faster, and vice versa. A red bar means the snapshot was dropped by the network. Green and yellow bars are properly received snapshots, with the height of the bar proportional to the ping. A yellow bar indicates that the previous snapshot was intentionally supressed to stay under the rate limit.

The upper graph indicates the consistancy of your connection. Ideally, you should always have blue bars of only a pixel or two in height. If you are commonly getting big triangles of yellow on the graph, your connection is inconsistant.

In a heavy firefight, it is normal for modem players to see yellow bars in the bottom graph, which should return to green when the action quiets down. If you are getting several red bars visible, you may want to look for a server that drops less packets.

There are a few tuning variables for people trying to optimize their connection:

The most important one is ”rate”, which is what the connection speed option in the menu sets.

We are fairly conservative with the values we set for the given modem speeds: 2500 for 28.8, 3000 for 33, and 3500 for 56k.

You may actually be connecting faster than that, and modem compression may be buying you something, so you might get a better play experience by increasing the values slightly.

If you connect at 50000 bps, try a rate of 5000, etc.

I err on the conservative side, because too low of a rate will only make the movement of other things in the world choppy, while too high of a rate can cause huge amounts of lag.

Note that the optimal rate will be somewhat lower than a rate for QW or

Q2, because I now include the UDP packet header length in the bandwidth estimate.

You can ask for a different number of snapshots by changing the ”snaps” variable, but there isn’t a lot of benefit to that. Dedicated servers run at 40hz, so stick to divisors of that: 40, 20 (default), 10. A snaps of 40 will usually just cause you to hit your rate limit a lot faster. It may be usefull for tuning rate, if nothing else.

You can adjust the local timing point with ”cg_timenudge ”, which effectively adds local lag to try to make sure you interpolate instead of extrapolate. If you really want to play on a server that is dropping a ton of packets, a timenudge of 100 or so might make the game smoother.

—–

One more addition to net cvars:
”cl_maxpackets” will restrict the maximum number of outgoing packets to prevent client to server rate problems. This does not limit the client framerate. This defaults to 20, which might actually be a bit low. You might try experimenting with raising this to 40.

”cl_maxfps” still exists, but it will never need to be used for networking reasons.

—–-
* converted cvar allocation to indexes to allow range checking
* cgame converted over to use vmCvar_t instead of cvar_t needed for interpreted cgame
* fixed server crashing string bug
* adjusted scoreboard for 8 players
* show hostname on connection screen
* fixed null model warning on startup
* more space for hostname on local servers screen
* fixed mac Open Transport memory buffer bug, this was causing most of the mac crashes
* made Info_ValueForKey() case insensitive
* sv_privateClients, sv_privatePassword. this allows you to reserve slots on a public server for password access while allowing most to be freely available
* ”server is full” message on connect screen

3.3. INTERPRETING THE LAGOMETER (THE GRAPH IN THE LOWER RIGHT CORNER) (APR 26, 1999)

* archive handicap in config file
* cheat protect r_nocurves
* byte order independent zip checksum
* removed cl_stereo, use glConfig.stereoEnabled

3.4 Apr 27, 1999

* cgame converted to use local buffer based lerpTag for interpretation
* cgame converted to use local buffer based argv for interpretation
* new sound code to remove latency
* added drop shadow to field characters and fixed scrolling
* fixed edge-of-bounce-pad misprediction error (server side)
* remove broken weapon-stay dmflag
* made menu gfx never picmip
* cheat protect r_lightmap
* clear sound buffer before any file IO
* use GetOSEvent instead of WaitNextEvent on mac when fullscreen. removes hitches caused by other tasks and gives a performance boost
* continuous scoreboard / ping update when tab is down
* put version number on menu background
* fixed toggle cvar bug
* dim out behind floating menus

3.5 Apr 28, 1999

* converted sound positioning away from callback method
* increased mac memory zone by 5 megs
* new memory allocator for temporary render use during init
* converted cmodel references to handles and range checked
* converted sound references to handles and range checked
* converted file references to handles and range checked

3.6 Apr 29, 1999

* rework versioning for architecture tracking
* use a seperate endpoint for address resolves on mac
* hide OTLook warnings if "developer" isn't set
* defered mac renderer scanning until after mode set so 8 bit desktops don't confuse it
* global motd/update server
* fixed view model animations on models with custom anims

technical note:

Q3 can run networked player movement in either an asynchronous or synchronous manner. The default mode is to allow all client movement to be done asynchronously with the servers advancement of time.

The primary reason is to allow player movement to be predicted on the client side. The primary drawback is that while your movement is smooth, the other players that you see running around in the world move with a jerkiness that is relative to their framerate and network connection quality. It is NOT necessarily relative to their ping - a player on a fast system with a clean modem connection can move smoothly. If you see a player stuttering around, either they have a bad framerate, or the network connection between them and the server or you and the server is poor. The amount of stuttering is sort of the sum of the dropped or variable packets on BOTH connections.

You can force Q3 to run all clients synchronously by setting "g_synchronousClients 1" on the server. This will make Q3 behave similar to Q1 for networking. All movement will be lagged except view angles, which are still short-circuited immediately.

Some people claim to prefer synchronous movement when everyone had a very good ping, but I don't personally think it is ever a play benefit. It makes leading players a bit easier, but I think the crisp movement control of client side prediction is a much better tradeoff.

However, there is still a reason for using it: recorded demos come out a LOT smoother looking when running with sync. Note that q3test does not allow demo recording and playback, so this is just for future reference...

3.7 Apr 30, 1999

I put together a document on optimizing OpenGL drivers for Q3 that should be helpfull to the various linux 3D teams.

http://www.quake3arena.com/news/glopt.html

——-
* vmtest framework, q3asm work
* converted scene building to procedural style. allows better error checking, better performance characteristics when interpreted, and is a setup stage for SMP optimizations in the renderer if I get around to it
* protected some potential div by 0 areas in cgame

Chapter 4

May

4.1 May 04, 1999

* seeded random numbers differently on tourney restarts
* fixed events on initial snapshots
* removed g_maxentities configuration, set by G_ENTITY_BITS
* cl_motd 0 to allow never sending request packets
* fixed map cache clearing bug
* cg_drawFPS 1 for running fps counter in corner
* remove all teleport destination pads
* moved checkmap out of cgame
* moved time positioning out of cgame
* made usercmd overrun freeze in place instead of snapping back
* slightly increased shotgun spread
* protected against using a cleared clientinfo
* use snapped origin from players for linking to prevent slight prediction
errors during player collisions

4.2 May 05, 1999

* client side predict item pickups. running over items was one of the few remaining locally perceived signs of lag
* new pont-in-patch test code
* fixed pathname errors when mac users had slashes in their paths: ”B/W mac”. sigh.

4.3 May 07, 1999

* changed grabbed items to SVF_NOCLIENT instead of EF_NODRAW now that the pickup event is on the player
* clear event flags with event on reset
* move playerstate to netfield bit communication
* fixed configstring delta sequencing issue after initial gamestate
* extended the netgraph: short red lines are missing client to server packets (need to drop 3 in a row)
* extended cg_debugevents
* increased cl_maxpackets to 30
* fixed bug with console field not getting drawwidth set
* fastsky implies noportals
* changed fastsky color
* q3map now fixes tjunctions at fog boundaries
* build optimized tree with visible hulls of sides
* adjusted plane culling to avoid some cracks
* r_facePlaneCull
* fixed too-lax colinear collapse to avoid some cracks

4.4 May 08, 1999

There is one must-fix issue and a couple smaller issues remaining before the release candidate build, then we have to do a lot of testing on it. I made a lot of significant changes in the last week, and I’m sure there are some things we still need to sort out before we inflict it on the general public.

We are aiming for sunday, but understand that that means sunday evening, not sunday morning.

If saturday night / sunday morning testing on the release candidate turns up significant problems, we will put off the release until they are fixed. That could be later sunday night, or it might not make it until monday night.

The previous release delays for win32 were issues out of our control, but this release rests squarely on me. The content and other issues are ready, but we still need to make sure all the new code is solid.

* fixed give item bug
* new first snapshot timing
* moved sun drawing outside of sky shader to fix showtris
* r_drawSun
* handle all shader tesselations in q3map with tjunc fixups
* different flatness epsilons for edge vs border grids
* reorganize sound directories
* removed footsteps on non-metalic and non-water surfaces
* fixed bug with multiple looping sounds
* client side predict teleporters, go to "hyperspace"
* precache remaining liquid sounds
* don't fire jumppad events if upward velocity

4.5 May 09, 1999

We would up making tweaks to both maps today, so the data didn't reach final form until a few hours ago.

I just finished making release candidates for all three architectures, but I already found a couple problems that need to be fixed.

If everything goes perfectly (ha), and I nail these problems immediately when I wake up, then we might make it out tonight, but it is looking a bit doubtful.

There are a few known issues that I decided NOT to hold the test up for:

The gauntlet is functioning correctly, but the visuals are wrong. The designed behavior is that when you hold down attack it will scan for a target and only punch forward when it hits. The visuals currently show it punching constantly.

Dynamic lighting is currently taking a really excessive amount of cpu time. If you are having performance problems in firefights, you may want to turn it off. The option is in the preferences, or you can just issue "r_dynamicLighting 0" at the console.

The powerup item sounds aren't global across the entire world since I went to the client side predicted items.

There are some cases when a weapon that was picked up with a predicted item and immediately fired doesn't make a muzzle flash.

* fixed fs_copyfiles after ospath split
* fixed look-at-killer
* changed railgun impact to plasma dish
* convert connect packet to infostring
* put footsteps back in...
* r_drawsun 0 by default to avoid probs for now
* fixed event clear overwrite problem
* client side predict weapon switch on item pickup
* changed sound fallbacks to "visor" from "male"
* made turbulent texcoords based off of xyz instead of st

4.6 May 10, 1999

A good day of work. I just finished a long test game with all three architectures, and everything looks solid.

As far as I can tell, these are ready to go after making installers and such, but everyone else has an oportunity to find bugs while I sleep...

We won't hold up for minor gameplay issues, but if anyone turns up a repeatable crasher I will rebuild everything.

Barring problems, we should start rolling the releases out tonight.

* fixed crash case on fallback from an unsupported fullscreen
* fixed overrun with very fast system connecting to a very lagged server
* fixed bad Z_Free on sounds not found
* fixed autoswitch with sync clients
* fixed losing console field on positive histories
* fixed demo recording and playback with new net code
* handle signed bit fields in msg code
* fixed playerstate event bit loss on encoding
* fixed "bad clientnum on player entity"
* reenabled corpses sinking into ground

4.7 May 11, 1999

I am offering a bounty for server crashing bugs. Q2 had several releases forced out because of malicious attacks on all the public servers, so I want to try and flush out what I can during Q3's testing phase.

There is a server running in the debugger here at crashtest.idsoftware.com (192.246.40.68). Anyone that can repeatably hang or crash this system can have a $100 prize and some misc bit of Q3A paraphenalia that I can dig up.

Operating system level attacks don't count – only things that I can actually fix or protect against in my code.

Denial of service attacks don't count if they require upkeep, but if there is a fire-and-forget DOS attack, it will still count.

Any actions you can perform with the released client are fair game. Crashing the client isn't good for a bounty, but I would still like to know about it.

Custom attack programs are also fair game. These are actually what I am most concerned about – malicious programs that goes through and crash all listed servers.

Ideally, you would practice on a private server under your control and only hit crashtest when you think you can repeat it.

If you find one, email me the instructions so I can reproduce it. Include ”CRASHTEST” in the subject so I won’t miss it.

First come, first served, one bounty per bug. I will update crashtest with our internal builds, so it will certainly be possible that an attack on the released servers no longer functions on crashtest.

————-

Now that the first win32 test is out, here is The Plan for going forward:

All future releases should be same-day for all architectures.

There may be an exe-only update to the current distributions if there are significant problems, but it isn’t scheduled.

The next major test release will include a new one on one map designed for tournement play, and new executables with server and game modifications, but will not require downloading a new pak0.pk3.

The release after that will introduce various teamplay rules on the original two maps. This version will likely be another full download, because I know that I still have a couple things to change in the map format. This will probably be the first test running with the virtual machine.

The final major test release will introduce the single player game with bots and ranks.

After any bugs are shaken out of that, it will be the ”Q3 Demo” instead of the ”Q3 Test”, and we should be ready to release the full game to stores.

In an ideal world, people that aren’t prepared to deal with in-development software would wait until then to form an opinion of the product.

——-

Everyone should realize that many popular net links are going to be clogged up with q3test downloads for a while, so net play may be a bit patchy to a lot of servers.

——-

BigImp wins the first prize. It doesn't crash the server, but fmtspec names will crash all clients that try to log on. Technically that would be an upkeep required DOS attack, but I'll let this one go.

I even had a "FIXME: make vsprintf safe" comment by the offending line...

I am going to update the server to filter out all % chars that come in over the net to prevent any other similar things.

——

Sami Tammilehto wins the second prize. Some large connectionless packets can cause crashes.

This one was a result of me having the maximum token size defined lower than the maximum string size.

——-

Do NOT send bug reports and game comments directly to me! If I have to filter through hundreds of emails a day, I won't get any more work done... Only crashtest related problems should come to me, everything else should go to q3feedback@idsoftware.com.

——-

You can bias the level of detail lower than allowed in the menu with "r_lodbias 2", which will force all models to the lowest lod. The view weapon will look very ugly.

Another little speedup option that isn't offered in the menus is: "cg_simpleitems 1" this removes the extra rings and spheres around some items.

You can also turn off all the gibs with "cg_gibs 0".

* clear game memory at init, which fixes the stuck-at-intermission problem on mac servers
* fixed mismatched free / Z_Free in demo menu

* removed unused reference to sprites/plama.md3
* automatically get sounds from model name
* scale sensitivity by zoom
* immediately archive changes to latched cvars
* cheat protect r_portalonly
* don't print "XXX connected" on level restarts
* fixed "give item" on levels where 0,0,0 is in solid
* fixed timedemo
* don't play pain falling sound if dead
* fixed falling damage sound not snd specific
* fixed crashtest 2
* fixed crashtest 1
* q3map_backshader
* q3map_globaltexture

4.8 May 12, 1999

We had to upgrade the crashtest machine to NT sp 5, because some people were attacking it with windows crashers. Those don't count.

Crashtest #3 from [iBO]QWhAX0R was a combination of two problems:

The symptom was disconnecting all clients with an illegible server message. This turned out to be caused by the fact that I was parsing strings out of my net buffers with a MSG_ReadChar() function, and I was checking for EOF as a negative one return value. I had to change this to a MSG_ReadByte() call, because -1 was showing up in the messages, which then caused a parse error because it wasn't really the end of the message.

The actual root of that issue was code like this:

```
{
        char buffer[MAX_STRING_CHARS];
        ...
        strncpy( buffer, input, sizeof(buffer) - 1 );
        ...
}
```

No buffer overruns are possible, but buffer is not forced to be zero terminated if on the stack. I'm pretty sure this was a result of copy-and-paste code where buffer used to be a static with a guaranteed zero, but it made me find several other places where similar things were happening.

I had started using a Q_strncpyz() function a while ago that guarantees a trailing zero and doesn't require the -1, but it turned out that between code I had written a long time ago, and code that either Cash or Brian had added, there were still a lot of normal strncpy calls around. A lot of them were wrong, too. Either missing the -1, or missing the dedicated 0 fill in.

Crashtest #4 from Jim Paris was a variation on the first part of #3.

Only one of these attacks so far has been a server crasher, but I have been giving the bounty for anything that immediately kicks all the players. I probably won't give it for attacks that only overflow some lagged clients, but I'll evaluate as they happen.

I am off to E3 now for a bunch of PR silliness, so if crashtest goes down, it won't be back up for a while...

* fixed crashtest 4
* fixed crashtest 3
* fixed jumping-over-item pickup prediction error
* made "Couldn't load sound" a developer only warning
* fixed demo recording not including portal surface entities
* precache grenade bounce sounds

4.9 Now that all the E3 stuff is done with, I can get back to work... (May 19, 1999)

I was stuck in a room the entire time doing press interviews, but it seemed to have gone well. It was mentioned to me that there were a few people on the show floor with forged badges that read ”John Carmack – Id Software”. As if forged email / irc / icq isn’t enough of a problem. Sigh.

The ”download” crashtest was first reported by Rick Hammerstone. That was a pure dumbass mistake on my part.

I should be sending the accumulated crashtest bounties off tomorrow.

The plan right now is to have an update release next week that will have lots of bug fixes and cheat protections, but not too many new user visible features.

I finally got around to implementing dual processor acceleration today. I still have a couple issues to resolve and some more rearranging to do, but it is giving 20%+ speedup right now in a worst-case situation for it.

When completed, I expect the average speedup to be in the 40% to 80% range, depending on what is going on and the video configuration. Scenes with lots of dynamic lighting and lots of sounds and other client processing going will show the largest speedups. It helps the slow scenes more than the fast scenes, which is basically what you want.

I am going to shake this out with the Windows (NT) code first, but it should definately make its way to the linux port eventually.

I know SMP is a que for all the BeOS folks to ask about ports, so I’m going to head that off: Be has all the code for Q3 (and Q2, for that matter), and a version of Q3test should be available by the time they ship a release OS with OpenGL hardware acceleration.

There will probably also be an SGI-irix port.

Regarding both of those ports: they are not supported ports, and will be maintained by volenteers (like the current MacOS X port). Update releases will lag the official ones, and we won’t committ to ANY dates.

I am doing all of my development on intergraph and sgi-NT hardware, but when I have everything rock solid, I will give Nvidia and ATI’s NT drivers a try. I would be somewhat shocked if they didn’t explode – I doubt multiple threads playing occasional tag team on a context has been well tested.

4.9. NOW THAT ALL THE E3 STUFF IS DONE WITH, I CAN GET BACK TO WORK... (MAY 19, 1999)

True, only a tiny fraction of our players (probably less than 1%) will be able to take advantage of this, but I consider SMP usage to be an important technology to nurture over the coming years.

The top of the benchmark chart should be an SMP system (assuming the NT drivers have all the optimizations of the '98 drivers), and it will also be possible to build a reletively cheap SMP system (say, dual 400's) that outperforms the best single processor system.

4.10 May 22, 1999

The SMP support is solid enough to play with now. The only feature that is still broken is light flares.

As a happy consequence, some of the cleanup work I did for SMP gave a couple percent speedup even when running without the separate thread.

On my development system, a dual 300 mhz intergraph realizm II, the low res timedemo scores went from 27.8 to 37.8 with "r_smp 1". This is only a 35% average speedup, but at some times (lots of dynamic lights in complex scenes) the speedup is 90%+. Gameplay is noticably smoother.

The rendering thread is almost always the blocking factor, so the faster the card and OpenGL driver, the larger the speedup will be.

This is explicitly a two thread producer / consumer, so there is no benefit to more than two processors. The app is well behaved, using sleeping syncronization so that you usually still have half a processor free for other operating system functions.

Hopefully we will be able to test with some fast consumer cards sometime soon.

——

A lot of people asked what was done differently this time vs the last time I tried (without benefit) to use SMP.

My original attempt was to make a DLL that intercepted all OpenGL calls and let a separate processor execute them. The benefit would have been that all OpenGL applications could have gone faster. The problem was that the bandwidth required to encode all the commands was enough that the processor overhead was as much as it would have taken to just do the geometry on the main processor.

It would have still been a win if the geometry side was doing lots of work, like multiple lights, user clip planes, and texgens, but for the vast majority of geometry, it didn't balance out. If someone wanted to try that using the PIII or AltiVec streaming memory operations, it could probably still work.

The current SMP code is implemented directly into the renderer, and a lot of things were moved around and double buffered to allow it to use data in place, instead of having to copy it off.

——

Some people expressed surprise that Quake3 wasn't threaded already.

Threading has been presented so often as the "high tech" "cool" way to program, that many people aren't aware of the downsides.

A multi-threaded program will always have somewhat lower throughput when running on a single CPU than a single threaded program that polls in explicit places. The cost of a context switch at the cpu level is negligible, but the damage that it can do to the cache hierarchy can add up to a noticeable amount in bad cases.

The much larger problem is that you lose tight control over when things occur. If the framerates are low enough, it isn't a huge issue, but for applications running at 30+ fps, you really don't want to trust the OS scheduler to coordinate multiple threads and have them all get in every frame. Yes, with explicit sleep() calls you can sort of get it working, but at that point, you might as well not be using threads.

A good example of not-quite-in-sync issues in the windows mouse performance. A PS/2 mouse only samples 40 times a second, so when you get an app updating at around that speed, you will get 0/1/2 scheduling variances.

They are also not terribly portable, and a pain in the ass to debug.

4.11 May 26, 1999

* basic joystick controls, some work still needed for advanced controlers
* r_dlightBacksides 0 option
* forced cvar_restart when version changes
* fixed some flare-in-fog problems
* fixed skin color in menus
* print obituary message even when you are the killer, so all kills get an entry in the logfile
* fixed bugs in line token parsing when quotes or commands aren't white space separated
* multiprocessor acceleration "r_smp 1"
* increase menu dimming
* increased rocket damage radius from 120 to 150 units
* check for running server in all server commands (dumpuser, etc)
* new cvar cheat setup – by default, only archived variables can be changed when not cheating
* "cg_drawstatus 0" only removes status bar
* "cg_draw2d 0" removes all 2d

4.12 May 27, 1999

* enable scissor test properly
* archive r_lodBias
* cg_draw3dIcons 0 option
* data cheating protection
* userinfo renamed to clientinfo, added state and current server address
* don't forward commands to a server when playing demos
* fixed NULL extension on dir command
* added one more shotgun pellet
* added CG_Shutdown for cgame cleanup
* fixed jitter in rising smoke
* increase minimum time before reusing an entity slot
* soundinfo reports current background streaming file
* changed IPX separator to . from :, moved port processing to system

independant code
* auto port scan wasn't updating the net_port cvar
* attack button presses reset inactivity timer now
* increased the forced respawn time from 10 to 20 seconds
* show smp on gfxinfo, slight reformat

4.13 May 30, 1999

For the past couple of weeks, I have been spending some development time on linux, and for the first time on a non-NEXTSTEP unix platform, I have actually been enjoying it.

While Id has been supporting linux since the Doom days, I have not personally been much of a linux user – it was always ddt or zoid doing the actual coding and testing. Every year or so I would install a linux distribution and play around with it for a few days, but I would always leave feeling that it was still pretty crude (UI wise) compared to the NEXTSTEP UI I was used to, or even what I had used on other commercial unix workstations and windows.

There have always been a ton of reasons to like linux, but the user interface was enough of an issue that I couldn't buy into it completely.

The gnome user environment in Red Hat 6.0 is finally at a level that I consider it a valid alternative to commercial desktop environments. Overall, its still not as smooth, consistant, or complete as windows or the mac, but is does have its strong points, and things seem to be progressing quite rapidly.

Its still not something you would give to a purely casual computer user, but I won't be surprised if even that changes in a couple years.

CodeWarrior for linux is also a significant aspect of my enjoyment. Its a sort of crappy 1.0 port with a lot of little issues, but the editor works well enough, which is the important thing for me. I have never been able to stand vi or emacs for long enough to become proficient in them.

The code that I have been playing with most is the matrox g200 GLX driver.

Matrox is the first of the major 3D chip vendors that has had the guts to publicly release register level documentation for their 3D chips.

An accelerated X windows OpenGL driver has been put together with this by building on top of the existing Mesa and GLX projects.

It actually runs quake, quake2, and q3test. It doesn't run them FAST, but the quality is good, and I am impressed nonetheless. It is bordering on playable with all quality options set to the minimum on a fast computer, but it still has a ways to go before casual users should take a look at it.

It is steadily improving, and I hope Matrox will be pleased enough with the progress that they will release the documentation for their setup engine to go with the rasterizer.

In testing q3 on it, I noticed that with picmip set to 0, textures would get corrupted and it would never settle on a working set. The current Apple OpenGL drivers also have exactly this problem.

The cool part is that this driver is completely open source. I downloaded the project code, browsed through it a bit, and changed two lines of code to fix the bug. That RULES.

The next thing is sort of funny. I had been suspecting that a lot of the OpenGL drivers were clearing the entire depth buffer for each of the 3D icons on the status bar, instead of limiting it to the scissor region. I added code to the g200 driver to explicitly crop the clear region in the driver, but it never got executed. A little more investigation showed that I had been making an improper assumption for years – scissor is not enabled by default. Doh.

Ever since noticing that glquake cleared the screen borders when the view is sized down, I had been operating under the assumption that intergraph just had a bug in their drivers. I had double checked that glClear was supposed to be limited to the scissor region, so I thought they were just messing it up.

Now I know that I was just being an idiot about that for the last three years... With scissor enabled, most of the cards got a few percent faster.

4.13. MAY 30, 1999

* dynamic curve level of detail. r_subdivisions determines the maximum level of detail, r_lodCurveError determines how quickly polygons are pulled out with distance
* devmap sets cheats 1, map sets cheats 0
* change weapon item upscale to 1.5 instead of 2
* always toss items forward, even if looking up or down
* draw ammo in grey while weapons are reloading
* change railgun shader while reloading
* fixed head models not showing proper skin
* skip all shell eject code when cg_brassTime 0
* fixed sound memory overallocation
* profiling and rearrangement
* fixed dead spectator bug

Chapter 5

June

5.1 Whee! Lots of hate mail from strafe-jupers! (Jun 03, 1999)

Some reasonable messages have convinced me that a single immediate jump after landing may be important to gameplay. I'll experiment with it.

Strafe jumping is an exploitable bug. Just because people have practiced hard to allow themselves to take advantage of it does not justify it's existance. When I tried fixing the code so that it just didn't work, I thought it changed the normal running movement in an unfortunate way.

In the absense of powerups or level features (wind tunnels, jump pads, etc), the game characters are supposed to be badasses with big guns. Arnold Schwartzenegger and Sigourney Weaver don't get down a hallway by hopping like a bunny rabbit.

This is personal preference, but when I play online, I enjoy it more when people are running around dodging, rather than hopping.

My personal preference just counts a lot. :-)

btw, here are the current weapon effects:

gauntlet: 50 pts, 400 msec / punch
machinegun: 10 pts, 100 msec / shot
shotgun: 11 pellets of 10 each, 1000 msec / shot
rocket launcher: 100 pts direct hit, or 100 pts splash damage falling off over 120 world units, 800 msec / shot
plasma gun: 20 pts direct hit or 15 pts splash damage over 15 units, 100 msec / shot
railgun: 100 pts, 1500 msec / shot
lightning gun: 8 pts, 33 msec / trace, max range 768 units
grenade launcher: 100 pts direct hit, or 100 pts splash over 150 units, 800 msec / shot.
bfg: 40 pts instant splash damage over 100 units, 100 msec / shot
flamethrower: to be determined, but short range / wide angle

Splash damage is calculated from the edge of the player's box, unlike quake1, where it was calculated froom the player's origin.

* ignore cl_maxpackets on LAN
* changed cl_packetdup to 1 by default, and archived
* defaulted com_maxfps to 100 and archived, automatically disabling during timedemo. It was possible to lag out some client connections on ultra fast systems even with cl_maxpackets set fairly low due to a huge number of individual commands being created
* 250 msec minimum time between landing and jumping again. I hate having players bouncing around all the time...
* fixed bug with large r_picmip values (white shotgun sight bug)
* new lightning and rail beam drawing
* player torso twitches with pain sounds
* r_drawstrips changed to r_primitives and archived, with changes: default "0" uses glDrawElements if compiled vertex arrays are present, or strips of glArrayElement if not. "1" forces strips, "2" forces drawElements, "-1" skips drawing
* increased rocket speed to 900 from 800. decreased direct hit damage from 120 to 100. splash damage same as 1.05
* removed sound-in-use dialog, auto skip after second try
* made userinfo persistant on server across level changes
* allow different servers to respond to a challenge, allowing redirecting server proxies

5.1. WHEE! LOTS OF HATE MAIL FROM STRAFE-JUPERS! (JUN 03, 1999)

* notice bad ip addresses for connection: 192.1234.123.122
* removed neck length pivot to prevent view poking into low subdivided curves. Also make aiming when looking up or down more precise.

5.2 Jun 27, 1999

For the past couple years when talking to chip makers about new 3D features, 3D texture maps would always come up, and I would have to mutter something like: ”I think it is a good thing, but I can't give you a really awesome example of using it. 3D Noise functions. Stuff like that.”

I have the awesome example now: Lighting.

Through the entire development of Q3, I have been wanting to do dynamic lighting of the world differently. Many of the efficiencies gained by Q3's new map format result in larger and larger lightmaps while it reduces polygon count. That meant that even small dynamic lights may force a lot of work to generate new lighting textures.

I had been wanting to find a way to trade off additional rendering passes for less CPU time.

This is a perfect example of me outthinking myself.

I knew it would have to be something with projecting a light spot texture onto the world geometry, but I am keenly enough aware of the issues and limitations in projecting a 2D texture onto 3D geometry that my mind kept looking at worst case scenarios – how do you project a continuous texture onto the inside of a dome with the proper intensities, and some situations with curves. I thought it would require some non-trivial per triangle analysis and operations, when I really just wanted something that could be done on a per-vertex level.

Recently, the 3D texture insight hit me.

Create a 3D texture of, say, 32*32*32 texels and fill it with your light attenuation function, bright point in the center, fading off to black at the edge. Set it to clamp mode, so any values beyond the edges stay black. Set up a texture coordinate generation matrix to position the light relative to your models. Its a simple translate and scale. Set up lighting calculation to only generate a value based on the relative angles, ignoring attenuation. Use the resulting modulated texture mapping as your ”lightmap” to modulate another texture pass, or add to a previous lighting pass.

This has several important benefits:

It looks good (and identical) on both small and large polygons. Vertex based lighting needs an appalling amount of tessellation to avoid visibly triangulation artifacts. Even if you tessellate to the same sample density as your lightmap grid (hundreds of thousands per map), vertex lighting still looks worse, because it is a triangular instead of bilinear interpolation.

It can be done completely by upcoming lighting and texgen hardware, but is efficient in software implementations.

It can be shadowed by either stencil volume shadows or shadow buffer tests.

Post Q3A, I plan on doing a research engine that is fully dynamically lit and shadowed instead of using lightmaps.

Ok, but we don’t have 3D textures in any hardware now, so this doesn’t help me for Q3A.

I continued thinking along complex lines, like ”Any planar slice of a spherical 3D texture will be a 2D circle, and the three triangle points can occupy any location on their respective rings”.

It still looked like a big mess.

I got tired of thinking about it, and just started coding in some infrastructure to do extra lighting passes. I just made a gradient circle texture for the lighting, and generated the texture coordinates by just translating and scaling X and Y from the world coordinates.

I got it running, grabbed a rocket launcher, and fired down a hallway. It looked just fine. I was stunned. I had been thinking about complex ramifications of weird edge cases for the past year when all it took was a couple hours of programming and the simplest possible approach to make it work decent. Sigh. :-)

The only real addition required was a fading of the light contribution with Z distance from the surface. The downside to this hack is that while you get a nice ball of light moving over floors and ceilings, you only get a wash of light on walls. In hindsight, I can analyze this in the context of our games and say ”Almost all movement and targeting takes place in the XY plane in a FPS, so accuracy along the Z axis is not necessary.”

The somewhat more obvious change to dynamic lighting that wasn't related to the projection issue is that I have it currently set up as a post-process pass, rather than an additive pass on the lightmap. Adding to the lightmap is more ”right”, but it really complicates the multitexture implementation, and on some surfaces, the lightmap is actually done after the base texture, so it couldn't be added to. The end result is that brighter areas are changed more by dlights than dark areas. I don't feel TOO bad about that, because its not like the dlight is much of a proper lighting simulation even in the best case...

* fixed tourney restart
* fixed jittering on plats
* fixed ref use without a world
* new default image that lets you see mapping coordinates
* fixed reliable sequences on restarts and demos
* allow maxclients to change between levels
* randomize shell ejection start position and angles
* display attackers head after every wound
* added gamma and overbright support to mac version
* removed table from sound mixing
* remove smoke and blood puffs when you run through them
* set cheats to 1 on disconnect
* shader sort value is now floating point
* new trajectory type ”TR_INTERPOLATE”, skip interpoaltion for other types
* fixed door open timing
* fixed bug with > 8 portal areas
* added area print to r_shwocluster
* removed all the partial shader match cruft, fixing a crash-on-load
* fixed bug with personal shadows not being setup properly

* nomipmap shader parm split and expanded:
nopicmip : ignores r_picmip so image will always be high res
nomipmap : forces a single level texture, used for console font
* shrank sound mixing buffer for better caching
* new shader option: q3map_lightsubdivide. larger values make q3map -light proceed faster
* new vertex array interleaving
* fixed bmodels not counting patches in bounds
* fixed patch sphere culling on rotating entities
* cg_simpleItems now draws sprite items
* fixed serverid being 0 when map is started from cmdline
* fixed UI on mode changes
* replaced 1280*960 mode with 1280*1024
* test all rotated orders for tristrips from faces
* fixed RB_SurfacePolychain to not duplicate vertexes
* changed planar face surface type from a convex polygon to a general collection of coplanar triangles
* change renderer to use bmodel surface list instead of tree
* changed areabits pointer to areamask array
* fixed bad loop when client reliable message overflows
* fixed swapinterval after vid_restart
* wall mark clipping moved out of cgame, extended for better wrapping over multiple brushes
* picmip defaults to 1 under all cases now
* automatic curve LOD grouping
* duplicated SCR_ functions into cgame
* implemented Micahel Julier's optimization work
* implimented Michael Gold's SMP patch
* new reliable command transport
* use ENTITYNUM_NONE and ENTITYNUM_WORLD constants, which are now in MAX_GENTITIES range for safe net transport
* wait for attack released after respawning before firing
* added a delay before moving everyone to intermission spot
* fixed bug with regibbing of gibbed body ques
* blood trails behind gibs
* changed localents over to trajectories
* removed world as entity zero, clients are now 0 to MAXCLIENTS-1
* changed game interface functions to use clientnums instead of point-

ers
* removed speculative usercmd_t from user packets
* new item pickup code, fixed silent item pickup on grazing hits, and expanded the pickup range by 20% without changing the physical bounding box
* allow a single ”quick jump” without delay

Chapter 6

July

6.1 AMD K7 cpus are very fast. (Jul 03, 1999)

Some timedemo numbers (a new demo, not comparable to previous scores):

Run at 640*480*16 bit color to emphasise the cpu/driver performance rather than the hardware fill rate.

	K7-600	K7-550	PIII-500
TNT2 ultra 16 bit	73.9	68.5	53.8
Voodoo3 3000 16 bit	70.5	65.2	46.0

This is with K7 optimized drivers vs seperate PIII optimized drivers.

There is still wiggle room there in that it is possible that more effort was expended to make the AMD drivers perform better. That is perfectly valid from a consumer's point of view, but muddies the technical CPU comparison.

On identical code run on the systems, there was some more interesting data:

On my map processing tools, the K7 was faster than the PIII, but only slightly more so than the reletive clock rate increase. I would guess that this is due to larger data sets that don't fit in cache as well.

On the matrox OpenGL drivers, which have not been optimized very much and (to my knowledge) contain no PIII specific code, the K7 was a LOT faster.

The bottom line is that I feel comfortable standing behind the statement that the K7 is faster than the PIII. I will have to wait for some stuff to come out of NDA to provide a more detailed technical analysis.

Architectural cleverness is all well and good, but if AMD can't keep the clock speed up with intel, they will still fall behind. A 700 mhz PIII would probably find a lot of applications (especially integer apps) where it would outperform a 600 mhz K7.

* stabilized cg_showfps
* added append support to module file opening
* automatic logging of game scores to games.log
* fixed guantlet firing action
* force a vid_restart on WM_DISPLAYCHANGE messages
* fixed sticking on stairs on very fast framerates
* fixed sticking on stair when jumping
* fixed sticking in corner while falling physics bug
* fixed slide down steep slope physics bug
* r_showimages texture use debugging tool
* cg_freezeDemo cvar
* cg_drawSnapshot cvar
* fixed warnings after demo playback
* changed "stopdemo" to "stoprecord"
* fixed phantom windows on task bar after exit
* check for unset player animation
* fixed the snap-down-look-up bug with very high sensitivities
* reduce inflicted damage by handicap
* all pmove results as events for proper demo playback

6.2 Jul 24, 1999

I was in San Jose for the past week. The original idea was to go into ”hermit mode” holed up in a hotel room and get a lot of work done without any phone calls, email, or coworkers to distract me, but I wound up with meetings scheduled every day with various valley companies. Next time I want to try that, I'll pick some state like Montana... :-)

The top priority was getting the virtual machine done, but I hoped to also get some more of the map data optimizations completed. I definately underestimated how big of a hole daily meetings would punch in the amount of work I could accomplish.

On wednesday I was sweating a bit, not sure if I would have the VM finished in time, but it all came together in the last two days.

The virtual machine interpreter is now completely functional, and cgame can switch between being loaded as a binary .dll or an interpreted .qvm at the change of a cvar.

The basic setup is that I have a modified version of the lcc compiler that generates assembly files that are processed by a new tool ”q3asm” into a .qvm file that can be interpreted by q3. You can still use normal dll's for debugging, then release interpreted code. You can release a binary dll if you need some native system services (some networking stuff, for example) or are doing very compute intensive work, but I strongly encourage everyone to try to use the virtual machine.

The interpreted code has two prime benefits: portability and security.

User mods done with qvms will automatically work on mac, linux, and any other oddball ports that get released.

A qvm program cannot modify anything outside its private address space, and the ”system calls” to the game strictly limit what can be touched. I was scared about the binary dll's in Q2, but rationalized it to myself that people running public servers should be aware of the dangers. With Q3 allowing client side programming, it just needs to be safe for everyone.

Packaging is also improved somewhat, because the virtual machine programs can be included in pak files and are automatically handled through the search path.

Unfortunately, even after doing most of the straightforward optimizations, the interpreter is causing a 20% slowdown in timedemos right now.

I am pretty committed to running cgame interpreted, but if I don't get a significant speedup, we may have to leave the server side game module as a native dll. The ui module can obviously be interpreted.

There are lots of paths I can take to get the performance up:

Write more efficient cgame code. I will definately be looking at this. A lot of things that just didn't matter at all when in native code now add up enough that they should be fixed. I want to avoid flexibility tradeoffs if possible.

Move more calculation from the cgame to system calls. I have already done this for the obvious things like memset, matrixmultiply, etc. Any future work will involve restructuring cgame code to focus lots of work into simple stateless functions. The trick is to leave all the flexibility in the cgame while moving the work.

Implement LCC compiler optimizations. I doubt I will pursue this, unless there are existing optimization projects based on LCC with minimal backend changes required.

Implement more complex interpreted instructions and peephole combine the bytecodes into the new instructions. I think this has some promise.

Assembly code the interpreter loop. The compiler is generating good code, but there is still room for gain. I don't want to pursue this until the high level optimizations are wrung out.

Load time compilation from bytecode to native code. This would bloat a lot, especially on RISC cpus. I'm not sure that a straightforward implementation would have significant speed benefits over an assembly interpreter loop, and I won't have time for writing an aggressive native optimizer.

Deal with the performance cost and optimize the renderer some more to compensate.

6.3 Jul 29, 1999

* log all client transitions, item pickups, and kills
* changed joystick axis to act just like arrow keys so they can be bound in the controls menu for strafing. Yes, this does remove slow walking from joystick movement, but it makes everything a lot cleaner.
* fix up PantherXL trackball support
* removed bad clear command from dedicated console clear button
* tournement queuing of spectators to enter the game
* track wins and losses as long as you stay on a tourney server
* spectators are now in fly mode instead of noclip, and use teleporters
* pass serverTime instead of msec for command timing, prevents timescale cheating
* track dual eventParms on player state
* draw crosshair and name in spectator mode
* fixed rcon
* r_colorMipLevels texture visualization tool
* don't allow weapon select and cycle when in follow mode
* archive cl_yawspeed and cl_pitchspeed
* don't draw place line on scoreboard when spectating
* fixed console chatting during intermission
* better recursive error handling
* fixed curve surface flags (no impact when landing on a curve bug)

6.4 I apologize for not getting the Mac version released this last time. (Jul 30, 1999)

All conspiracy theories aside, here is what actually happened:

I had my San Jose travel scheduled a while ago, and we were expecting to have the release done before I left. We didn't quite make it, and nobody else at the office knew how to build the mac version after Cash made some last minute changes.

When I got back, Graeme was taking off to move his family down here.

Graeme is in charge of building all the installs for our releases.

I considered just tossing new executables with the latest fixes for everything, but some data has changed, and it just isn't worth the hassle right now.

We will be making proper new releases for all architectures monday night when he gets back.

* fixed marks fading properly in fog volumes
* show weapon in fov > 90, adjusting position down as needed
* allow run/bob variables to be changed in non-cheat games
* update scoreboard info while at intermission
* fixed angles on resetplayerentities, corrects twitch on players as you come through a teleporter or respawn
* print "waiting to play" for tourney spectators
* fixed tied rank with 0 score and spectators
* return to roaming spectator when a followed client quits or spectates
* release windows cursor when running windowed and the console is down

6.4. I APOLOGIZE FOR NOT GETTING THE MAC VERSION RELEASED THIS LAST TIME. (JUL 30, 1999)

Chapter 7

August

7.1 Aug 01, 1999

* changed RF_PORTALSURFACE to an entity type
* add sprite indexes in strip order
* changed builtin models to refentity types
* prevent respawn until resting on ground
* fixed unecessary dlight shader checking
* no splashe when lg hitting sky
* removed cvar_restart on version change, caused problems with dedicated server startup
* fixed stupid error that made skies double render
* fixed stupid error that made skies draw an uneeded bakground pass
* support add-mode multitexture for skies

7.2 Aug 16, 1999

As I mentioned at quakecon, I decided to go ahead and try a dynamic code generator to speed up the game interpreters. I was uneasy about it, but the current performance was far enough off of my targets that I didn't see any other way.

After getting over being sick the start of the week (someone from QC must have brought me a flu present), I decided to dive in to it.

At first, I was surprised at how quickly it was going. The first day, I worked out my system calling conventions and execution environment and implemented enough opcode translations to get ”hello world” executing.

The second day I just plowed through opcode translations, tediously generating a lot of code like this:

```
case OP_NEGI:
    EmitString( "F7 1F" );      // neg dword ptr [edi]
    break;

case OP_ADD:
    EmitString( "8B 07" );     // mov eax, dword ptr [edi]
    EmitString( "01 47 FC" );  // add dword ptr [edi-4],eax
    EmitString( "83 EF 04" );  // sub edi,4
    break;

case OP_SUB:
    EmitString( "8B 07" );     // mov eax, dword ptr [edi]
    EmitString( "29 47 FC" );  // sub dword ptr [edi-4],eax
    EmitString( "83 EF 04" );  // sub edi,4
    break;

case OP_DIVI:
    EmitString( "8B 47 FC" );  // mov eax,dword ptr [edi-4]
    EmitString( "99" );        // cdq
    EmitString( "F7 3F" );     // idiv dword ptr [edi]
    EmitString( "89 47 FC" );  // mov dword ptr [edi-4],eax
    EmitString( "83 EF 04" );  // sub edi,4
    break;
```

(yes, I could save a few bytes in those opcodes by moving the sub edi, but I am trying to leave the subs at the bottom and the adds at the top, in case I want to add a peephole optimizer)

I was writing test programs as I went, so I thought it was still going quite well. I quit for the day with only six opcodes left to write.

Today I got in, wrote the last opcodes, and started running the full cgame module.

The first problem was obvious: the loading screen graphics came up with the default image instead of the text font. I quickly found and fixed a problem with system call return values.

It was then able to get into the game, but the FOV was clamped out to 160, and it crashed when you fired your gun. That turned out to be my failure to fix the operand stack correctly in the rarely used structure copy opcode.

The next problem was a little more distressing. You could run around the level, but all the items had an X coordinate of 0. This took a little while to find, and turned out to be a twitchy case of sometimes getting an extra value on the operand stack.

At that point everything looked like it was working perfectly. I was ecstatic. I started running timedemos to see what the performance looked like. Then I ran a demo without timedemo on, and near the end of the demo it crashed. Then I found out that if you play in a level for a few minutes, it either gets an error of some kind, or crashes.

I was quite unhappy about that. Debugging a non-deterministic crash in generated code. Joy.

What I wound up eventually doing was to make gigantic log files of all system call interactions and compare the compiled code against identical runs with the interpreter. I had to tweak a few things to make the processing exactly the same between them, but it let me find where things first started to diverge. It turns out I was letting the top of my compiled code's local stack creep down a bit with each call. Let it run long enough, and it started hitting important things. If I had logged stack pointers instead of parameter values, I would have found it a whole lot quicker...

Now, I am pretty confident that it is correct.

The generated code is pretty grim if you look at it, in part due to the security measures (mask and add for each load/store), and in part due to the fact that it is a straight bytecode translation:

```
06214DD0 83 C7 04             add         edi,4
06214DD3 C7 07 00 4D 0A 00    mov         dword ptr [edi],0A4D00h
06214DD9 8B 1F                mov         ebx,dword ptr [edi]
06214DDB 81 E3 FF FF 1F 00    and         ebx,1FFFFFh
06214DE1 8B 83 30 00 DC 05    mov         eax,dword ptr [ebx+5DC0030h]
06214DE7 89 07                mov         dword ptr [edi],eax
06214DE9 83 C7 04             add         edi,4
06214DEC C7 07 2C 00 00 00    mov         dword ptr [edi],2Ch
06214DF2 8B 07                mov         eax,dword ptr [edi]
06214DF4 01 47 FC             add         dword ptr [edi-4],eax
06214DF7 83 EF 04             sub         edi,4
06214DFA 8B 1F                mov         ebx,dword ptr [edi]
06214DFC 81 E3 FF FF 1F 00    and         ebx,1FFFFFh
06214E02 8B 83 30 00 DC 05    mov         eax,dword ptr [ebx+5DC0030h]
06214E08 89 07                mov         dword ptr [edi],eax
06214E0A 8B 07                mov         eax,dword ptr [edi]
06214E0C 29 47 FC             sub         dword ptr [edi-4],eax
06214E0F 83 EF 04             sub         edi,4
06214E12 83 C7 04             add         edi,4
06214E15 C7 07 40 00 00 00    mov         dword ptr [edi],40h
```

Code bulk is also up there, at about 5x the bytecode version. There is definately some savings to be had with better opcode selection, but no more than 30% or so at best.

Performance is within my tolerance now:

Q3demo1 dll: 52.9
Compiled: 50.2
Interpreted: 43.9

Q3demo2 dll: 50.1
Compiled: 46.5
Interpreted: 38.7

I will probably work a bit more on performance, but that is the ballpark that it will be in. 5% speed hit in most levels, somewhat more in the big open arenas. Next week I will be getting the other modules set up for running in the virtual machine and see how their performance is.

It is a pretty cool setup - you can have some modules as dlls, some as interpreted bytecodes, and some as compiled bytecodes. We will leave the user interface interpreted to save memory.

Tomorrow I am going to get all the byte order issues worked out for powerPC. The interpreter doesn't even work there yet because of inline constant byte order issues. Fixing that will slow the interpreter a little more, but that shouldn't be any problem, with the performance oriented stuff being compiled.

Doing the PPC compiler will be a bit messier because the tools aren't as nice, and the fact that it will involve a whole lot of Mac crash/reboot cycles before it stabilizes, but I think I know what to watch out for now.

VC6 crashed on me about a dozen times in the last few days, probably due to my having show-code-bytes on in the dissassembly window, but it was still pretty damn useful through the whole process.

I am curious to see how the RISC code bulk turns out. The instructions are going to be longer, but all the constants can be held in registers. Should be interesting.

I don't think I am going to be in any hurry to do MIPS/ALPHA/SPARC code generators. One or two code generators and execution environments is educational, but that will be more than enough for me. If we do port to other architectures, they can still run with the interpreter or binary modules until someone else gets up the inclination to do a code generator.

7.3 Aug 26, 1999

The current plan is that we will have another test release around the middle of next month. This version will be running game/cgame/ui with the virtual machine, and will include single player play against bots. No new maps.

I will be releasing the source for all the VM modules with it, and setting the executable up so that it will allow modified modules to be used.

The modified LCC and q3asm will be available both in source form and precompiled, so a professional development environment will not be requried. Using MSDEV to debug binary dll's does make exploration a lot easier than adding prints to interpreted code...

Some minor porting work on the tools will be necessary to do development under linux. The effort would be greater for mac development, because the tools are inherently command line based.

The map editor and tools will not be released until after the game hits store shelves. To be completely clear: you are not legally licensed to create or use addon maps with the test.

I am hoping that this public review will turn up bugs before we complete the game. 50k lines of code is quite a bit to go over, but people familiar with previous games will have a good head start in the game module.

The best possible situation would be if exploration of the code evolves into a tier system, with either moderated or limited access lists that I can follow without being swamped. I can't afford to be too involved in helping everyone figure out the basics. I have plenty of confidence in the mod communities ability to work that out. :-)

Some people have been mistaking memory swapping in 1.08 for network problems.

We did controlled, back to back tests against the previous versions, and the networking is identical if you have enough memory.

The addition of the new character model and all the new menu code and graphics has caused the game to begin to have some swapping problems on 64mb machines if you have all the quality options up high or are running other things.

I am looking into what I can do to reduce memory consumption for the next release, but in the meantime, you can turn down sound quality, geometry, or texture detail to get rid of it.

If you have less than 64mb, go buy more memory! The final game will have an option to run in less memory, but the graphics and sound quality will be a lot lower.

———-

I made a simple change in the file management that I think is clearly a big win.

Instead of scanning for pak0.pk3, pak1.pk3, ... pak9.pk3 in game directories, I now scan the entire directory for all .pk3 files, and add them to the search path in alphabetical order.

This gives us the same needed functionality we have now – overriding things in pak0.pk3 with updated data in pak1.pk3, but it also gives a signficant benefit to the user community.

There has been a lot of requests to have textures inside maps like Quake1 did, but I was not willing to do that. Having the files separate saves an immense amount of duplication, and keeps the internal architecture uniform.

Now, you can just add all your new art into a pk3 (zip) file along with your map and users can just drop that single file into their quake3/baseq3 directory as a single operation.

This will also be nice for custom models, which require several component parts: legs, torso, head, animations, levels of detail, and skins.

So, if you are strictly ADDING media (models, textures, maps, etc), then you can just drop the pk3 files in the normal directory.

If you are REPLACING data (code modules, menus, etc), you should make a new game directory and put the pk3 there. Starting up with ”quake3 +set game mygame” will make all the mygame pk3 files override anything in baseq3. You could do it by just naming your pk3 file ”zzzstuff.pk3”, but then you wouldn't have a way to run the game without the addons.

A prudent person might choose to put ALL addons into a separate directory and leave baseq3 pristine for official additions.

———-

Other stuff that has been done lately:

* don't clamp dedicated server or client times until much later – prevents time resets under ordinary conditions
* fixed CG_ProcessSnapshots: cg.snap-> serverTime
* only drift time on receipt of packets with a steady ping – improve catchup after drops
* fixed players getting stuck together
* new pak file support
* derez mac resource file
* intro/loop option for music files
* fixed lerp-through-world when changing teams
* show ping and netgraph for spectators
* fixed timescale off by one problem
* tracked down player count wrong on server list
* info_spectator_start entity
* shader language change: clampTexCoord removed, added clampmap
* r_debugSort cvar for working on transparency sorting problems
* changed minimum cl_maxpackets to 15
* fixed can't-respawn-when-someone-is-on-your-body
* fixed dlighting over alpha tested surfaces
* z_stats lists all blocks > = given size
* fixed wasted model slots
* increased com_hunkmegs
* com_buildScript cvar to force loading all media and quit on errors
* fixed bad playerstate interpolation across teleporters
* converted local sounds to sfxHandle_t
* new fog code doesn't require subdivisions
* fixed sun positioning problem
* added fogging of triangle objects
* fixed devmap issue
* make g_log a filename instead of a 0/1
* g_logsync option to force a flush after each write must be set at time of log file creation

7.4 Aug 28, 1999

* spinning machinegun barrel
* changed q3data -origin option to -offset, defaulted to 0 0 24 for all player grabs
* removed second parm from -lod in q3data
* fixed 0 ping on last player killed before fraglimit
* better ping calculation right after transitions
* add time back to scoreboard
* sv_maxRate option to force all clients to play with a max rate. This can be used to limit the advantage of LPB, or to cap bandwidth utilization for a server. Note that rate is ignored for clients that are on the same LAN.
* fixed bad name vs name in tourney after first player left
* added hitch warning messages to server console
* new time clamping rules for net play
* avoid sending usercmds during connection
* send explicit heartbeats to the master server when a server transitions to or from empty or full
* shaders that aren't found will return index 0, but still keep the allocated slot to prevent rescanning if registered again
* use nextSnap for player prediction when available
* removed teleport dest invisible objects
* reduced client to server bandwidth by 35%
* changed logging for chats to guarantee parsing properly with names that conflict with commands:
from: G_LogPrintf("%s say: %s"
to: G_LogPrintf("say: %s: %s"

Chapter 8

September

8.1 Sep 01, 1999

I have been working on our memory footprint for the past couple days. There are two types of paging that occur:

One time only paging that eventually settles down after you have run around the entire level a few times. This can be a result of having lots of data around that isn't actively used during the game, or was only used at startup.

True capacity misses, where the game is actually touching more memory during play than you have available. This is usually due to there just being too many textures and models.

Loading a large map with full resolution, 32 bit textures and mipmaps consumes almost 40 megs of texture space. The default of picmip 1 and 16 bit textures reduces that to six or seven (not all images observe picmip) in theory, but some OpenGL implementations keep a 32 bit version of the texture around even if the card only uses 16 bit (I consider this inappropriate), so it may still be twelve or more megs just for the images.

I have been able to save a couple megs off of the true capacity requirements by not rolling through some larger buffers when not needed (not smp and not dedicated server), and several megs more of one-time data mostly by moving a lot of static tables to dynamic allocation.

I can probably save another meg of true capacity and I think three or four megs of initialization data. The big thing that I might be forced to do is go to a skeletal animation system. I would hate to do that at this late a point in the project, but it would save about two megs per player model in the game.

In the process of chasing down the static memory hogs, I finally got around to starting something I have needed to do for years: learn perl.

I was scanning through a linker map file looking for large gaps in addresses, thinking to myself ”this is one of those things you can probably do in three lines of perl code”. I have many and varied excuses for why I have never gotten around to it before, mostly involving the fact that I have C parsing code that lets me get what I need done with only minimal headache when I do force myself to do some text file grovelling.

I decided my excuses weren't good anymore, and went out to the bookstore and grabbed the llama book. Many of you would have been amused seeing me go through the

print ”Hello, $name!”

tutorial code as I did the examples in the first couple chapters. :-)

I got my task done, so now I just need to force myself to write little perl programs whenever a need comes up, until I get fluent with it.

* save 2.5 megs by reworking shader allocation
* save 1 meg by not double buffering backend if not smp
* convert all tr. arrays into pointers
* don't allocate as many snapshotentities when non dedicated
* new shader option: deformvertex move
* stackable deformvertex
* reduced lightning damage by 10%
* light emit from two sided surfaces
* reduced starting machinegun ammo in teamplay to 50 from 100

8.2 I have been getting a lot of requests for commentary on two subjects lately (Sep 02, 1999)

Nvidia's new geometry accelerated card with the funny name.

It is fast. Very, very fast. It has the highest fill rate of any card we have ever tested, has improved image quality over TNT2, and it gives timedemo scores 40% faster than the next closest score with extremely raw beta drivers.

The throughput will definately improve even more as their drivers mature.

For max framerates in OpenGL games, this card is going to be very hard to beat.

Q3's target of about 10,000 triangles a frame doesn't stress this card at all. If you want more polygons out of Q3, you can do:

```
r_lodBias -2 // don't use lower detail models
r_subdivisions 1 // lots more triangles in curves
r_lodCurveError 10000 // don't drop curve rows for a long time
```

I haven't looked at the stencil shadow stuff in a long time, but it gives the largest increase in triangle use (and a lot of fill rate as well):

```
cg_shadows 2 // turn on stencil shadows (if you have a stencil buffer)
```

Apple's new G4 systems.

The initial systems are just G4 processors in basically the same systems as the current G3. There will be some speedup in the normal C code from the faster floating point unit, and the Apple OpenGL has AltiVec optimizations, so framerates will improve somewhat. The limiting factor is going to be the fill rate on the rage128 and the bandwidth of the 66mhz pci bus and processor to main memory writes.

The later G4 systems with the new memory controller and AGP will have better performance, but probably still limited by the new 3D card.

After Apple gets all their driver tuning done, it will be interesting to try running timedemos at low resolution to factor the fill rate out. Apple has a shot at having the best non-geometry accelerated throughput, but it will still be tough to overcome a K7 with an extra hundred or so mhz.

On a purely technical note, AltiVec is more flexible for computation than intel or AMD's extensions (trinary ops), but intel style write combining is better for filling command buffers than the G4's memory streaming operations.

8.3 Sep 06, 1999

It looks like we are going to go to a skeletal model system. Jim Dose of Ritual had already started on an exporter from character studio, so we decided to just meet in the middle.

I implemented the loading and rendering support this weekend and tested it with a couple hand-inserted bones, so now we just need to write the glue between character studio and the new .md4 format.

The new format is bone based, but it is NOT hierarchial. Each vertex just has an arbitrary weighted list of the bones that influence it. Bones are just 4x3 matricies of floats.

A hierarchial skeleton has some advantages (angles instead of matricies, ability to do IK, etc), but this is a direct and simple replacement for our existing infrastructure that doesn't require any cached state per model instance.

A single .md4 file holds multiple level of detail surface sets, which all share the same bone frames.

In use, it is exactly like the existing models (interpolate between two frame numbers), it just saves a huge amount of space.

I used perl to generate my test data, and it was definately faster than having a separate msdev open and doing it in C.

I am trying to use parenthesis on all perl functions, but when I type ”print”, my fingers seem to have a flashback to applesoft basic fifteen years ago, and I wind up with bare quotes on prints and parens on everything else...

Does anyone know if there is an existing msdev syntax coloring file for perl? (no, I don't want to switch to a different editor!)

* md4 model loading and displaying
* removed clip models from cgame, use renderer models instead
* fixed mover pushing again
* fixed bug with culling of mirrors made of multiple faces
* fixed quad on spinning machinegun
* surfaceparm alphashadow - This causes q3map -light to check individual texture pixels on transparant surfaces for light shadowing instead of making the entire surface either cast or not cast shadows

8.4 Ok, obviously we didn't get a release out in the middle of the month... (Sep 28, 1999)

We are still hashing out the single player game, so it still isn't immediately immenent.

* weapon switch animations at 20hz instead of 15hz, cuts switch time from 600 msec to 450 msec
* initial spawn flag for single player
* finished new fog code
* fixed walking underwater friction problems
* autosprite2 now selects the longest axis to pivot on, and allows any texture mapping, not just unity
* fixed autosprite on entities
* fixed lurching during low timescale
* reduced machinegun damage
* set clamp mode on 2D pics
* take hostname off of single player connect
* remvoed dlighting on skies and solid lava
* fixed lower body twitch when copytobodyque with a motion
* always show your score in second box if not in first
* sarge as default model

* com_blood 0 option for no gibs and no blood on hits
* mouse click aborts cinematic
* show tourney scores in all games, add fraglimit
* removed tripple bunny-hop protection, it was too arbitrary and didn't accomplish it's goal
* pump event loop during level loading
* added pass count to shaderlist
* default to CGEN_IDENTITY / CGEN_IDENTITY_LIGHTING based on blendSrc
* optional simplified blendfuncs: blendfunc
* new shader command: deformVertexes normal
* new shader command: tcgen vector ( ) ( )
* fixed fog on alpha tested surfaces
* reduced com_maxfps to 85
* defined shaders for menu and console backgrounds
* reset players on clientinfo transitions
* windows icons
* fixed powerups on spinning barrels
* removed some latency from lightning endpoint
* fixed lightning bolt drawing too far
* removed color clamping from entity lighting
* moved all 2D drawing to shader pipeline
* r_printShaders tool
* moved dlighting into world node descent
* pause when menu is up in single player
* fixed double EV_FIRE_WEAPON
* r_singleShader optimization tool
* some renderer optimizations
* better multitexture collapsing

8.5 Sep 29, 1999

I wrote this in answer to a question on the mac opengl programming list, but it is of general enough interest to programmers that I am repeating it here.

> You also mentioned display lists... Can you explain what some of the
> major things that should be rendered using a display list are? I see
> the importance of using them for characters (animation) and objects, but
> what about using them for the rest of the world, particles, and other
> things that are nifty.

This is not yet a big issue, although even pure software OpenGL's could perform some optimizations with display lists that aren't possible with vertex arrays. With hardware geometry acceleration, it can be an honest 4x improvement in throughput.

The important point is that once geometry acceleration becomes a primary target, practically everything will have to be rendered with display lists, or you will run into a nasty case of Amdahl's law.

In a busy Q3 battle, the triangle count may be split roughly evenly between character models and world geometry. Going from an empty scene to a pitched battle can result in a 50% performance drop if you are triangle limited. Not great, but livable.

If we kept the same ratios and designed for geometry acceleration with all the static world geometry in display lists, then the empty scene could have 4x the geometry and still be running the same speed. However, current OpenGL display lists can't really accelerate high quality skinned characters, so when an equal number of character polygons was in scene and passed through normal direct rendering, the performance would drop to 20% of the original. Unacceptable.

So, either you would have to use significantly different polygon counts in characters and the world, or some new API features would need to be defined. Nvidia has a skinning extension that gives some benefit, but still requires a character to be broken up into one static list per bone pair, instead of a single list for the entire character.

Rendering a few thousand particles or other procedurally generated triangles directly isn't going to be a big issue, but the bulk of the work is going to move towards static vertex data.

My advice for display lists is to use them for just raw vertex/color/texcoord data, and keep your state changes done with direct commands. This allows you to still sort display lists to minimze state changes, and prevents drivers from ever having to check state internally. Some hardware architectures can nicely encapsulate all state changes in a single dma buffer, but register sharing among different fields sometimes requires the driver to do manual masks, negating much of the async display list benefits. Texture swapping also complicates state changes inside display lists.

You want to make the display lists as big as practical, but there is a tradeoff betwen culling tightness and display list size.

Chapter 9

October

9.1 Oct 01, 1999

* 10 crosshairs to select from (cg_drawCrosshair 1 - 10)
* cg_crosshairx / cg_crosshairy adjustment. I'm not convinced these are a good thing, because the crosshair is accurate in Q3 at the default position (unlike Q2, which had an offset firing position)
* more packet encryption
* join as spectator in all team games
* cg_predictItems 0 option to not do local prediction of item pickup
* rank players counting ties, so the third player when the lead is tied is third place, not second
* properly stack all status elements in upper right and lower right corners so they can all be visible
* faced q3map problem giving black corners in vertex light
* fixed +button4 not causing footsteps
* fixed bad groundplane clipping on angled jumppads
* show "snc" on lagometer when g_syncronousClients
* new shader command: rgbgen const ( )
* new shader command: agen const
* snap dropped item positions
* randomized offsets of bubbles
* fixed cgame restarts processing snapshots from 0

* proper setup for external/predictable player events
* fixed q3map vertex lighting bug after alphashadows
* do personal pain sounds on health transitions so they can never be missed
* timeout clear player events
* fixed hang in UI_ProportionalStringWidth
* quad event implicit on weapon fire
* fixed gamestate not retransmitting bug
* fixed timeout issue when paused
* subdivide command times if < 15fps, fixing low com_maxfps physics exploits
* fixed footsteps playing when walking backwards
* new options at start of animation.cfg file: footsteps, headoffset

9.2 Oct 04, 1999

* allow cg_thirdPerson and cg_thirdPersonRange in games
* added execed .cfg files to journal file
* single pass plasma explosion effect to save overdraw
* rescaled and sized gib and mark blood to save a lot of overdraw
* fixed cg.time < snapshot time on vid_restart
* cl_showSend network debugging tool
* back to requiring a before commands on the console to distinguish them from chat messages. Tab completion automatically adds the slash.
* new tab command completion with complete to longest common, multiple match listing, complete first token, etc
* separate version check for game/cgame in addition to system
* show r_finish in gfxinfo
* never show self as attacker icon
* callvote vote - Caller automatically votes yes vote has a 30 second timeout each client can only call 3 votes a level vote is displayed on screen with totals
* renamed cg_gun to cg_drawGun
* box cull triangle soup models

9.3 Oct 05, 1999

* fixed steady snapshot test
* fixed incorrect 0 ping if past client messages
* fixed loser-disconnecting-at-tourney-intermission sorting problem
* general purpose flood protection, limiting all user commands to one a second by stalling the client, so the commands don't actually get dropped, but are delayed as needed
* replace headnode overflow with lastCluster
* fixed bad extrapolation on unpausing
* fixed player twitch on unpausing
* print client names on loading screen

9.4 Oct 07, 1999

* r_primitives 3 path for non-vertex array testing
* specify sex in model animation.cfg file
* proper dropping of failed bot inits
* removed identical pain sounds
* serverTime strictly increasing across levels
* added GL_DECAL multitexture collapse
* windowed mouse on mac
* fixed byte order issue with curve clipping on mac
* made com_defaultextension buffer safe
* fixed levelshot and added antialiasing to image
* don't clear bot names before kick message
* made servercommand sequences strictly increasing across level changes
* unpause on vid_restart

9.5 Oct 11, 1999

* handle window close events properly
* enable r_displayRefresh selection and feedback on mac

* avoid AGEN_IDENTITY when CGEN_DIFFUSE_LIGHTING
* colorized railgun muzzle flash, impact flash, and mark
* exactly synced animating textures with waveforms and collapsed all explosion sequences into single shaders
* removed unneeded black pass on hell skies
* fixed grenades sticking to steep slopes
* scan for next highest fullscreen resolution when exact mode fails (fixes SGI flat panel failing to init)
* all cgame cvars now have a cg_ prefix (crosshair, fov, etc)
* clear clientinfo before parsing configstring
* make all feedback voiceovers share the same channel
* fixed nodraw curves
* fixed obits from shooter entities
* fixed chat char issue
* separate gentity_t fields into sharedEntity_t
* reintegrated q_mathsys with q_math
* cg_forcemodel also forces player sounds
* unknown cmd commands don't chat
* fixed strip order on text quads

9.6 Oct 14, 1999

* make sure video is shutfown for errors on startup
* automatic fallback to vm if dll load fails
* compressed jump tables for qvm
* removed common qfiles.h and surfaceflags.h from utils and game
* don't load qvm symbols if not running with developer
* "quake3 safe" will run the game without loading q3config.cfg
* ignore network packets when in single player mode
* dedicated server memory optimizations. Tips:
com_hunkMegs 4
sv_maxclients 3
bot_enable 0
* fixed logfile on mac
* new time drifting code
* fixed file handle leak with compressed pk3 files

* q3data changed to remove shader references from player models
* throw a fatal error if drop errors are streaming in
* fixed com_hunkMegs as command line parm
* spawn spectators at intermission point (info_spectator_start has been removed)
* new sound channel for local sounds
* fixed follow toggle on bots
* don't write to games.log in single player
* fixed improper case sensitivity in S_FindName

9.7 Oct 17, 1999

The next release will be the full "demo" release for quake 3. It will include bots and a new, simple level that is suitable for complete beginners to play, as well as the existing q3test maps.

The timing just didn't work out right for another test before we complete the game.

We plan on releasing the demo after code freeze, when the entire game is in final testing, which will give us a few days time to fix any last minute problems that show up before golden master.

No, I don't have an actual date when that will be.

-

I got an iBook in on friday. It is sort of neat (I need to go buy an AirPort, then it will definately be neat), but it is currently way too slow to play Q3 on.

Apple's high end G3 and G4 systems with rage128/rage128pro cards and latest drivers are just about as fast as equivelant wintel systems, but the rest of the product line is still suffering a noticable speed disadvantage.

The new iMac has a rage128, but it is only an 8mb/64bit version. Still, with agp texturing it is a solid platform with performance that is certainly good enough to play the game well on.

Existing iMacs have 6mb ragePro video. ATI's windows drivers for the pro have come a long ways, and Q3 is plenty playbale on windows with a rage pro (ugly, but playable). On apple systems (iMacs and beige G3's), the performance is sometimes as low as HALF the windows platform. The lack of AGP contributes some to this, but it is mostly just a case of the drivers not being optimzed yet. The focus has been on the rage128 so far.

The iBook is also ragePro based, but it is an ultra-cheap 32 bit version. It does texture over AGP, but it is slooooow. I suspect it is still driver issues, because it is still very slow even at 320x240, so that leaves hope that it can be improved.

Another issue with the Apple systems is that Apple 16 bit color is actually 15 bit. I never used to think it made much difference, but I have been doing a lot of side by side comparying, and it does turn out to be a noticable loss of quality.

* new lg splash
* added channel number for local sounds so feedbacks don't override announcers
* removed scoreup feedback sound
* expand score tabs as needed for large scores
* fixed bfg obit
* fixed swimming off floors
* fixed swim animation when jumping in shallow water
* fixed first weapopn switch problem
* convert all joystick axis to button events (twist is now bindable)

9.8 An announcement (Oct 23, 1999)

We have hired Robert Duffy as a full time employee, starting in December.

He has been maintaining the level editor since the release of Q2, but a number of things have convinced me it is time to have a full time toolguy at id.

The original QE4 editor was my very first ever Win32 program, and while

Robert has done a good job of keeping it on life support through necessary extensions and evolutions, it really is past its designated lifespan.

I want to take a shot at making the level editor cross platform, so it can be used on linux and mac systems. I'm not exactly confident that it will work out, but I want to try.

Many of the content creation tasks seem to be fairly stabilized over the last several years, and our next product is going to be pretty content directed, so I can justify more engineering effort on writing better tools.

It is time for a re-think of the relationships between editor, utilities, and game. I am still an opponent of in-game editors, but I want to rearrange a lot of things so that some subsystems can be shared.

All of that added up to more than I was going to be able to do in the time left after the various research, graphics, and networking things I want to pursue.

* added r_speeds timing info to cinematic texture uploads
* fixed loop sounds on movers
* new bfg sound
* "sv_pure 1" as default, requires clients to only get data from pk3 files the server is using
* fixed fog pass on inside of textured fog surfaces
* properly fog sprites
* graphics for scoreboard headers
* show colored names on attacker display and scoreboard
* made "no such frame" a developer only warning
* count a disconnect while losing in tournement mode as a win for the other player
* fixed running with jump held down runs slow
* draw "connection problems" in center of screen in addition to phone jack icon
* cut marks properly on non-convex faces
* fixed bug with command completion partial match and case sensitivity
* fixed console dropping on level start
* fixed frags left feedback after restarts
* fog after dlight
* removed fogged stages from shader, dynamically generate

* removed fogonly shader keyword, infer from surfaceparm
* removed uneeded reinit of shader system on vid_restart

Chapter 10

November

10.1 Nov 01, 1999

* play chat sound during votes
* draw 2D icon for ammo if cg_draw3dicons 0
* fixed losing input on menu vid_restart
* made ”vote” and ”callvote” completable
* remove mac about dialog
* fixed demos skipping inital time due to loading
* fixed timing on timedemo startup
* don’t flash attacker when damaging self
* display capturelimit instead of fraglimit on score tabs in ctf
* recursive mirror/portal changed to a developer warning
* fixed bug with follow mode spectators
* battle suit shader
* notsingle spawn option
* separated torso and legs priority animation coutners so gesture doesn’t mess with legs
* adjusted value to prevent missed launch sound on accelerator pads
* setviewpos x y z yaw, same parms as viewpos command
* stop sound on file access
* fixed developer prints from renderer
* defered client media loading, only load models and sounds for new

players when you die or bring up the scoreboard
* fix for double colliding against same plane and getting stuck
* dropped LG damage to 160 pts / sec
* don't snap predicted player entity, smooths deaths and movers
* all sine movers are instant kill on block
* fixed items riding on bobbers

10.2 Nov 02, 1999

* remove grapple from give all
* fixed pick-up-two-healths-even-if-you-don't-need-the-second bug
* moved wrap/clamp to image generation function and added to imagelist, fixed an improper clamp on macs
* different menuback for ragepro
* fixed mac button problems with OS9 and wheel mice
* teamplay rule mods:
less MG damage (5 instead of 7)
weapons always have a full load of ammo
don't drop powerups
* changed low detail r_subdivisions to 25 to prevent poke through
* removed warning on empty servercommand when systeminfo changes
* g_debugDamage
* when a vote is tied with all votes in, immediately fail it
* haste smoke puffs

10.3 Nov 03, 1999

* cd check in single player
* removed drop shadow on console input line
* swapped mynx pain sounds
* fixed cleared music buffer on track loop again
* force skins on spectators in team games to prevent having a default waste memory
* only defer to a model with same team skin

* fixed grenade-disappearing-at-floor bug when about to explode
* draw reward medals during gameplay
* added "humiliation" feedback for gauntlet kills
* spread respawn times to prevent pattern running:

```
#define RESPAWN_ARMOR 25
#define RESPAWN_TEAM_WEAPON 30
#define RESPAWN_HEALTH 35
#define RESPAWN_AMMO 40
#define RESPAWN_HOLDABLE 60
#define RESPAWN_MEGAHEALTH 120
#define RESPAWN_POWERUP 120
```

10.4 Nov 05, 1999

* check for bad weapon number in non-3d ammo icon on death
* fixed plane catagorization
* error command does an ERR_DROP if given a parm
* don't load high LOD models if r_lodbias
* nobots/nohumans options for player spawn spots
* prevent voice rewards on frag that enters intermission
* dissallow native dll loading if sv_pure
* loaddefered cgame command, issued on addbot
* drop the weapon you are changing TO if you only had a MG or gauntlet
* fixed bounce pad event prediction for all angles
* allow empty tokens in map files
* fixed infos exceeded warning on bot parse
* warning on mismatched mipmap/picmip/wrapclamp image reuse
* fixed pain echo sound after predicted falling damage
* move sound to hunk
* move vm to hunk
* restart game vm in place for map_restarts
* avoid all lightmaps entirely when using vertex light
* pretouch all images after registration

* pretouch all known in-use memory before starting a game
* on error, shutdown client before server, to be more likely to get out of fullscreen before a recursive error
* new pre-allocated memmory manager to crutch up mac
* meminfo command replaces hunk_stats and z_stats
* adjusted scoreboard titles
* no guantlet reward on corpses
* fixed snd_restart when paused
* PRE_RELEASE_DEMO hack

10.5 Nov 09, 1999

* fixed spinning barrel on respawn issue
* clear eflags before intermission
* shutdown menu on starting a cinematic
* mask name colors to 0-7 range
* fixed jpeg loading alpha channel
* try for not-nearest spawn twice instead of once
* made unzoomed exactly identity mouse modifier
* cl_debugmove [1/2]
* m_filter
* fixed time warnings
* allow timelimits to hit with only a single player
* filter local games with different protocol versions
* fixed bad arg 0 after sysinfo configstring
* removed unneeded svc_servercommand at start of command strings
* fixed redundantly loaded level bug
* fixed journal playback from demo build
* removed background image from viewlog window

10.6 Nov 11, 1999

* teamplay menu comment
* shrank and moved "RECORDING demo:" text

* identified and worked around Apple input queue issue
* properly send configstring resets on map_restart
* don't clip sound buffer on file writes
* don't draw scoreboard during warmup
* auto load added bots in single player
* swapped order of map_restart and warmup configstring
* disable dynamic lights on riva128 due to lack of blend mode
* put frags left warning back in all gametypes
* removed joystick button debug prints

10.7 Nov 13, 1999

* graphic for defer
* don't set any systeminfo vars from demos
* A3D fix
* spectator follow clients that disconnect
* stop follow mode before going to intermission so you can ready
* use (fullbright) vertex lighting if the bsp file doesn't have lightmaps
* auto set demo keyword on servers
* finished cd key authorization
* fixed symbol table loading for interpreter
* reconnect command
* removed limit on number of completed commands
* changed default name to "UnnamedPlayer"
* awards over people's heads in multiplayer
* fixed global powerup announcements

10.8 The demo test is built. It should go up in a couple hours if nothing explodes. (Nov 14, 1999)

Mac and linux builds won't be out tonight.

* clear SVF_BOT when exiting follow mode
* render temp memory
* new mac GL initialization code
* no zone memory use in music thread
* added check for trash past zone block
* explicitly flush journal data file after a write
* added FS_Flush

10.9 Nov 15, 1999

The way vertex lighting is working in the existing demos is that only two pass shaders (lightmap * texture) were collapsed to a single pass, all other shaders stayed the same.

Xian added some chrome and energy effects to parts of q3tourney2, which changed them from two pass to three pass shaders. We felt that that 50% increase on those polygons was justified in normal play, but as people have pointed out, when you are playing with vertex lighting, that three passes stays three passes instead of collapsing to a single pass, resulting in a 300% increase on those polygons over the way it was before. Still faster than lightmap mode, but a large variance over other parts of the level.

Today I wrote new code to address that, and improve on top of it.

Now when r_vertexlight is on, I force every single shader to a single pass. In the cases where it isn't a simple light*texture case, I try and intelligently pick the most representative pass and do some fixups on the shader modulations.

This works our great, and brings the graphics load down to the minimum we can do with the data sets.

Performance is still going to be down a couple msec a frame due to using dynamic compilation instead of dll's for the cgame, but that is an intentional tradeoff. You can obviously slow things down by running a lot of bots, but that is to be expected.

I am still investigating the high idle dedicated server cpu utilization and a few other issues. The server cpu time will definately be higher than 1.08 due to the dynamic compiler, but again, that is an intentional tradeoff.

A set of go-fast-and-look-ugly options:

```
r_mode 2
r_colorbits 16
r_texturemode GL_LINEAR_MIPMAP_NEAREST
r_vertexlighting 1
r_subdivisions 999
r_lodbias 2
cg_gibs 0
cg_draw3dicons 0
cg_brassTime 0
cg_marks 0
cg_shadows 0
cg_simpleitems 1
cg_drawAttacker 0
```

* icons for bot skills on scoreboard
* r_vertexlight is now ”force single pass” for all shaders
* modified cd key check to be fire and forget on the client
* file handle debugging info in path command
* network address type of NA_BAD for failed resolves
* better command line variable overriding
* cache scoreboard for two seconds
* sync sound system before starting cinematics
* fixed many escapes disconnect from server exiting the game
* fixed shotgun pellets underwater expending all temp entities

10.9. NOV 15, 1999

10.10 The demo servers have general purpose flood-protection that has caused some confusion. (Nov 16, 1999)

Clients are only allowed to make one command a second of any kind. This prevents excessive flooding by chats, model / name changes, and any other command that could possibly be exploited. The command streams are stalled, so it doesn't have any effect on processing order or reliability.

This means that if you issue two commands immediately after one another, there will be a one second stall before the second command and all movement clears. You see this on the lagometer as yellow spiking up for a second, then dropping away.

Hitting tab for the scoreboard sends a command, so you trigger the flood protection if you bang tab a couple times. This has been fixed so that the scoreboard will never send more than one update request every two seconds, but you will need to watch out for it in the existing demo.

The defered model loading has also caused some confusion, but that is a feature, not a bug. :)

In previous tests, you hitched for a second or two whenever a client with a new model or skin joined a game.

In the demo, when a client joins the game they will be given the model and skin of someone else temporarily, so there is no hitch. The only time it will hitch on entry is if it is a team game and there isn't anyone on the team they join. I make sure the skin color is correct, even if the model isn't.

These ”defered” clients will be loaded when you bring up the scoreboard. You can do this directly by hitting tab, or you can have it happen for you when you die.

The point is: you died BEFORE it hitched, not as a result of the hitch.

The scoreboard header is up, but it is still a bit easy to miss.

* fixed high server idle cpu usage (it was spinning in place until maxfps was used!)
* fixed g_password, which is crashing in the demo
* moved svs.snapshotEntities to the hunk
* enable lagometer whenever running a non-local game
* cg_drawTeamOverlay cvar, set to 0 by default
* finished authorize work
* better reporting of unused highwater memory

10.11 The mac version is out. Go to www.quake3arena.com for links. (Nov 18, 1999)

The mac version going out has the executable fixes that we have made in the last couple days, but most of the fixes have been in code that runs in the virtual machine, and we can't update that without making it incomptable with the pc version.

The game remains very marginal in playability on 266mhz imacs and iBooks.

A 333mhz imac should be playable for a casual gamer if the graphics options are turned down to the "fastest" setting.

There is still a lot of room for improvement on ATI's side with the RagePro drivers. Almost all the effort so far has been on the Rage128 drivers.

The G3 systems run fine, but a little slower than a pc of equal mhz

The rage128 cards in the G3s are only clocked at 75mhz, so you can't run too high of a resolution, but you can get very nice image quality. I usually play with these settings:

```
r_mode 2 // 512*284 res
r_colorbits 32 // 32 bit color
r_texturemode gl_linear_mipmap_linear // trilinear filtering
```

I haven't played on one of the new iMacs or G4's but they both use the rage128 driver, which is fairly high quality now, so they should perform well.

We found a fairly significant problem with inputSprockets and mouse control (motion is dropped after 40msec). I have done a little working around it, so mouse control should be somewhat better in this version, but it will hopefully be fixed properly by Apple in the next IS rev. It isn't an issue if your framerate is high enough, but iMacs never see that framerate on their very best days..

Linux version tomorrow night, if nothing horrible happens.

Some advance warning about something that is sure to stir up some argument:

We should be handing off the masters for all three platforms within a day or two of each other, but they aren't going to show up in stores at the same time. Publishers, distributers, and stores are willing to go out of their way to expedite the arrival of the pc version, but they just won't go to the same amount of trouble for mac and linux boxes.

THE EXECUTABLES FOR ALL PLATFORMS WILL NOT BE AVAILABLE FOR DOWNLOAD UNTIL AFTER CHRISTMAS. This means that if you want to play on the mac or linux, don't pick up a copy of the pc version and expect to download the other executables.

Our first update to the game will be for all platforms, and will allow any version to be converted into any other, but we intend to hold that off for a little while.

We are doing this at the request of the distributors. The fear is that everyone will just grab a windows version, and the separate boxes will be ignored.

A lot of companies are going to be watching the sales figures for the mac and linux versions of Q3 to see if the platforms are actually worth supporting. If everyone bought a windows version and the other boxes sold like crap in comparison, that would be plenty of evidence for most executives to can any cross platform development.

10.11. THE MAC VERSION IS OUT. GO TO WWW.QUAKE3ARENA.COM FOR LINKS. (NOV 18, 1999)

I know there are a lot of people that play in both windows and linux, and this may be a bit of an inconvenience in the short term, but this is an ideal time to cast a vote as a consumer.

Its all the same to Id (I like hybrid CD's), and our continued support of linux and mac (OS X for the next title) is basically a foregone conclusion, but the results will probably influence other companies.

* fixed getting your own dropped / kicked message
* added developer print for all file open write's
* fixed occasional bad color on connecting background
* fixed occasional telefrag at start of skirmish game
* fix not being able to ready at intermission if you were following a bot
* never timelimit during tourney warmup
* fixed local timer on map_restart
* offset sorlag's head model for status bar
* added g_gametype to the votable commands: map, map_restart, kick, g_gametype
* changed sound registration sequence to fix losing default sound
* "sv_floodProtect 0" to turn off flood protection
* converted sequenced messages to static arrays
* fixed custom skin reassignment on all LOD

10.12 Nov 19, 1999

Linux version isn't going to make it tonight. We got too busy with other things. Sorry. Tomorrow.

* shrink zone, grow hunk
* flush memory on an error
* fixed crash pasting from clipboard
* test all compiler optimizations - 5% speedup
* fixed major slowdown in team games with large numbers of players and location markers

10.13 Nov 21, 1999

* fixed not-telefragging bug
* disabled flood protection with local clients
* fixed headoffset and gender on some model changes
* init cg_autoswitch cvars in cl
* fixed clearing of vm bss on restart
* fixed hang when looped part of song isn't found
* fixed two NAT clients connecting to same server
* fixed warning on random once only triggers
* added "g_allowVote 0"
* added developer background sound underrun warning
* move sound loading before clients so low memory defer works across maps
* changed cgame load failure to a drop

10.14 d:> mkdir research (Nov 22, 1999)

Ahhhh...

I am very happy with how Q3 turned out. Probably more than any game we have done before, it's final form was very close to its initial envisioning.

I will be getting all the Q3 code we are going to release together over the next week or so. I will write some overview documentation to give a little context, and since you can do game mods without needing a commercial compiler now, I will write a brief step-by-step to modifying the game code.

I'm looking forward to what comes out of the community with Q3.

The rough outline of what I am going to be working on now:

We will be supporting Q3 for quite some time. Any problems we have will get fixed, and some new features my sneak in.

I have two rendering technologies that I intend to write research engines for.

I am going to spend some time on computer vision problems. I think the cheap little web cams have some interesting possibilities.

I am going to explore some possibilities with generalizing 3D game engines into more powerful environments with broader uses. I think that a lot of trends are coming to the point where a ”cyberspace” as it is often imagined is begining to be feasible.

I am going to spend more time on some Free Software projects. I have been stealing a few hours here and there to work on the matrox glx project for a while now, and it has been pretty rewarding. People with an interest in the guts of a 3D driver might want to look at the project archives at `http://glx.on.openprojects.net/.` The web pages aren't very up to date, but the mailing list covers some good techie information.

Chapter 11

December

11.1 WANTED: Independant OpenGL conformance nazi (Dec 12, 1999)

I think there is a strong need for a proactive, vendor-neutral OpenGL watchdog, or even a small group, especially in the linux space.

I have been working on the utah-GLX team for quite a while now, and while I have been very pleased with the results, I would like to see more effort spent on doing things as right as possible. Because the developers (me included) are basically just doing the work in their spare time, testing usually only consists of running their favorite OpenGL application, and a few of the mesa demos, or some of the xscreensaver hacks.

Recently I did the initial bringup of a RagePro driver on linux, and I was much more conscious of the large untested feature space, and the tunnel vision I was using to get it to the point of running Q3.

What we need is someone, or a group of someones, who can really exercise different implementations through all corners of the OpenGL specification and provide detailed lists of faults with minimal test cases to reproduce the behavior.

In most cases, the bugs could then be fixed, but even if it is decided that the incorrect behavior is going to stay (to avoid a software fallback in a common accelerated case), there would be clear documentation of it.

I consider performance on the matrox driver right now to be ”good enough”. There is definately more performance oriented work going on, but given a choice of tasks to work on, I would rather improve quality and coverage instead of kicking a few more fps out of Q3.

One of Alex St. John’s valid points was that ”The drivers are always broken”. There are a lot of factors that contribute to it, including fierce benchmarking competition causing driver writers to do some debatable things and diminish focus on quality. With open source drivers, some of those factors go away. Sure, it is nice to beat windows drivers on some benchmarks, but I wouldn’t let pursuit of that goal introduce dumb things into the code.

Some of the windows IHVs have good testing proceedures and high quality drivers, but even there, it would be nice to have someone hounding them about things beyond how well quake releated games run.

The same goes for Apple, especially now that there is both ATI and 3dfx support.

Conformance would be my primary interest, but characterizing the performance of different drivers would also be usefull, especially for edge cases that may or may not be accelerated, like glDrawPixels.

On linux right now, we have:

The traditional fullscreen 3dfx mesa driver
The DRI-GLX based banshee/voodoo3 driver
The utah-GLX matrox G200/G400 driver
The temporary utah-GLX nvidia driver
The newly born utah-GLX ATI Rage Pro driver

If anyone is interested, join the developer list off of: `http://glx.on.openprojects.net/`

Doing a proper job would require a really good knowledge of the OpenGL specification, and a meticulous style, but it wouldn’t require hardcore registers-and-dma driver writing skills, only basic glut programming.

If someone does wind up developing a good suite of tools and procedures and gives one of the drivers a really good set of feedback, I would be happy to provide extra video cards so they could beat up all the implementations.

11.2 Dec 15, 1999

Anna Kang left Id a couple weeks ago to found her own company - Fountainhead Entertainment.

It wasn't generally discussed during her time at Id, but we had been going out when she joined the company, and we were engaged earlier this year. We are getting married next month, and honeymooning in Hawaii. At her thoughtful suggestion, we are shipping a workstation out with us, so I don't fall into some programming-deprivation state. How great is that? :)

Now that Q3A has shipped, the official winner of her Id Software figurine chess set contest is Rowan Crawford for his prose and art.

An honorable mention goes to Reine Hogberg and Peder Hardings for their Q3A Blair Witch Project. They will receive silver Q3A medallions.

11.3 Dec 21, 1999

The Q3 game source code is getting pushed back a bit because we had to do some rearranging in the current codebase to facilitate the release, and we don't want to release in-progress code before the official binary point release.

We still have a Christmas present for the coders, though:

`http://www.idsoftware.com/q1source/`

Happy holidays!

11.4 Dec 25, 1999

There are a number of people upset about the Quake 1 source code release, because it is allowing cheating in existing games.

There will be a sorting out period as people figure out what directions the Quake1 world is going to go in with the new capabilities, but it will still be possible to have cheat free games after a few things get worked out.

Here's what needs to be done:

You have to assume the server is trusted. Because of the way quake mods work, It has always been possible to have server side cheats along the lines of "if name == mine, scale damage by 75%". You have to trust the server operator.

So, the problem then becomes a matter of making sure the clients are all playing with an acceptable version before allowing them to connect to the server. You obviously can't just ask the client, because if it is hacked it can just tell you what you want to hear. Because of the nature of the GPL, you can't just have a hidden part of the code to do verification.

What needs to be done is to create two closed source programs that act as executable loaders / verifiers and communication proxies for the client and server. These would need to be produced for each platform the game runs on. Some modifications will need to be done to the open source code to allow it to (optionally) communicate with these proxies.

These programs would perform a robust binary digest of the programs they are loading and communicate with their peer in a complex encrypted protocol before allowing the game connection to start. It may be possible to bypass the proxy for normal packets to avoid adding any scheduling or latency issues, but it will need to be involved to some degree to prevent a cheater from hijacking the connection once it is created.

The server operator would determine which versions of the game are to be allowed to connect to their server if they wish to enforce proxy protection. The part of the community that wants to be competetive will have to agree to some reasonable schedule of adoption of new versions.

Nothing in online games is cheat-proof (there is allways the device driver level of things to hack on), but that would actually be more secure than the game as it originally shipped, because hex edited patches wouldn't work any more. Someone could still in theory hack the closed source programs, but that is the same situation everyone was in with the original game.

People can start working on this immediately. There is some prior art in various unix games that would probably be helpfull. It would also be a good idea to find some crypto hackers to review proposed proxy communication strategies.

11.5 I have been playing a lot of Q3 on a 28.8 modem for the last several days. (Dec 29, 1999)

I finally found a case of the stuck-at-awaiting-gamestate problem that turned out to be a continuous case of a fragment of the gamestate getting dropped. I have changed the net code to space out the sending of the fragments based on rate.

Note that there have been a few different things that result in stuck at gamestate or stuck at snapshot problems. We have fixed a few of them, but there may well still be other things that we haven't found yet.

You can still have a fun game on a 28.8 modem. It is a significant disadvantage, no question about it, but you can still have a good game if you play smart. If there is someone that knows what they are doing on a server with a ping in the low 100s, there won't usually be much you can do, but a skilled modem player can still beat up on unskilled T1 players..

Make sure your modem rate is set correctly. If you have it set too high, large amounts of data can get buffered up and you can wind up with multiple seconds of screwed up delays.

Only play on servers with good pings. My connection gives me a couple dozen servers with mid 200 pings. 56k modems often see servers with sub 200 pings. If you ignore the ping and just look for your favorite map, you will probably have a crappy game.

If you have a good basic connection to the server, the thing that will mess up your game is too much visible activity. This is a characteristic of the number of players, the openness of the level, and the weapons in use.

Don't play on madhouse levels with tons of players. None of the normal Q3 maps were really designed for more than eight players, and many were only designed for four.

Don't play in the wide open maps unless there are only a couple other players. Four very active players in a wide open area are enough to bog down a modem connection.

I just implemented "sv_minPing" / "sv_maxPing" options so servers can restrict themselves to only low ping or high ping players. This is done based on the ping of the challenge response packet, rather than any in-game pings. There are a few issues with that - a LPB may occasionally get into a HPB server if they happen to get a network hiccup at just the right time, and the number used as a gate will be closer to the number shown in the server list, rather than the number seen in gameplay. I would reccomend "sv_minPing 200" as a reasonable breakpoint.

11.6 Dec 30, 1999

Several people have mentioned an existing anti-cheat QW proxy that should also be applicable to modified versions:

`http://www.students.tut.fi/~zibbo/qizmo/`