Títol: Sistema automàtic de bots pel Quake 3

Volum: 1

Alumne: Jaume Palència Fernández

Director: Javier Verdú Mulà

Ponent: Manuel Alejandro Pajuelo González

Departament: Arquitectura de Computadors

Data: 27 d'octubre de 2008

DADES DEL PROJECTE

Títol del Projecte: Sistema automàtic de bots pel Quake 3

Nom de l'estudiant: Jaume Palència Fernández
Titulació: Enginyeria en Informàtica
Crèdits: 37,5
Director: Javier Verdú Mulà
Ponent: Manuel Alejandro Pajuelo González
Departament: Arquitectura de Computadors

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Jordi Guitart Fernández

Vocal: Antonio Cañabate Carmona

Secretari: Manuel Alejandro Pajuelo González

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

A mons *pares*, ma *germana* i la meva *família* que m'han recolzat en tot moment durant la realització d'aquest projecte.

Als *peores*, especialment en *Xavi*, per donar-me la motivació que m'ha calgut en els moments més difícils i que m'ha ajudat a tirar endavant.

A l'*Eva*, per ser com és, per donar-me els ànims i el recolzament quan ho he necessitat i, en general, per ser-hi sempre.

A en *Xavi* i l'*Àlex* per la seva comprensió, atenció i dedicació no només al projecte, si no també a la meva persona.

Gràcies a tots i continueu sent així

Índex

Capítol 1

Introducció

En els últims temps el món dels videojocs ha patit una forta evolució i expansió, degut, en gran part, a l'entrada de la tecnologia al quotidià de l'oci de al societat. Un fet important ha estat l'adequació d'internet a tots els sectors de la població. Aquest augment de l'interès pels videojocs ha fet aparèixer una gran competència entre les empreses desenvolupadores, cosa que ha portat a obtenir productes d'una gran complexitat.

1.1 Introducció

Actualment, els videojocs ocupen un sector molt important dins del món de l'oci, tal com podem observar a la figura 1.1. És l'industria que més beneficis genera, per sobre del cinema i la música. Això ha fet que siguin molts els que han passat a mirar els videojocs amb uns altres ulls. Dins el món de la informàtica i les comunicacions també trobem un important "projecte" comunitari que mou milions de persones, aquestes però, paradoxalment, sense un afany de lucre. És la comunitat del software lliure.

Font aDeSe

fig 1.1: Estimació del consum d'oci

L'augment de les xifres que mou el món de la informàtica en general, i dels videojocs en particular, ha sigut en gran part per la diversificació del públic consumidor. Avui en dia, persones de diferent edat, sexe i/o situació econòmica, poden compartir videojocs amb total normalitat. Cal dir que Internet ha sigut un important responsable d'aquesta diversificació degut a la seva facilitat d'accés.

El creixement del sector dels videojocs ha fet que la creativitat i la innovació siguin un punt clau dins l'èxit dins el sector degut a la gran competitivitat que hi existeix. Així doncs, tant els suports com els dissenys dels videojocs s'han vist obligats a patir una significant evolució. Destinant gran quantitat de diners i un gran esforç de personal a tal efecte.

Actualment els videojocs han optat per aconseguir un realisme extrem per al qual és necessari un maquinari d'altes prestacions. Per aconseguir tal maquinari és important la investigació tecnològica, i amb una especial importància la simulació de sistemes complexes. Però en arribar en aquest punt ens trobem que no existeixen jocs de proves per al sector dels videojocs.

Així doncs, vist les necessitats dins el sector dels videojocs, i recolzats com hem vist per un potencial de personal com es el cas del software lliure, veiem que existeix la necessitat d'una eina de *benchmark* que ens permeti la simulació del maquinari mentre està executant un videojoc.

1.2 El Videojocs

Han estat molts els videojocs que han intentat buscar realisme a les seves sagues. Sempre han procurat que cadascuna de les seves entregues fes posar al usuari cada cop més dins la pell del propi protagonista. A continuació es descriuen els més rellevants, es pot observar algunes imatges d'ells a les figures 1.2 – 1.6.

El primer videojoc de l'historia fou creat com a curiositat científica per

William Nighinbottham l'any 1958, consistia en una simulació d'un joc de tennis de taula en un oscil·loscopi.

El primer videojoc que situava el jugador dins un entorn de 3 dimensions fou el 3D Monster Maze, creat l'any 1981. En aquest videojoc el jugador havia d'escapar d'un laberint on hi residia un monstre.

Més endavant aparegué el Wolfenstein 3D, en aquest videojoc ja hi apareixien textures i el jugador estava proveït d'armes per eliminar els nazis que residien dins el castell per on el jugador es movia.

El següent videojoc en afegir un component nou dins el mercat va ser el Doom, que afegia la possibilitat de moure's amunt i avall així com moure la vista en totes direccions.

Més endavant aparegué Quake, videojoc on tot l'entorn estava elaborat dins una representació 3D, des de el propi jugador fins als monstres. Sens dubte una de les sagues més importants dins el món dels videojocs.

fig 1.2: Videojoc de William Nighinbottham

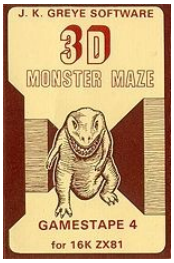

fig 1.3: 3D Monster Maze

fig 1.4: Wolfenstein 3D

fig 1.5: Doom I

fig 1.6: Quake I

1.3 Id Software

John Carmack i John Romero es van conèixer treballant junts a l'empresa Softdisk. Allà ambdós desenvolupaven videojocs pel magazine en disquet, anomenat per ells mateixos *magazzete*. Un cop fora i ja sota el nom de Id Software, en podem veure el logo a la figura 1.7, començaren a dissenyar jocs per Apogee Software. La majoria d'aquests videojocs ho feren sota una llicència shareware.

fig 1.7: Logo de Id Software

Tot i que tant John Carmack com John Romero eren uns experts programadors, des d'un inici el sistema de renderització va caure sota la increïble capacitat creativa de John Carmack, mentre que John Romero participava més en la creació conceptual i testeig del videojoc.

Degut a la profunda creença de John Carmack en el software lliure i en la compartició de codi, tot *motor* dels jocs creats per ell han estat llicenciats sota la *General Public License* (GPL) al cap de 5 anys del seu llançament.

1.4 El software lliure

La idea del software lliure resideix en que si els propis programadors i usuaris poden llegir, modificar i redistribuir el codi font d'un programa, aquest evoluciona, es desenvolupa i al cap i a la fi millora.

Per a que un software es consideri lliure ha de poder assegurar les 4 següents llibertats als seus usuaris:

- La llibertat d'utilitzar el programa.
- La llibertat d'estudiar com funciona el programa, podent-lo modificar i adaptar a les seves necessitats.
- La llibertat de distribuir-lo.
- La llibertat de millorar el programa i fer públiques les millores.

1.5 Quake 3

Corria l'any 1999 quan Id Software treia la seva tercera entrega de l'exitosa saga del joc Quake, el logo del qual es pot observar a la figura 1.8. En aquesta

fig 1.8: Imatge del Quake 3

entrega l'equip de John Carmack i John Romero, cofundadors de Id Software, continuen sorprenent als fanàtics dels jocs amb diverses innovacions tant pel que fa a la jugabilitat com pels avenços extraordinaris en el camp de la renderització 3D.

1.5.1 Motor

El motor gràfic és l'encarregat de tractar amb els objectes dins el món virtual. En el cas de Quake 3 aquest utilitza models amb el format MD3. Aquest format utilitza moviments per vèrtex, en contra dels moviments per esquelets que eren els que s'utilitzaven llavors. Això va permetre una alta complexitat en els models facilitant així un major realisme, tal com es pot observar a la figura 1.9.

fig 1.9: Screenshoot del Quake 3

Un altre aspecte important d'aquest format es que els models estan dividits en tres parts lligades entre elles, normalment cap, tronc i cames. També és destacable la utilització de *shaders* (ombrejats) per definir imatges que permeten barrejar fins a 3 textures. A més el sistema està proveït d'un llenguatge d'alt nivell per definir aquests efectes.

1.5.2 Quake 3 i GPL

L'any 2005 Id Software allibera el codi del joc sota llicencia GPL permetent augmentar el desenvolupament del món del videojoc ja que permet a multitud de desenvolupadors aprendre del codi d'aquest i inclús utilitzar-lo per llençar les seves pròpies versions. L'única part del joc que no fou alliberada foren les textures i altres parts artístiques.

El fet de no disposar dels components artístics va fer aparèixer al cap de poc temps un projecte lliure per posar remei a aquesta qüestió, OpenArena. Gràcies a aquest projecte la comunitat té a la seva disposició un joc complet tant per a l'oci com per a la investigació.

A partir del motor alliberat del Quake 3 i la part gràfica de OpenArena la comunitat pot desenvolupar qualsevol funcionalitat que necessiti per millorar o variar la jugabilitat del joc, sempre hi quan aquesta variació sigui també lliure.

1.6 L'elecció de Quake 3

L'elecció del videojoc Quake 3 per a la realització d'aquest projecte, doncs, no és cap casualitat. La complexitat i qualitat del joc ens permeten exposar aquest benchmark a màquines actuals. El fet que sigui lliure ens permet oferir el nostre producte a la comunitat de desenvolupadors, contribuint a la formació d'aquests. El fet que sigui lliure també ajuda a la portabilitat a diferents arquitectures i això ens permet fer testejos amb diferents entorns fent encara més útil aquesta eina.

1.7 Motivació personal

Si hagués de buscar 3 paraules ràpides que haguessin de descriure la meva relació amb els ordinadors, sent sincer, hauria de dir Doom, Gentoo i Google. El primer, Doom, em va fer descobrir el potencial que tenia un ordinador, per crear móns complexos dins una simple pantalla al mateix temps que em feia descobrir l'excitació que produeix un joc realista. El segon, Gentoo, em va fer descobrir com milers de persones desinteressades podien crear un sistema per aprofitar al màxim el rendiment d'un ordinador. I l'últim, Google, el qual m'ha permès la relació amb milions de pàgines web d'on he pogut aprendre al majoria de coneixements que tinc gràcies a la gent que comparteix el que sap a través d'Internet. He crescut gràcies, en part, al món virtual, fig 1.10.

Fig 1.10: Dibuix creat per Tipatat Chennavasin
(Nascut per trossejar [matar])

Així que al moment d'escollir el projecte final de carrera no vaig tenir cap dubte en seleccionar la modificació del Quake 3. Ja que sabia que el podria

desenvolupar sense cap mena de problema sobre el meu sistema operatiu. I que, gràcies un cop més a Internet, em permetria compartir els meus coneixements amb els de tants d'altres que han treballat en projectes semblants.

1.8 Estructura de la memòria

L'estructura de la memòria és la següent:

- En el capítol 2 es defineixen els objectius d'aquest projecte, i els requisits necessaris per assolir-los.

- En el capítol 3 es descriu les característiques que té la modificació que s'ha realitzat del Quake 3, que s'ha batejat amb el nom de PFC_Quake.

- En el capítol 4 es realitza un anàlisi del funcionament del Quake 3. S'estudien els subsistemes en que es troben dividides les funcionalitats des d'un punt de vista teòric.

- En el capítol 5 es realitza un estudi en profunditat sobre el funcionament tècnic del videojoc. S'analitza la distribució dels diferents fitxers del codi així com del fluxe d'execució. També es realitza una explicació en detall del funcionament de l'estructura de dades més utilitzada, la cvar.

- En el capítol 6 s'explica les modificacions realitzades al codi font per satisfer els objectius plantejats anteriorment.

- En el capítol 7 es detalla l'entorn de programació utilitzat per desenvolupar el projecte, així com el sistema operatiu i les aplicacions utilitzades per entendre i

desenvolupar les modificacions.

- En el capítol 8 es realitza l'avaluació i el testeig del PFC_Quake.

- En el capítol 9 s'explica quina ha estat la planificació del projecte, i quin ha estat el cost i el benefici potencial que aporta el projecte.

- Per acabar, en el capítol 10 es descriuen les conclusions obtingudes al acabar el projecte així com un breu resum sobre les tasques aconseguides amb aquest.

Capítol 2

Objectius del projecte

Aquest projecte té per objectiu principal modificar i adaptar les funcionalitats del Quake 3 per permetre crear partides del joc entre jugadors controlats per la màquina sense la necessitat de cap jugador humà. En un segon pla també s'afegirà un control per crear partides preconfigurades i una recollida d'informació de la pròpia partida.

2.1 Objectius principals

Aquest projecte té per objectiu principal oferir una eina de benchmark als mercat de desenvolupament de hardware destinat a les aplicacions de videojocs. Per aconseguir aquest objectiu s'ha escollit seleccionar un videojoc ja existent llicenciat sota la GPL i modificar-lo per a que ens serveixi a tal efecte. El videojoc seleccionat ha estat el Quake 3, com ja s'ha explicat a la secció anterior.

Per aconseguir aquest producte s'han estructurat el sistema d'objectius de la següent manera:

Objectiu 1: L'objectiu principal és modificar i adaptar les funcionalitats del Quake 3 per permetre crear partides del joc entre jugadors controlats per la màquina sense la necessitat de cap jugador humà. S'haurà, doncs, de modificar l'estat inicial d'un jugador humà per a que sempre entri per defecte en mode espectadors, per a que pugui estar dins la partida sense intervenir-hi.

Objectiu 2: Facilitar la creació de partides preconfigurades a partir d'un arxiu de configuració. Això ens permetrà crear partides d'una manera sistemàtica i ràpida, sense la necessitat de parametritzar-les a través dels menús.

***Objectiu* 3:** Adaptar el sistema de recollida d'informació de la partida, que ens permetrà comprovar el seu correcte funcionament. Es desenvoluparà el sistema d'extracció d'informació per a que en un futur es puguin treure resultats de les diferents configuracions executades, i es pugui realitzar un estudi. Com a objectiu

actual es planteja obtenir els *frames per segon* (FPS) així com la posició relativa dels successos. Amb aquests paràmetres es podrà estudiar el rendiment de la targeta gràfica de la màquina.

2.2 Objectius previs

Per tal d'aconseguir els objectius citats en primer lloc cal entendre profundament el funcionament tècnic del joc, és a dir realitzar una tasca d'enginyeria inversa per descobrir com han estats programades les funcionalitats existents. Aquest objectiu serà el més complex degut a la manca de documentació dels mecanismes interns del joc.

2.3 Requisits

Per poder aconseguir els objectius citats als apartats anteriors i que l'eina de benchmark sigui adequada a la intenció per a que ha estat creada el projecte ha de complir els següents requisits:

Requisit 1: Les modificacions al codi existent han de ser clares i simples, per poder incorporar les futures modificacions al codi del propi videojoc.

Requisit 2: Les eines utilitzades per al desenvolupament del projecte han de ser estàndards, lliures i multiplataforma. L'ús d'aquestes eines faran que el benchmark es pugui executar sobre qualsevol plataforma facilitant la portabilitat a

tots els sistemes existents.

Requisit 3: La utilització de l'eina de proves, així com l'extracció d'informació ha de ser clara i senzilla ja que ens interessa que l'etapa de proves de la investigació de hardware ocupi el mínim temps possible.

Capítol 3

Descripció de PFC_Quake

L'eina de benchmark PFC_Quake redueix el cicle d'investigació de hardware per a videojocs ja que elimina la necessitat de desenvolupar el joc de proves per testejar el funcionament dels nous dissenys de hardware. Així doncs, l'eina és d'un gran interès tant el sector empresarial que investiga noves tecnologies com per la formació de desenvolupadors.

3.1 Descripció del benchmark PFC_Quake

PFC_Quake és una modificació lliure del videojoc Quake 3 que el converteix en una eina de benchmark amb la intenció que serveixi d'eina per a la investigació del hardware durant l'execució d'un videojoc. Aquesta modificació s'ha realitzat a partir del codi de ioquake3, un projecte de software lliure que té per objectiu la revisió i actualització del codi del Quake 3.

El públic al qual està destinada aquesta eina és a les empreses i/o investigadors interessats en testejar el funcionament de les seves simulacions en un entorn d'execució real i actual. Amb aquesta eina l'usuari estalvia la creació d'un nou videojoc per testejar els nous avenços.

Cal destacar, també, l'ús formatiu d'aquesta eina. Ja que com a software lliure que és permet a futurs desenvolupadors i/o investigadors aprendre del seu funcionament.

PFC_Quake permet l'execució d'entorns reals de partides del videojoc Quake 3 sense la necessitat de jugadors humans, cosa que permet testejar paràmetres externs del joc fàcilment. Aquest també ens permet la creació de partides preconfigurades cosa que facilita l'execució automàtica i regular de partides entre els bots.

3.2 Què ofereix PFC_Quake al sector d'investigació empresarial

El sector d'investigació de hardware destinat a videojocs realitza una etapa de prova sobre els dissenys que els investigadors proposen. Aquesta etapa es desenvolupa sobre simulacions dels dissenys, però alhora necessiten poder simular entorn de partides de videojocs. El procés de desenvolupament d'aquestes proves encareix i retarda la investigació.

Una possible solució és utilitzar les versions normals de videojocs actuals del mercat, però aquests, al necessitar la interacció amb algun jugador humà, resulten ineficients ja que necessitem un equip exclusivament destinat a aquest efecte.

Una altra solució és desenvolupar o adaptar un producte del mercat a les necessitats de testeig. Aquesta opció necessita de la negociació amb el sector mercantil a més d'un equip destinat a desenvolupar tal modificació.

PFC_Quake opta per aquesta segona opció, escollint un producte de software lliure, que ens redueix a zero la negociació i modificant-lo per que serveixi com a eina de testeig. Així doncs, PFC_Quake és una solució a l'etapa de prova que el sector d'investigació de hardware necessita un cop simulats els nous dissenys que potencialment implementaran.

3.3 Què ofereix PFC_Quake a la comunitat de desenvolupadors

PFC_Quake també neix amb la intenció de retornar a la comunitat el que aquesta ha proveït. Gràcies a aquesta eina els projectes d'investigació i aprenentatge de la comunitat de desenvolupadors lliures es veuen recompensats amb una nova eina de benchmark d'un videojoc complet.

Aquest joc de proves permet el desenvolupament de multitud de projectes a l'entorn del sector dels videojocs i obre les portes a la possibilitat de la creació de nous projectes.

Capítol 4

Anàlisi del funcionament del Quake 3

La necessitat de dividir el funcionament en parts més reduïdes que puguin funcionar independentment ajuda molt al desenvolupament i testeig del software. El Quake 3 no és un excepció i separa el cos del codi en dos gran mòduls destinats a emmagatzemar i controlar el món per una banda i a reproduir-lo i mostrar-lo a un jugador concret per l'altre.

4.1 Subsistemes

El Quake 3 es troba dividit en dos parts bàsiques, el client i el servidor, i un sistema de xarxa que permet la comunicació entre els dos primers, tal com es pot observar a la figura 4.1. El client bàsicament està dedicat a la interfície d'usuari, és a dir, dibuixa la realitat 3D del videojoc i captura les ordres enviades a través del teclat, ratolí i qualsevol perifèric utilitzat. La part servidor és la responsable de moure els personatges a través del món mitjançant les ordres rebudes, detectar col·lisions, processar la intel·ligència dels *bots*, determinar impactes, morts i mantenir tot l'estat del món virtual.

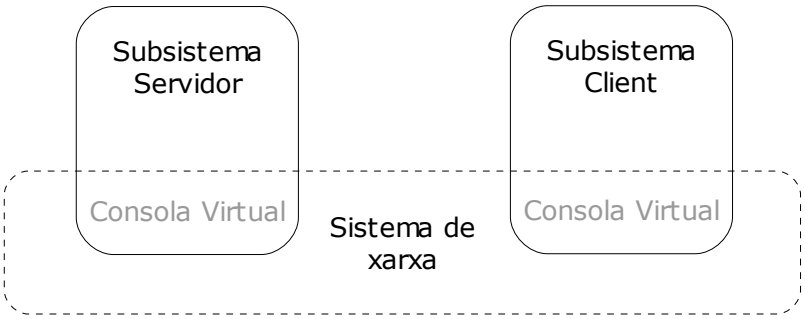

fig 4.1: Esquema conceptual dels subsistemes

4.1.1 Subsistema client

El subsistema client s'encarrega de la part del videojoc necessària per a que un jugador pugui participar en una partida. En un principi el client necessita crear una consola virtual que serà l'encarregada de rebre, organitzar i executar les accions que el jugador realitzi.

Un cop s'estableix la via de comunicació és posarà en marxa el sistema encarregat de capturar la interacció del jugador amb els dispositius d'entrada/sortida. És necessari inicialitzar tant la visualització d'imatges, com capturar les senyals del teclat i el ratolí.

El subsistema client també és el responsable de la gestió de menús per a poder configurar i/o seleccionar la partida a la qual el jugador es vol afegir. A partir d'aquests menús també s'enviarà la comunicació necessària al subsistema servidor.

Un cop el sistema de menús ens ha permès crear la partida el client ha de mantenir la informació que rep del servidor en coherència amb el que el motor de *rendering* mostra al usuari a través dels perifèrics. Així doncs el client rep el món a dibuixar i un cop analitzat pel motor gràfic mostra al jugador la visió que aquest té del món virtual. També realitza el procediment invers, enviant al servidor els moviments i canvis al món que el jugador ocasiona.

Els bots també són subsistemes client tot i que les seves funcionalitats estan limitades i predeterminades dins el codi. Tot i això el seu funcionament és molt similar al d'un client d'un jugador humà.

4.1.2 Subsistema servidor

Un cop el jugador ha creat la partida desitjada el codi servidor manté tota la informació del món virtual i la modifica a mida que els clients li comuniquen les accions a emprendre. Cal destacar, dins el manteniment de la coherència del món virtual, els càlculs necessaris per reproduir la física que governa el sistema.

El subsistema servidor s'encarrega de la part del codi responsable de mantenir el món coherent i que les entitats evolucionin en conseqüència. Aquest té la seva pròpia consola virtual amb la qual els diferents subsistemes client es comuniquen.

Així doncs, el subsistema servidor conté les estructures destinades a controlar les sessions que els diferents clients, tant humans com bots, les puntuacions i estadístiques dels jugadors dins la partida, la validació dels jugadors i, en general, tota la informació necessària per que la partida tingui continuïtat en el temps.

Pel que fa a la part de coherència del món, el servidor és el responsable dels càlculs dels moviments dels jugadors, la situació i reproducció dels objectes variables dins el mapa, els combats i impactes de les armes utilitzades pels jugadors entre altres responsabilitats per a que el món es mantingui en coherència i gaudeixi d'una jugabilitat realista.

4.1.3 Parts comunes i interconnexió dels subsistemes

A l'inici de l'execució del videojoc el programa abstreu la capa de sistema operatiu per a cadascun dels possibles sistemes operatius, configurant la consola, la memòria, les crides a sistema, la gestió del sistema de fitxers i la capa d'interfícies. A continuació realitza les inicialitzacions necessàries, així com l'activació i inicialització dels subsistemes client i servidor.

La interconnexió entre els subsistemes es realitza a través del sistema de xarxa que permet la comunicació entre clients i servidor. Per cada consola virtual del subsistema client es crea una connexió amb el servidor amb el qual el client vol connectar. A través d'aquest port es transmetran tots els missatges necessaris pel funcionament del videojoc.

Capítol 5

Estructura interna del Quake 3

L'adequació entre l'estructura conceptual i l'estructura real d'un projecte de software sempre és veu compromesa degut a les limitacions que comporta treballar sobre una base de software genèrica. Dins el codi de Quake 3 aquest compromís es troba ben solucionat desglossant les responsabilitats de cadascun dels subsistemes i creant divisions a cavall entre funcionalitat i subsistema. Així doncs no ens ha d'estranyar trobar parts d'un subsistema mesclats amb d'altres per comoditat de programació al pertànyer a funcionalitats similars.

5.1 Implementació

L'estructura conceptual es troba implementada per tot un teixit de fitxers i carpetes que s'encarreguen de cadascuna de les funcionalitats assignades a cadascun dels subsistemes. Aquesta estructura de fitxers crea una subdivisió més clara de funcionalitats, segons la pertinença a una carpeta, però alhora menys definida. Ja que els fitxers presents dins carpeta poden pertànyer a subsistemes diferents; o inclús funcions en un mateix fitxer s'utilitzen en més d'un subsistema. Així doncs, l'esquema d'aquesta adaptació tal com es pot observar a la figura 5.1.

El conjunt del codi del Quake 3, en la seva versió revisada per ioquake3, conté un total de 11 carpetes destinades al propi videojoc (dins el codi en podem trobar 9 més utilitzades a mode de llibreries), amb un total de 374 fitxers, que això acaben sent 296.033 línies de codi.

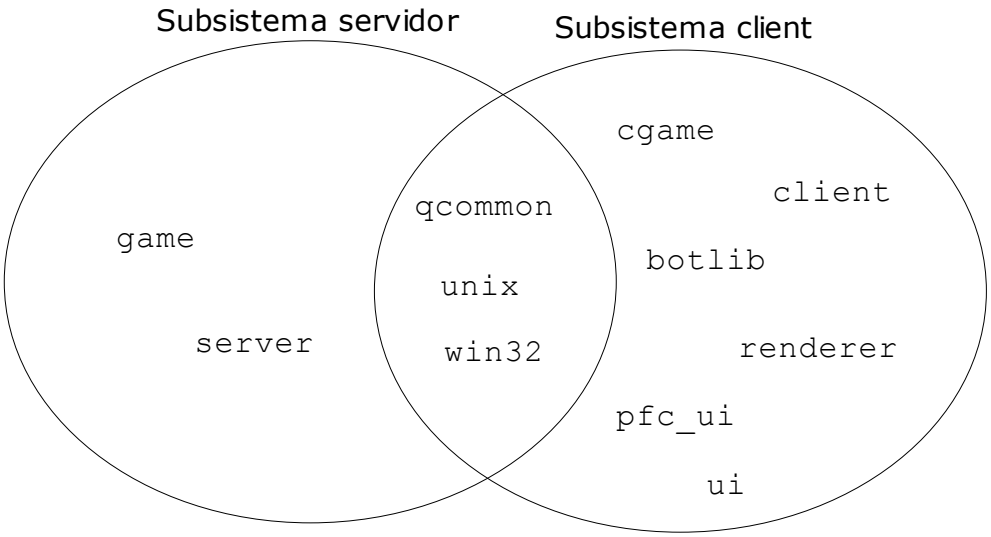

fig 5.1: Entrellaçat entre l'esquema conceptual i el sistema de fitxers

5.2 Estructura de fitxers del codi

A continuació s'especifica a nivell general les responsabilitats dels codis dins de cada carpeta de l'estructura de fitxers del codi del Quake 3.

5.2.1 Carpetes subsistema servidor

game: Aquest codi pertany a la part servidor i s'encarrega de crear la màquina virtual d'aquest. S'encarrega de la gestionar i interpretar la informació rebuda pel client. També és el responsable de tots els càlculs per mantenir l'estat del món virtual i enviar els canvis al client. *Conté 61 fitxers, amb 49.043 línies.*

server: Els codis continguts dins aquest apartat s'encarreguen de les funcionalitats relacionades amb mantenir el món en coherència, tant pels propis càlculs de l'evolució del món com de la interacció d'aquest amb els jugadors. *Conté 13 fitxers, amb 9.820 línies.*

5.2.2 Carpetes subsistema client

botlib: El codi dins aquesta part s'encarrega de tot el funcionament dels *bots* dins el joc. *Conté 64 fitxers, amb 36.193 línies.*

cgame: Aquesta secció és la part del client que s'encarrega crear la màquina virtual del client que recull la informació rebuda del servidor i s'encarrega d'interpretar-la. També s'encarrega de comunicar al motor de rendering les *entitats* a dibuixar. *Conté 31 fitxers, amb 28.163 línies.*

client: Aquesta part del client crea el propi executable del videojoc. S'encarrega d'inicialitzar les dades necessàries per a la creació de la partida així

com recollir la informació introduïda a partir dels perifèrics. *Conté 31 fitxers, amb 21.433 línies.*

pfc_ui & ui: Dins aquest apartat resideix tot el codi encarregat de gestionar els menús. *Conté 52 fitxers, amb 26.965 línies i 13 fitxers, amb 16.330 línies, respectivament.*

renderer: *Aquesta part del client es pròpiament el motor gràfic, s'encarrega de dibuixar el món que un client pot veure. Conté 29 fitxers, amb 25.833 línies.*

5.2.3 Carpetes comunes dels subsistemes

qcommon: Aquest part s'encarrega realitzar totes les inicialitzacions un cop abstret el sistema operatiu, també inicialitza els subsistemes client i servidor. *Conté 35 fitxers, amb 36.130 línies.*

unix: Dins aquesta secció es troba el codi específic pels sistemes operatius Unix, Linux i Mac dins la part de client i servidor. *Conté 25 fitxers, amb 16.898 línies.*

win32: En aquesta secció hi ha el codi específic per als sistemes que pertanyen a windows. *Conté 20 fitxers, amb 29.225 línies.*

5.3 Estructura i funcionament intern

Tot i aquesta mescla de funcionalitats distribuïdes per les diferents carpetes, analitzant exhaustivament el codi es pot entendre els fluxos de codi que segueix cadascun dels subsistemes. Ens trobem primerament amb unes inicialitzacions que s'encarreguen de preparar el sistema pel correcte funcionament del videojoc i a continuació es passa a crear els subsistemes necessaris.

5.3.1 Inicialitzacions

A l'inici d'una execució estàndard es passa el control del flux a la comanda *Com_Init()*, dins common.c que pertany a *qcommon,* que s'encarrega d'inicialitzar la línia de comandes, inicialitza els *buffers* de memòria interns, carga les cvars, i inicialitza el subsistema servidor, el client i el canal de xarxa.

A continuació s'executa infinitament la funció Com_Frame(),també dins de common.c, que serà l'encarregada de calcular cadascun dels *frames*, altrament dit batecs, del videojoc. Dins aquesta funció es passa el control al servidor mitjançant la funció SV_Frame(), descrita dins el fitxer sv_main.c de server, i si no es tracta d'un *servidor dedicat* es crida a CL_Frame(), descrita dins el fitxer cl_main.c de client, que passarà el control al subsistema client. Aquests passos es succeeixen infinitament.

Durant la funció Com_Frame() també es tracta la comunicació entre els subsistemes client i servidor. Aquesta comunicació consisteix en seqüències de paquets UDP de client a servidor i viceversa.

5.3.2 Subsistema client

El subsistema client s'inicialitza durant la funció Com_Init(), on es guarda espai per les variables necessàries i es crea la màquina virtual de client.

Després i durant tota l'execució del videojoc, és a dir cada cop que es crida a la funció Com_Frame(), s'executa la funció CL_Frame(). Aquesta funció s'encarrega de calcular tota la informació necessària dins cada *frame*.

Si el client encara no s'ha connectat a cap servidor aquest crida el dibuixat del menú a través de la seva màquina virtual. A continuació comprova si hem perdut la connexió amb el servidor a través de la funció CL_CheckTimeout(), i envia al servidor les accions que té intenció de realitzar a través de CL_SendCmd(). Per acabar sincronitza el vídeo i la música a través de SCR_UpdateScreen() i S_Update(). La funció CL_CheckTimeout() es troba definida dins el fitxer cl_main.c, CL_SendCmd() dins cl_input.c, SCR_UpdateScreen() dins cl_scrn.c i S_Update() a snd_main.c, tots els fitxers es troben dins la carpeta client.

El dibuixat del menú, sempre i quan el client no s'estigui connectat a un servidor, s'inicialitza amb la funció UI_MainMenu(), definida dins el fitxer ui_menu.c de la carpeta pfc_ui, un cop comprovat l'autenticitat de la clau de CD. Aquesta funció inicialitza la memòria necessària per la creació del menú i a continuació el dibuixa. Segons les seleccions dins el menú es criden les funcions necessàries i s'assignen els valors escollits a les variables pertinents i s'omple el *buffer* amb les comandes necessàries que en acabat s'envien a la màquina virtual del servidor.

5.3.3 Subsistema servidor

El subsistema servidor també s'inicialitza durant la funció Com_Init(), on també es guarda espai per les variables necessàries i es crea la màquina virtual del servidor.

Tal com hem vist amb el subsistema client, el subsistema servidor també disposa d'una funció, SV_Frame(), que s'executa durant cada Com_Frame(). Aquesta s'encarrega de tots els processos requerits per a cada acció dels clients així com de mantenir els processos que s'estan executant dins la seva màquina virtual.

A través del paràmetre msec de la funció SV_Frame(), el servidor gestiona el temps de servidor permetent el sincronisme entre clients i el propi servidor. A l'inici de la funció es realitzen tots aquests càlculs. A continuació es passa el control al bots per a que realitzin les seves accions dins el frame actual a través de la funció SV_BotFrame(). Després comprova si algun client ha perdut la connexió amb la funció SV_CheckTimeouts() i envia les modificacions del món virtual als clients amb SV_SendClientMessages(). Per acabar, i si el servidor està obert a les connexions a través d'Internet, envia al servidor master el batec global del servidor així com informació sobre l'estat del servidor. La funció SV_BotFrame() es defineix dins el fitxer sv_bot.c, SV_CheckTimeouts() dins sv_main.c i SV_SendClientMessages() a sv_snapshot.c, tots els fitxers es troben dins la carpeta server.

5.4 Estructura de dades CVAR

El Quake 3 utilitza, com ja havia fet en els seus predecessors, l'estructura cvars, acrònim de console var. Aquesta estructura permet la comunicació entre els subsistemes, mitjançant les consoles virtuals, i també entre l'usuari i els subsistemes.

En un principi pot semblar que és una estructura senzilla ja que simplement s'utilitza per emmagatzemar valors, però indagant dins el codi es pot observar que no és així. L'estructura i les seves respectives funcions estan dissenyades pensant en l'optimització del seu ús. Així doncs, les funcions mantenen diversos valors calculats per la seva ràpida utilització. En el moment d'inicialització de la variable el codi intenta fer un *parsing* de l'string tant a enter com a número en coma flotant.

Les variables també poden tenir un conjunt de *flags* que faciliten la posterior utilització d'aquestes, així com funcionalitats per reinicialitzar el seu contingut a un valor per defecte. Els diferents *flags*, bàsicament, ens permeten controlar en quines etapes del codi es permet modificar el valor de les variables. Així doncs, segons quines *flags* estan activades el valor de la variable es podrà modificar només a l'inici de l'execució (és a dir, dins del codi o en la pròpia línia de comanda), o només si els *cheats* estan desactivats, o que no permetin resetejar el valor, entre d'altres funcionalitats.

Cadascuna de les variables cvar tenen una comanda de consola associada de nom igual al nom de la cvar que ens permet assignar un valor a la variable, d'aquí el nom cvar. Aquesta comanda també imprimeix el valor després de la crida,

així com el valor per defecte de la pròpia variable.

5.4.1 Variables de tipus CVAR importants

Son moltes les variables de tipus cvar que tenen una gran importància dins al funcionament del videojoc. A continuació es passarà a descriure el comportament d'algunes d'aquestes variables que prenen rellevància degut a les modificacions que s'han realitzat dins el marc del projecte PFC_Quake.

La variable **fs_game** s'utilitza per definir quina és la carpeta on el videojoc ha de buscar les maquines virtuals. Gràcies a aquesta variable l'usuari pot mantenir diferents versions i modificacions del Quake 3 convivint en el mateix sistema de fitxers i, per tant, compartint la informació.

La variable **fs_basegame** s'utilitza per definir la carpeta on el videojoc busca el fitxers pk3. Així doncs, tant la versió original del Quake 3 com totes les seves modificacions es poden proveir dels dissenys utilitzats dins el joc original.

Un altre joc de variables importants son la **sv_pure**, la **vm_game** i la **vm_ui**, aquestes tres variables defineixen si el videojoc utilitza les consoles virtuals creades dins els fitxers pk3 o utilitza les llibreries creades a partir del codi (dll, en cas de windows, o so, en cas de Mac o Linux). Aquestes variable son realment útils durant el desenvolupament del videojoc, ja que permeten testejar els diferents subsistemes per separat.

Capítol 6

Modificacions de les funcionalitats

Per poder realitzar modificacions adequades és important utilitzar un mètode que ens asseguri el màxim benefici a les expectatives creades i, alhora, realitzi el mínim possible de modificacions a les funcionalitats. Així doncs, en primer lloc s'ha adequat l'estructura de compilació i nomenclatura al nou projecte. A continuació s'ha decidit estructurar les modificacions en dues etapes consistents en un procés d'anàlisi de les funcionalitats implicades i en la pròpia modificació del codi.

6.1 Modificacions prèvies

Com a primer pas, s'ha modificat l'estructura de fitxers i l'arxiu de Makefile, per adaptar el codi al nou projecte. S'ha renombrat el videojoc a pfc_quake, així com els arxius que fan referència a les funcionalitats de jugabilitat per fer-los independents de la part de rendering. Dins l'arxiu de Makefile

A continuació, s'ha importat el sistema de fitxers a l'IDE Code::Blocks per realitzar les modificacions i lectura de codi amb més senzillesa i eficiència.

6.2 Control d'inicialització d'usuaris

Per facilitar la jugabilitat, el Quake 3 sempre inicialitza els jugadors dins la partida, tant sigui en mode *deathmatch* o en mode per equips, inclús escollint ell un equip on jugar. Per aconseguir l'objectiu que ens plantegem és necessari que els jugadors en principi controlats per jugadors humans comencin en mode espectador, és a dir que no interfereixin en el joc dels *bots*. Per aconseguir tal efecte s'ha modificat la inicialització de la informació de sessió dels jugadors.

6.2.1 Procés de desenvolupament

En un primer moment, amb la finalitat de descobrir en quin punt se li assigna al jugador l'estat de joc, es va seguir la traça dins del codi.

Per començar, es centra l'atenció dins l'evolució de la interacció amb el

"menú" (codi que trobem dins la carpeta pfc_ui). La primera funció que s'executa és la UI_MainMenu, que s'encarrega de realitzar les inicialitzacions pertinents així com d'omplir l'estructura *s_main*, que conté tota la informació referent al menú.

Navegant pel sistema de *callbacks* es van creant els diferents menús que ens permeten configurar la partida. En aquesta traça es pot veure bàsicament dos camins que s'hauran d'analitzar, el mode *single player* i el *multiplayer*. Dins d'aquestes traces de crides s'observen diversos punts que afecten la inicialització de jugadors, l'objectiu és descobrir quin és l'adequat.

Basant-se en el fet que la traçada de mode *multiplayer* és el cas més genèric s'opta per analitzar-lo en primer lloc; tot i que més endavant s'observarà que analitzant el mode *single player* en primer lloc la investigació hagués resultat més directa.

Dins el fluxe de creació d'una partida multiplayer, observem que la interfície de menús ens permet escollir el número de jugadors humans i de bots. També ens facilita assignar l'equip al que pertany cada jugador en cas de ser una modalitat de joc per equips. Així doncs, ja que el que busquem és inicialitzar els jugadors humans en estat espectador, és en aquest punt on centrarem les primeres aproximacions al codi.

6.2.2 Anàlisi del funcionament

Donant un primer cop d'ull al codi de configuració dels menús s'observa que aquest omple l'estructura s_serveroptions. El camp que s'analitza en primer lloc és el playerTeam, on sembla es pugui escollir l'equip que se li assigna a cada jugador.

Analitzant més profundament aquesta estructura observarem que no està preparada per inicialitzar un jugador en mode espectador, ja que l'ordre que utilitza per aplicar l'equip només ens permet escollir un dels equips. A més, si tornéssim a definir la funció no ens seria vàlida per els altres modes de joc.

Al adonar-se de la impossibilitat de realitzar la modificació en aquest punt, donada la poca adequació del procés de configuració a les nostres necessitats, s'opta per analitzar la funcionalitat de connexió de client dins el codi servidor. Amb això s'intentarà realitzar la modificació en un punt més proper a la creació o inici de la partida. Així doncs es deixa de banda la funcionalitat menú i es passarà a analitzar la carpeta *game*.

Analitzant l'arxiu *g_main* dins la carpeta *game* s'observa la funció *vmMain* que és la pròpia consola virtual del subsistema servidor. Aquesta es qui s'encarrega de capturar les funcionalitats entre client i servidor. Es pot veure l'esquema d'aquesta funció al codi 6.1.

Una de les funcionalitats bàsiques que es pot observar és la d'inicialització de partida i la de connexió d'un client, G_InitGame i ClientConnect respectivament. S'opta, en primer lloc, en valorar la possibilitat de realitzar la modificació dins el procés d'inicialització, és a dir dins la funció G_InitGame. En aquesta, s'inicialitza les dades i variables necessàries per la creació de la partida, així com certes inicialitzacions de les entitats dins el món. Tot i això, no inicialitza ni modifica les dades referents a clients, per tant es decideix avançar analitzant la funcionalitat de connexió de client.

```
intptr_t vmMain( int command, int arg0, [...]
int arg11 ){
    switch( command ){
    case GAME_INIT:
        G_InitGame( arg0, arg1, arg2 );
        return 0;
    case GAME_SHUTDOWN: [...]
    case GAME_CLIENT_CONNECT:
        return (intptr_t)ClientConnect([...]);

    case GAME_CLIENT_THINK: [...]
    case GAME_CLIENT_USERINFO_CHANGED[...]
    case GAME_CLIENT_DISCONNECT: [...]
    case GAME_CLIENT_BEGIN: [...]
    case GAME_CLIENT_COMMAND: [...]
    case GAME_RUN_FRAME: [...]
    case GAME_CONSOLE_COMMAND: [...]
    case GAME_START_FRAME: [...]
    }

    return -1;
}
```

Codi 6.1: funció vmMain de g_main

El procés d'inicialització d'un client a un servidor comença per la funció ClientConnect. Aquesta funció es crida tant a l'inici de la connexió com en cada canvi de mapa, i es crida tant si el servidor l'ha creat el propi usuari o es connecta a un servidor d'internet. Els bots també utilitzen aquesta funció.

Si el client connectat és el primer cop que entra al joc o hi ha hagut un canvi de modalitat de joc, llavors s'inicialitza la informació de sessió cridant a la funció G_InitSessionData dins l'arxiu g_session. Dins aquesta funció és on s'assigna al jugador a un estat de joc, actiu o espectador.

6.2.3 Explicació de les modificacions finals

Així doncs s'ha modificat aquesta funció per a que el jugador que es connecta a una partida o en el moment que hi ha un canvi de modalitat de joc se li

assigni automàticament l'estat d'espectador, fent així que per defecte els *bots* puguin jugar sense presència humana.

Aquesta modificació s'ha realitzat creant una variable init_session de tipus cvar que, tal com s'ha explicat anteriorment, ens permetrà escollir entre l'inici dissenyat per Id Software i l'inici que interessa a aquest projecte. D'aquesta manera no es desaprofita el codi elaborat per el Quake 3 i podrem en qualsevol moment escollir inicialitzar els jugadors de la manera convencional.

Cada cop que es crida la funció G_InitSessionData el nou codi observa el valor dins la variable init_session. Si aquesta té el valor per defecte "", s'assigna al jugador l'estat espectador. Si per contra el valor de la variable ha estat modificat a un altre valor, l'inici serà el convencional, assignant el jugador a un equip concret. A continuació es mostra part del codi d'aquesta funció, el codi 6.2.

```
Void G_InitSessionData( gclient_t *client, char
*userinfo){
    clientSession_t *sess;
    const char *value;
    sess = &client->sess;

    cvar_t *is = Cvar_Get("init_session", CVAR__INIT);
    char i_sess[1024];
    strcat(i_sess,if->string);
    if( !strcmp(i_sess,""){
        [...]
    else{
        sess->sessionTeam = TEAM_SPECTATOR;
    }
[...]
    G_WriteClientSessionData( client );
}
```

Codi 6.2: funció G_InitSessionData de g_session

6.3 Configuració de partides a través d'arxiu

El subsistema servidor ens proveeix d'un sistema de comandes per a poder configurar les partides a través d'aquestes comandes. No obstant, no està ideat per poder configurar partides abans de l'execució del joc. Així doncs s'ha creat un sistema amb el qual a partir d'un arxiu es poden enviar les comandes per configurar el joc en forma de bateria de comandes.

6.3.1 Procés de desenvolupament

L'elecció del punt d'execució de l'arxiu que es vol passar per configurar la partida és de vital importància, no només per que les partides s'executin correctament, si no també per a que quedi un codi amb sentit, ordre i fàcilment comprensible.

Per aconseguir aquest objectiu s'escull que l'execució del fitxer es produeixi just després de la creació del sistema de comandes ja que així es podrà utilitzar la metodologia de comandes ja implementada.

6.3.2 Anàlisi del funcionament

Analitzant el sistema d'inicialitzacions comunes dins la carpeta common, s'observa que el fitxer q_common és l'encarregat d'inicialitzar el sistema de comandes a través de la funció Com_Init.

```
void Com_Init( char *commandLine ){ [...]
    Sys_init();
    NetchanInit( [...] );
    VM_Init();
    SV_Init();
[...]}
```

Codi 6.3: funció Com_Init de q_common

Al final d'aquesta funció tots els subsistemes ja estan inicialitzats i preparats per al seu funcionament, així doncs, serà en aquest punt on es modificarà el codi. Al codi 6.3 es pot observar les diferents funcions d'inicialització dins Com_Init.

6.3.3 Explicació de les modificacions finals

Com hem explicat anteriorment, l'ús de les variables cvar ens permeten passar paràmetres entre els subsistemes. Així doncs, per poder escollir a cada execució quin és l'arxiu que definirà com serà la partida, passarem a través de la crida de l'executable el nom del fitxer assignant-lo a una variable. En aquest cas s'ha decidit anomenar-la game_file.

S'ha decidit inicialitzar aquesta variable a valor nul, d'aquesta manera si es decideix no assignar un valor es podrà executar el videojoc de la forma convencional, és a dir a través del sistema de menú.

Per tant, al final de la funció Com_Init es consulta la variable game_file, i en cas de tindre un valor diferent a nul, s'executarà les comandes descrites dins el fitxer de configuració amb nom igual a game_file. Les modificacions realitzades dins aquesta funció es poden observar dins el codi 6.4.

```
void Com_Init( char *commandLine ){
    [...]
    cvar_t *gf = Cvar_Get ("game_file","",CVAR_INIT);
    if( strcmp(gf->string,"")){
        char exec[1024];
        strcat(exec,"exec ");
        strcat(exec, gf->string);
        strcat(exec,"\n");
        Cbuf_AddText(exec);
    }
}
```

Codi 6.4: funció Com_Init de q_common

6.3.4 Exemple d'arxiu de configuració

En primer lloc s'ha de configurar les característiques de la partida a través de les variables *cvar* que la defineixen (veure apèndix D.6). A continuació, utilitzant la comanda *map* s'ha de carregar el mapa desitjat. Per finalitzar es carreguen el bots que es volen que participin de la partida amb la comanda *addbot*. Podem observar un exemple al codi 6.5.

```
g_gametype 2
map kaos
addbot Sarge 3
addbot Sarge 3
addbot Sarge 3
```

Codi 6.5: Exemple d'arxiu de configuració

6.4 Extracció d'informació durant l'execució

El Quake 3 proveeix un sistema de *log* que ens permet descobrir els successos que han passat durant el joc, aquest emmagatzema tota la informació referent al servidor i/o client independentment.

6.4.1 Procés de desenvolupament

El funcionament del sistema de logs del Quake 3 és realment senzill. Aquest simplement consisteix en extreure per la *sortida estàndard* del programa missatges amb la informació del fet que acaba de succeir. Tot aquest funcionament es troba desenvolupat dins el fitxer cg_event a la carpeta cgame, aquest fitxer s'encarrega de tot el tractament dels successos que rep del servidor.

6.4.2 Anàlisi del funcionament

El sistema de logs del Quake 3 extreu informació de la partida a mida que aquesta es va processant dins el codi de captura dels events. Cada frame del client, aquest comprova quin és l'últim event succeït, i en el cas que encara no s'hagi processat el codi l'analitza i realitza les modificacions necessàries. És en aquest punt on el codi imprimeix per la sortida estàndard el que ha succeït.

6.4.3 Explicació de les modificacions finals

Així doncs, la modificació realitzada consisteix en l'adaptació del propi sistema de log, en concret la part de client, per extreure la informació necessària per al benchmark. L'objectiu és afegir a cadascun dels successos del videojoc l'estimació de *FPS* que en aquell moment el client està obtenint. Això ens permetrà tindre una visió general de com afecten els successos dins del videojoc a la visualització del client donant-nos una estimació general del funcionament del videojoc segons la càrrega d'aquest.

Per aconseguir aquest objectiu s'ha afegit a tots els successos possibles una escriptura mitjançant CG_Printf, que és la funció que permet escriure per sortida estàndard. Com s'ha comentat anteriorment s'ha decidit extreure la informació dels FPS que el jugador veu en el moment del succés així com la posició on ha succeït l'event. Informació que trobem dins l'estructura de dades *cg_fps* i *position* respectivament.

6.4.4 Exemple de configuració del sistema de log

Per habilitar el nou sistema de log és necessari assignar a la variable log_file el nom de l'arxiu on volem que es guardi la informació. Així doncs, aquesta assignació s'inserirà a l'inici de l'arxiu de configuració. Es pot contemplar un exemple al codi 6.6, on s'ha afegit la comanda *log_file ftixer_de_log.log* a l'arxiu del codi 6.5.

```
log_file fitxer_de_log.log
g_gametype 2
map kaos
addbot Sarge 3
addbot Sarge 3
addbot Sarge 3
```

Codi 6.6: Exemple d'arxiu de configuració de log

6.4.5 Exemple d'una arxiu de log

A continuació es mostra un exemple d'una possible sortida de l'arxiu de log, codi 6.7. En les línies senars es pot observar el succés dins del videojoc, les línies parells mostren els frames per segons en mitjana en el moment del succés i a continuació, separat pel caràcter "|", la posició del succés en coordenades del mapa.

```
Sarge was gunned down by Grunt.
25 FPS | position : 354 140 -103
Grunt squished.
27 FPS | position : 354 140 -103
Major was rocket by Gargoyle.
22 FPS | position : 354 140 -103
```

Codi 6.7: Exemple d'arxiu de log

Capítol 7

Entorn de desenvolupament

Per realitzar qualsevol projecte informàtic un pas molt important és l'elecció de l'entorn de desenvolupament. És de vital importància escollir productes on el seu procés d'aprenentatge estigui amb correspondència amb el resultat obtingut. És per aquests motius que s'han escollit productes d'una gran senzillesa utilitzant únicament les funcionalitats necessàries per la realització del projecte. A més a més, corresponent a la defensa de la llibertat de software promoguda pels creadors del Quake 3, s'ha decidit utilitzar software de les mateixes característiques.

7.1 Sistema operatiu

Per al desenvolupament del projecte s'ha utilitzat el sistema operatiu Linux, en la seva distribució creada per Gentoo, logo a la figura 1.7. El kernel utilitzat ha estat el 2.6.24-r8 modificat per l'equip de Gentoo, aplicant també les optimitzacions mm (Andrew Morton). Totes les aplicacions tant de sistema com de gestió han estat compilades manualment a través de la interfície emerge proveïda per Gentoo.

Fig 7.1 Logo de Gentoo

7.2 Aplicacions utilitzades

Les aplicacions utilitzades durant tot el procés de l'elaboració del projecte han estat mrxvt, Code::Blocks, OpenOffice, Gimp, GanttProject i la diversitat de comandes proveïdes per Linux. Totes les aplicacions utilitzades en aquest projecte estan llicenciades sota la GPL.

7.2.1 Mrxvt

Descripció:	**Mrxvt** és el gestor de consoles que s'ha utilitzat per la navegació i cerca per el sistema de fitxers del codi del videojoc.
Versió:	0.5.1
Empresa:	SourceForge.net Project (gi1242, jimmyzhou)
Web:	http://sourceforge.net/projects/materm
Avantatges:	**Mrxvt** permet tindre varies consoles obertes amb diferents tabs. **Mrxvt** és portable, lleuger i ràpid.
Inconvenients:	Com qualsevol altre consola dels sistemes Linux el domini de la consola té una corba d'aprenentatge llarga.

7.2.2 Code::Blocks

Descripció:	**Code::Blocks** és un entorn de desenvolupament integrat (IDE), s'ha utilitzat per la modificació del codi i la navegació per aquest.
Versió:	svn build rev 3536
Empresa:	The Code::Blocks Team
Web:	http://www.codeblocks.org/
Avantatges:	**Code::Blocks** permet una navegació senzilla i intuïtiva per la estructura de fitxers del codi, visualitzant les funcions segons el llenguatge amb el que està escrit.
Inconvenients:	Al ser un IDE gràfic necessita un cert temps d'adaptació a la situació de les opcions necessàries. Les opcions de compilació de projectes no està suficientment adaptat al sistema Makefile, motiu pel qual, no s'ha utilitzat.

7.2.3 OpenOffice

Descripció:	**OpenOffice** és un paquet ofimàtic, del qual s'ha utilitzat el seu mòdul d'escritura (Writer) per l'elaboració de tota la documentació.
Versió:	2.2.0
Empresa:	OpenOffice.org (sponsored by Sun Microsystems)
Web:	http://www.openoffice.org/
Avantatges:	**OpenOffice** és un paquet ofimàtic senzill d'utilitzar i ràpid d'aprendre. També té un bon mòdul de PostScript que ens permet convertir arxius a PDF amb una gran fidelitat.
Inconvenients:	Com qualsevol programa amb interfície necessita un procés d'aprenentatge, però en aquest cas el funcionament és molt intuïtiu.

7.2.4 Gimp

Descripció:	**Gimp** és un paquet de manipulació gràfica, s'ha utilitzat per l'adequació de les imatges a la memòria del PFC.
Versió:	2.2.14
Empresa:	The GIMP Development Team
Web:	http://www.gimp.org/
Avantatges:	**Gimp** ofereix una eina de manipulació i edició gràfica amb una gran potència. Posseeix un llenguatge de programació que permet qualsevol manipulació gràfica imaginable.
Inconvenients:	Té un procés d'aprenentatge realment complex, amb un inici senzill però un domini difícil.

7.2.5 GanttProject

Descripció:	**GanttProject** és un programa de creació de gràfics de Gantt.
Versió:	2.0.6
Empresa:	SourceForge.net Project (bbadmin, dbarashev)
Web:	http://ganttproject.biz/
Avantatges:	**GanttProject** és un eina senzilla només destinada a la creació de gràfics de Gantt, cosa que el fa realment fàcil d'utilitzar.
Inconvenients:	La conversió a imatge és molt limitat, sense permetre configurar escales de temps ni dimensions de les fonts.

Capítol 8

Avaluació de PFC_Quake

Una etapa important dins l'elaboració de qualsevol projecte és l'anàlisi i revisió d'aquest. És important, doncs, realitzar un procés de testeig de les funcionalitats que s'han pretès desenvolupar i avaluar com s'adeqüen aquestes als objectius establerts.

8.1 Descripció del procés de testeig

Per comprovar el funcionament del PFC_Quake, s'ha realitzat un sistema de testeig consistent en l'execució de l'aplicació en diferents escenaris amb diferents configuracions. Tots els escenaris s'han testejat en dues màquines diferents. En cadascuna de les proves d'execució s'ha seguit el mateix esquema d'execució.

8.1.1 Configuracions de les màquines utilitzades

Les configuracions de les màquines utilitzades per les proves realitzades han estat les següents.

Màquina A:

Microprocessador	Intel Dual Core2 6600 2.4GHz
Memòria RAM	2 GB
Disc Dur	200 GB
S.O.	Gentoo Linux (kernel 2.6.19)
Altres	Nvidia Gforce 9600 GT (512 MB)

Màquina B:

Microprocessador	Pentium M 1.4 GHz
Memòria RAM	653 MB
Disc Dur	30 GB
S.O.	Gentoo Linux (kernel 2.6.19)
Altres	Intel 855GM (128 MB)

8.1.2 Esquema del procés d'execució de cada escenari

1. **Execució del videojoc:** En primer lloc es formula una crida correcte a l'executable del videojoc i es comprova que aquesta es processi adequadament. És en aquest apartat on es comprova el correcte funcionament del sistema de configuració de les partides.
2. **Connexió del jugador a la partida:** Un cop creada la instància del videojoc es connecta el jugador a la partida. Aquest procés s'executa d'acord amb la configuració concreta de cada escenari.
3. **Comprovació de l'estat espectador:** Una vegada connectat el jugador a la partida es comprova que aquest ho hagi fet en mode espectador.
4. **Comprovació del correcte funcionament del sistema de bots:** A continuació es comprova que els bots funcionin correctament. Aquest procés es divideix en dos comprovacions, una a través de la pròpia interfície del joc i una altre a través dels missatges de consola.
5. **Comprovació del manteniment d'estat entre mapes:** Dins el mode espectador es comprova que l'estat es mantingui amb els canvis de mapa. Després es passa a comprovar que un cop seleccionat un equip, aquest també es mantingui en els canvis de mapa.
6. **Comprovació de l'arxiu de log:** Un cop tancada la partida es comprova que el sistema de log hagi emmagatzemat la informació correctament.

Si tots aquest passos es realitzen correctament, el test és vàlid.

8.2 Proves de funcionament

Per modelitzar les diferents situacions d'execució del videojoc s'han creat set escenaris diferents caracteritzats pels diferents mètodes de creació i especificació de les partides. Per validar cadascun dels escenaris es comprova que realitzi tot el procés d'execució explicat a la secció 8.1.2.

- **Escenari 1:** El primer escenari de la prova consisteix en la creació de diferents partides en mode *single player* a partir del menú proveït per Quake 3. Totes les partides han satisfet els requisits establerts.
- **Escenari 2:** El segon escenari consisteix en la creació de partides en mode *multiplayer* de diferents partides a través del menú. En aquest cas també s'ha provat totes les modalitats de joc. Totes les partides han estat satisfactòries.
- **Escenari 3:** El tercer escenari ha consistit en la creació de partides *multiplayer*, però aquest cop el client ha testejar ha estat una nova instància del videojoc des de la qual s'ha conectat al primer servidor. En totes les modalitats testejades els resultats han estat els esperats.
- **Escenari 4:** El quart escenari, al igual que el primer, ha consistit en la creació de partides *single* player, en aquest cas, però, les partides han estat configurades a través del sistema de configuració per fitxer. Totes les proves han estat satisfactòries.

- **Escenari 5:** En aquest cas les partides a testejar han estat en mode *multiplayer* creades a partir del sistema de configuració a través de fitxer. Les partides s'han executat correctament.

- **Escenari 6:** Per últim s'ha testejat les partides *multiplayer* on el jugador es connecta a través de la xarxa, per crear les partides s'ha utilitzat el sistema de configuració per fitxers. Totes les proves s'han satisfet correctament.

8.3 Avaluació dels objectius

8.3.1 Objectius previs

L'objectiu inicial, d'entendre profundament el funcionament tècnic del joc, s'ha satisfet. El procés d'enginyeria inversa ha permès descobrir com han estat programades les funcionalitats del videojoc i modificar-les per al benefici del projecte.

8.3.2 Objectius principals

Objectiu 1: Com s'ha pogut comprovar gràcies a les proves anteriorment comentades, l'objectiu de modificar i adaptar les funcionalitats per crear partides del joc entre jugadors controlats per la màquina, ha estat satisfet.

Objectiu 2: Els escenaris 4, 5 i 6 permeten observar que el sistema de creació de partides preconfigurades a partir d'un arxiu de configuració funciona correctament.

***Objectiu* 3:** Analitzant tots els escenaris també es pot observar que s'ha adaptat correctament el sistema de recollida d'informació de la partida. Actualment permet l'extracció d'informació per estudiar el rendiment del videojoc.

Capítol 9

Planificació i valor econòmic

Els projectes d'investigació en general, i d'enginyeria inversa en concret, comporten la dificultat de tindre un elevat desconeixement de la situació a estudiar. És per això que en aquests el grau de planificació està sotmès al propi funcionament de la investigació i a la seva evolució. Així doncs el timing de les tasques a realitzar ha d'estar en continua avaluació i readaptació.

9.1 Planificació inicial del projecte

L'enginyeria inversa és un procés complex de planificar, degut a que es parteix d'un desconeixement elevat del producte a tractar. El fet de desconèixer en un principi de com estan estructurades les funcionalitats del programa obliga a tindre molta cautela a l'hora de calcular temps exactes per cadascun dels punts.

Així doncs, en un primer moment es va decidir realitzar una planificació lleugera dels marges temporals destinats a cadascuna de les etapes. Es va separar la duració del projecte en tres etapes, en una primera es realitzaria la tasca d'enginyeria inversa consistent a estudiar en profunditat el funcionament del videojoc i les capacitats que aquest oferia per aconseguir una bona eina de benchmark. La segona etapa consistiria en implementar les funcionalitats desitjades en un principi, és a dir, la possibilitat de l'execució de partides sense cap jugador humà, i també de les funcionalitats que emanessin de la investigació realitzada. Per finalitzar es passaria a realitzar la redacció de la memòria del projecte, veure figura 9.1.

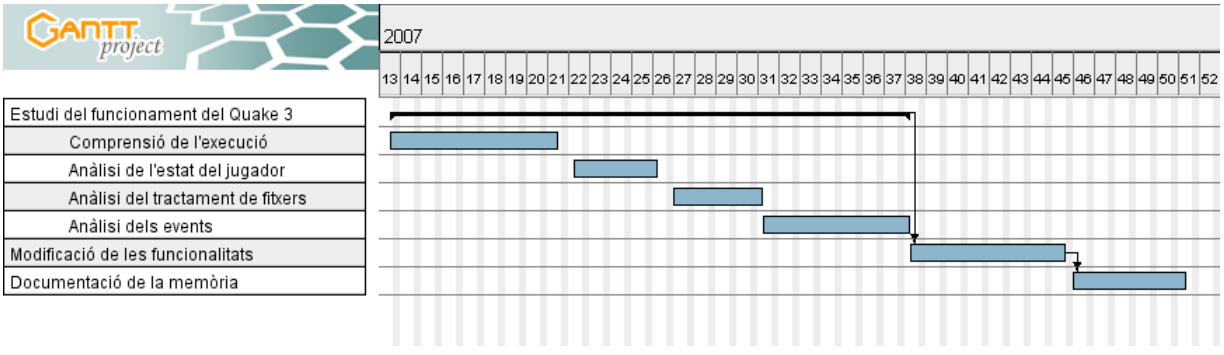

fig 9.1: Planificació inicial del projecte

9.2 Planificació posterior al procés d'enginyeria inversa

Un cop realitzada la primera fase del projecte es remodelaren els objectius aconseguint concretar-los com prenen forma al capítol 2. Fou llavors quan es va decidir desglossar els objectius en tres punts bàsics d'acció permetent el desenvolupament en paral·lel de les funcionalitats. Gràcies a aquest procés ha estat possible realitzar el capítol 4 de la memòria a mida que les tres línies de treball anaven avançant.

A continuació es descriu el diagrama de Gantt amb la planificació realitzada, figura 9.2.

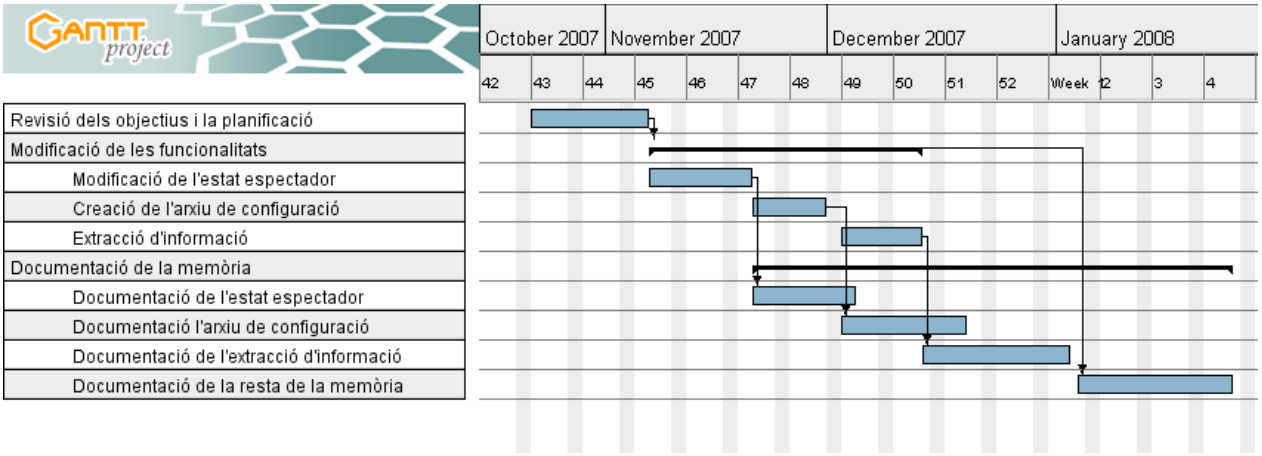

fig 9.2: Planificació realitzada

9.3 Error respecte la planificació estimada

Degut a la compaginació del projecte amb diverses activitats d'àmbit personal han fet impossible el compliment dels terminis prevists al inici. Després d'una revisió i reestructuració de la feina es prengué la decisió de realitzar una

planificació a dia 28 d'abril, veure figura 9.3.

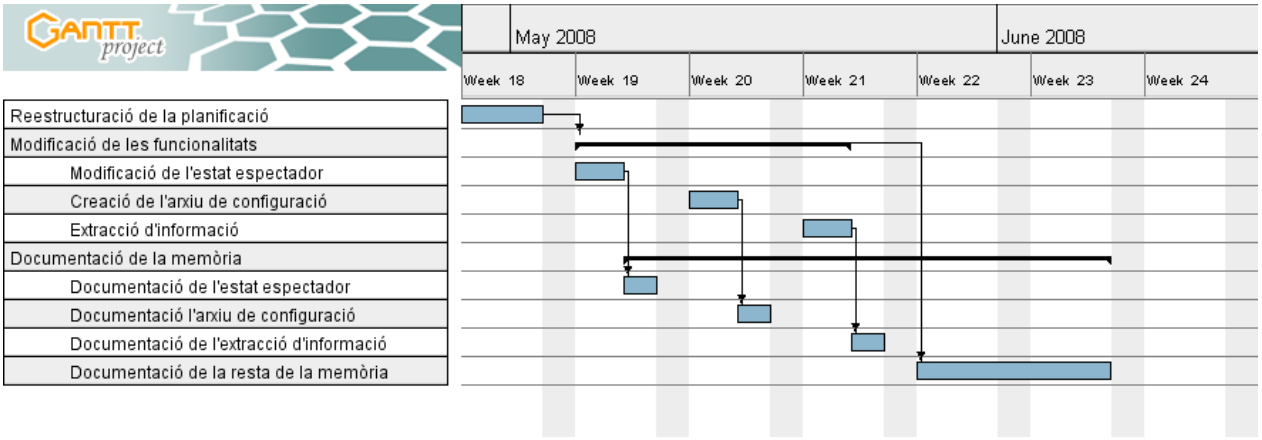

fig 9.3: Planificació final del projecte

Aquest cop l'error de planificació ha estat minúscul i exclusivament referent als punts de documentació.

9.4 Costs del projecte

El còmput total de dedicació al projecte ha estat de 689 hores. D'aquestes, 324 hores han estat destinades al procés d'enginyeria inversa i programació. És difícil posar una línia concreta entre les tasques considerades d'enginyeria inversa i de programació, ja que el procés de testeig/error ha sigut un recurs molt utilitzat. Per calcular el cost, però, aproximarem les responsabilitats del programador a un terç del temps assignant els dos terços restants a la figura d'analista. Per realitzar la documentació s'ha invertit un temps estimat de 365 hores.

Considerant que la tasca d'investigació la realitza un *analista* especialitzat en videojocs, i que el cost aproximat és de 48 € l'hora, l'etapa d'enginyeria inversa té

un cost teòric de 13.608 €.

La implementació la realitza un *programador* amb un cost de 30€ l'hora, fent que el cost teòric de la modificació de les funcionalitats ascendeixi a 3.240 €.

Per últim considerarem que la tasca de documentació s'atribueix conjuntament al perfil d'un programador i un analista, distribuint la càrrega a un terç per l'analista i la part restant al programador. Això suposa un cost total de documentació de 13.140 €.

Així doncs, el cost econòmic total del projecte és de 26.748 €, quantitat que es pot observar desglossada a la figura 8.4. A aquesta quantitat caldria afegir els costs col·laterals d'amortització de negoci, així com la dedicació d'administració i gestió de grup necessària per a que la comunicació entre el personal del projecte sigui satisfactòria.

	Temps	**Responsable**	**Cost**
Enginyeria inversa i Implementació	324 hores	2/3 Analista	10.368 €
		1/3 Programador	3.240 €
Documentació	365 hores	1/3 Analista	5.840 €
		2/3 Programador	7.300 €
Total	689 hores		26.748 €

fig 9.4: Taula de costs del projecte

9.5 Beneficis del projecte

Tot i la xifra de cost del projecte, no hem d'oblidar que aquest projecte neix amb una vocació de llibertat del producte. Aquest ha estat realitzat gràcies a la mateixa vocació que d'altres informàtics han tingut durant l'evolució de la informàtica. Així doncs, aquest projecte s'ha d'avaluar segons la contribució que realitza dins la comunitat informàtica i els beneficis futurs que aquest pot generar.

És complex calcular els beneficis econòmics concrets que PFC_Quake pot generar en un futur, però es obvi que gràcies a aquesta eina la comunitat disposa de mecanismes més senzills i ràpids de testejar el funcionament d'aplicacions relacionades amb els videojocs, així com també facilita la comprovació del correcte funcionament de nous dissenys d'arquitectura del hardware.

Capítol 10

Conclusions

El mercat de la investigació i el desenvolupament de maquinari al voltant del món dels videojocs es veu enriquit per aquesta eina de benchmark que permet provar i avaluar el rendiment de possibles futurs dissenys. I no tan sols això, la comunitat de software lliure també es veu enriquida per una nova solució que permet tant l'aprenentatge com l'aplicació d'aquesta eina en projectes futurs

L'actual creixement del sector dels videojocs ha fet aparèixer una competitivitat elevada dins el mercat. Les empreses desenvolupadores d'aquest software han hagut d'optar per estar al capdavant de la innovació invertint grans quantitats d'esforços per aconseguir part del mercat.

El realisme dels videojocs ha estat i continua sent un dels punts més importants per al que fa a innovació, i cada cop son més les necessitats de hardware per poder visualitzar aquestes realitats virtuals complexes.

Aquesta mateixa necessitat fa que el sector del hardware destinat a videojocs també posseeixi unes grans possibilitats de mercat, veient-se també amb l'obligació de mantenir una quota d'investigació i innovació elevades. Dins aquesta investigació és necessari la simulació de possibles dissenys amb el seu testeig corresponent. I és aquí un apareix la necessitat de disposar d'una eina de *benchmark* per comprovar el correcte funcionament dels nous dissenys.

PFC_Quake té la intenció de solucionar aquest problema oferint un videojoc real i complet preparat per a ser utilitzat com a joc de proves per futurs dissenys informàtics. Cal destacat, també, la funció pedagògica que aquest producte ofereix a la comunitat de software lliure, retornant així part del benefici del qual tot informàtic de la comunitat disposa.

Així doncs, PFC_Quake compleix amb els objectius del projecte esdevenint el primer *benchmark* de software lliure destinat específicament per a l'execució d'un videojoc. Aquest projecte, doncs, no és més que un dels petits grans de sorra que

fan de la comunitat de software lliure un mercat de compra/venta on la única moneda de canvi és la predisposició, l'esforç i la voluntat d'apendre.

És per això que em sento molt orgullós del resultat obtingut. Puc dir, sense cap por, que s'han assolit les expectatives creades a l'inici del projecte. Espero poder continuar investigant dins el món dels videojocs gràcies als coneixements apresos dins aquest projecte.

Apèndixs

A – Manual d'execució

A.1 Requisits mínims/recomanats

Requisits mínims del sistema:

Microprocessador	Pentium III 1.2 Ghz
Memòria RAM	256 MB
Disc Dur	1 GB
S.O.	Windows 98/ME/00/XP, Mac o Linux
Altres	Targeta Acceleradora 16 MB

Requisits recomanats del sistema:

Microprocessador	Processador Intel® Pentium® 4
Memòria RAM	512 MB
Disc Dur	1 GB
S.O.	Windows 98/ME/00/XP, Mac o Linux
Altres	Targeta Acceleradora 64 MB

A.2 IoQuake i OpenArena

El projecte PFC_Quake està basat en el codi alliberat de Quake 3, concretament la revisió realitzada per l'equip IoQuake. El codi es pot trobar a http://ioquake3.org/get-it/ .

Els arxius gràfics del Quake 3 continuen sent propietat dels respectius artistes, així que per al projecte s'ha utilitzat una recreació d'aquests creada per

l'equip OpenArena. Els arxius utilitzats es poden trobar al *torrent* https://cat-man-du.com/torrent/torrents/OpenArena 200.8.0.torrent

A.3 Compilació

Un cop desempaquetat IoQuake, cal modificar l'arxiu Makefile per tal que compili les màquines virtuals a la carpeta pfc_quake. Per tal d'aconseguir aquest canvi s'ha de canviar les aparicions de baseq3 per pfc_quake. En aquest punt és on s'han de realitzar les modificacions del projecte. A continuació es crida a la comanda *make* per tal que es compili es videojoc. Un cop compilat és necessari utilitzar els arxius pk3 de OpenArena, que els ubicarem dins la carpeta ioquake/build/pfc_quake.

A.4 Execució de PFC_Quake

Per executar el joc s'ha de realitzar una crida al executable pfc_quake.i386 dins la carpeta *build*. El sistema de *cvars* permet modificar aquestes a través de la crida de l'executable de la forma següent:

./pfc_quake.i386 +set "cvar_name" "cvar_valor"

Així doncs, per realitzar una crida a PFC_Quake la sintaxis utilitzada serà de l'estil:

./pfc_quake.i386 +set game_file "game_file_path"

Cal observar, que l'arxiu de log es pot especificar tant a la pròpia crida del programa o dins el fitxer de *game_file*.

B - Glossari

Benchmark: Joc de proves utilitzats per comprovar el correcte funcionament de les funcionalitats implementades per un software o simulació.

Bot: Jugador d'un videojoc controlat per la màquina, disposa de intel·ligència artificial.

Cheats: Terme anglès que es refereix al fet de fer trampes, aprofitant errors del videojoc o aprofitar funcionalitats que no han estat dissenyades per tal efecte.

Flags: Conjunt d'opcions assignables a una variable que posteriorment s'utilitzen per definir certs comportaments de funcionament de la pròpia variable.

Frame: Cadascuna de les imatges que el jugador pot veure per pantalla, dins el Quake 3 s'equipara a un batec que es el temps mínim d'acció de qualsevol succés dins el joc. L'alternació continua d'aquestes imatges dona la sensació que les accions succeeixin en paral·lel.

Frames per segon: Quantitat de frames dibuixats per segons, també anomenat FPS. És un dels paràmetres més importants a l'hora de determinar l'eficiència d'un videojoc.

Gentoo: Distribució de Linux que té la peculiaritat de compilar a partir del codi font tots els programes utilitzats permetent optimitzar i configurar individualment totes les funcionalitats.

GPL: General Public License, llicència sota la que s'empara el software lliure per garantir els drets de la seva llibertat.

MD3: Format de models 3D utilitzat per el motor del Quake 3 així com molts d'altres videojocs.

Motor de rendering: Part del codi d'un videojoc que s'encarrega de mantenir i reproduir la realitat virtual i la seva evolució.

Parsing: Procés informàtic consistent en general a convertir una entitat d'un tipus a un altre. En el cas tractat en aquest document, el procés consisteix en convertir una cadena de caràcters, és a dir un string, a un número enter o coma flotant.

Servidor dedicat: Modalitat de funcionament de qualsevol servidor que permet la inexistència de client. Això ens permet executar la part servidor del videojoc sense cargar cap part client.

Servidor master: El servidor master són un conjunt de servidors proveïts per Id Software en el moment de llançament del joc que permetien registrar les partides d'aquest per a que la des d'Internet qualsevol persona s'hi pogués connectar. Entre els servidor i el servidor master s'estableix un sincronisme amb diversa informació.

Sortida estàndard: Canal de comunicació d'una aplicació per la qual pot treure informació. Normalment aquesta sortida està dirigida cap a la consola per tal de poder visualitzar la informació treta.

Tipus de partida single player: Modalitat de partida on el jugador entra

en una partida multiplayer amb la singularitat que tots els jugadors adversaris estan controlats per la màquina. Aquesta modalitat ofereix un grau ascendent de dificultat tant en nombre d'adversaris com en la intel·ligència artificial dels bots.

Tipus de partida multiplayer: Dins la versió oficial de Quake 3 existeixen 4 modalitats de joc. La modalitat Free for All consisteix en una lluita tots contra tots, Team Deathmatch consisteix en batalles entre 2 equips compatibilitzant les morts per equip, el Torneig permet un sistema de classificació entre els jugadors i el Capture the Flag (CTF) que consisteix en capturar la bandera de l'equip contrari.

C - Índex de figures i codis

C.1 Introducció

C.4 Anàlisi del funcionament del Quake 3

C.5 Estructura interna del Quake 3

C.6 Modificacions de les funcionalitats

C.7 Entorn de desenvolupament

C.9 Planificació i valor econòmic

D - Variables cvar del Quake 3

D.1 Variables referents al sistema de bots

Variable	Descripció
bot_aasoptimize "0"	optimize the .aas file when one is written
bot_challenge "0"	make the bot a bit more challenging
bot_debug "0"	toggle debugging tool for bot code
bot_developer "0"	toggle developer mode for bots
bot_enable "0"	enable and disable adding of bots to the map/game
bot_fastchat "0"	toggle between frequent and less frequent bot chat strings 1 = more often
bot_forceclustering "0"	force recalculating the aas clusters - MrElusive
bot_forcereachability "0"	force recalculating the aas reachabilities - MrElusive
bot_forcewrite "0"	force writing out a new .aas file - MrElusive
bot_grapple "0"	toggle determines weather the bots will use the grappling hook
bot_groundonly "1"	this is a debug cvar to show areas which does not work in the retail version special thanks to - MrElusive
bot_interbreedbots "10"	number of bots used for goal fuzzy logic interbreeding - MrElusive
bot_interbreedchar ""	bot character to be used with goal fuzzy logic interbreeding - MrElusive
bot_interbreedcycle "20"	number of matches between interbreeding - MrElusive
bot_interbreedwrite ""	file to write interbreeded goal fuzzy logic to - MrElusive

bot_maxdebugpolys "128"	max number of polygons available for visualizing things when debugging MrElusive
bot_memorydump "0"	possibly displays memory allocation/use for bots used for debugging?
bot_minplayers "0"	this is used to ensure a minimum numbers of players are playing on a server bots are added/removed to get the specified number of players in the game special thanks to - MrElusive
bot_nochat "0"	toggle determines weather bots will chat or not 0 = bots will chat
bot_pause "0"	debug command to pause the bots - MrElusive
bot_predictobstacles "1"	possibly tells bot's to predict an obstacle and turn before running into it
bot_reachability "0"	this is a debug cvar which does not work in the retail version - MrElusive
bot_reloadcharacters "0"	this cvar if set to 1 disabled bot character file caching. used when creating bot characters while keeping Q3A running. kicking and re-adding a bot will reload the bot character files - MrElusive
bot_report "0"	debug command to have the bots report what they are doing in CTF MrElusive
bot_rocketjump "1"	toggle determines weather the bots will use the rocket jump technique
bot_saveroutingcache "0"	possibly allows the BOT AI to save routes for custom maps in memory.
bot_testclusters "0"	possibly a debug variable for testing BOT's on new terrain maps
bot_testichat "0"	used to test the initial bot chats. set this to 1 and add a bot. the bot will spit out all initial chats.
bot_testrchat "0"	used to test the reply chats. set this to 1 and add one bot. the bot will always reply

bot_testsolid "0"	test for "solid areas" in the .aas file (read the q3r manual)
bot_thinktime "100"	this is the time in milliseconds between two AI frames.
bot_usehook "0"	toggle determines weather the bots will use the grappling hook
bot_visualizejumppads "0"	visualizes the default arch of a jumppad (read the q3r manual)

D.2 Variables de les opcions de les partides del client

Variable	Descripció
cg_animspeed "1"	toggle linear interpolation between successive frames in a player animation. 0 = no interpolation 1 = it does interpolate - Coriolis + WhatEver
cg_autoswitch "1"	auto-switch weapons (on pick-up)
cg_bobpitch "0.002"	set amount player view bobs forward/back while moving
cg_bobroll "0.002"	set amount player view rolls side to side while moving
cg_bobup "0.005"	set amount player view bobs up/down while moving
cg_brassTime "1250"	set amount of time a shell casing gets displayed if set to 0 the game engine will skip all shell eject code
cg_cameraOrbit "0"	change the step or increment units of the orbit rotation from one angle how much of a step to next angle

cg_cameraOrbitDelay "50"	change the rate at wich the camara moves to the next orbit position the higher the number the slower
cg_centertime "3"	set display time for center screen messages (0 off)
cg_crosshairHealth "1"	show health by the cross hairs (only works with #10 now?) - LOKi
cg_crosshairSize "24"	crosshair size...incase you have crosshair envy (c:
cg_crosshairX "0"	set X coordinates of the crosshair if cg_crosshairSize not 0
cg_crosshairY "0"	set Y coordinates of the crosshair if cg_crosshairSize not 0
cg_debuganim "0"	toggle model animation debug mode
cg_debugevents "0"	toggle event debug mode
cg_debugposition "0"	toggle player position debug mode
cg_deferPlayers "1"	the loading of player models will not take place until the next map, or when you die, or toggle the scoreboard (tab) this prevents the "hitch" effect when a player using a new model or skin joins the game after you. if you join the game after them the models and skins will download as you join?
cg_demoLook "0"	possibly to change the look of a recorded demo?
cg_draw2D "1"	toggle the drawing of 2D items or text on the status display
cg_draw3dIcons "1"	toggle the drawing of 3D icons on the HUD off and on draw 2D icon for ammo if cg_draw3dicons 0 "John Carmack"
cg_drawAmmoWarning "1"	toggle low-ammo warning display
cg_drawAttacker "1"	toggle the display of last know assailant
cg_drawCrosshair "1"	select crosshair (change to zero if you have really good aim ha! ha!) 10 crosshairs to select from (cg_drawCrosshair 1 - 10) "John Carmack"

cg_drawCrosshairNames "1"	toggle displaying of the name of the player you're aiming at
cg_drawFPS "0"	toggle Frames Per Second display (when set to one "0" is default)
cg_drawFriend "1"	toggle the display of triangle shaped icon over the heads of your team mates
cg_drawGun "1"	toggle determines if the weapon you're holding is visible or not
cg_drawIcons "1"	toggle the drawing of any icons on the HUD and scoreboard
cg_drawRewards "1"	toggle display of award icons above the "you fragged..." message - LOKi
cg_drawKiller "1"	toggle display of player's name and picture that fragged you last
cg_drawSnapshot "0"	toggle the display of snapshots counter (# of snaps since game start)
cg_drawStatus "1"	draw the HUD. (toggle weather or not health and score are displayed)
cg_drawTeamOverlay "0"	set the drawing location of the team status overlay 1=top right 2=bottom right 3=bottom left of the screen it shows team player names, location, ammo (and what type weapon), and frag count for each player - LOKi
cg_drawTimer "1"	show timer on HUD. shows time since map start counts up - LOKi
cg_errordecay "100"	helps to smooth animation during player prediction while experiencing packet loss or snapshot errors. "detect prediction errors and allow them to be decayed off over several frames to ease the jerk." from the source code comments cg_predict.c
cg_extrapolate "1"	toggle blending of animations from one to the next (like a segue) - Andre

cg_footsteps "1"	toggle the footstep sounds of all players (cheat protected) - LOKi
cg_forceModel "0"	force model selection, also forces player sounds "John Carmack"
cg_fov "90"	field of view/vision "90" is default higher numbers give peripheral vision.
cg_gibs "1"	toggle the display of animated gibs (explosions flying body parts!)
cg_gun "1"	toggle determines if the weapon your holding is visible or not
cg_gunX "0"	set X coordinates of viewable weapon if cg_drawGun is set to 1
cg_gunY "0"	set Y coordinates of viewable weapon if cg_drawGun is set to 1
cg_gunZ "0"	set Z coordinates of viewable weapon if cg_drawGun is set to 1 moves the gun model forward or backward in relation to the player models hold
cg_ignore "0"	used for debugging possibly like the notarget command
cg_lagometer "1"	toggle the display of Lag-O-Meter on the HUD 1=netgraph 0=frag counter which changes color to reflect what place you're in.
cg_markoffset "1"	set marks (decals) offset. some video cards display the marks with the wrong offset, so you will be able to see the square decal that encapsulates the effect because the offset rises above the wall surface. change the offset the square goes away
cg_marks "1"	toggle the marks the projectiles leave on the wall (bullet holes, etc)
cg_noplayeranims "0"	toggle player model animations.

cg_nopredict "0"	toggle client-side player prediction. (disabling causes the client to wait for updates from the server before updating the player location.) .
cg_noProjectileTrail "0"	toggle the display of smoke trail effect behind rockets - Jax_Gator Dekard
cg_noTaunt "0"	possibly turn off the ability to hear voice taunts
cg_noVoiceChats "0"	possibly turn off the ability to hear voice chats
cg_noVoiceText "0"	possibly turn off the display of the voice chat text copied to the console
cg_oldPlasma "1"	toggle the use of old or new particle style plasma gun effect - 20 20
cg_oldRail "0"	toggle the use of old or new spiral style rail trail effect - 20 20
cg_oldRocket "1"	toggle the use of old or new style rocket trail effect - 20 20
cg_predictItems "1"	toggle client-side item prediction. 0 option to not do local prediction of item pickup - John Carmack
cg_railTrailTime "400"	set how long the railgun's trails last
cg_runpitch "0.002"	set amount player view bobs up and down while running
cg_runroll "0.005"	set amount player view rolls side to side while running (in 3rd person only?)
cg_scorePlums "1"	toggle the display of the floating scoring number balloons when a player scores a point or points (including negative points) in any game type, the awarded point value floats up from the target like a balloon and slowly fades out.
cg_shadows "0"	set shadow detail level (0 = OFF, 1 = basic discs, 2 = stencil buffered 3 = simple stencil buffered)
cg_showcrosshair "1"	appeared in version 1.06 then removed in 1.07 now back in 1.08 then removed again in 1.09… hmm (replaced with multi-crosshairs)

cg_showmiss "0"	toggle the display of missed packets or predictions on the HUD
cg_simpleItems "0"	toggle the use of 2D sprite objects in place of the 3D animated objects makes some objects more "simple" (faster to render) - hacker
cg_smoothClients "0"	when g_smoothClients is enabled on the server and you enable cg_smoothClients then players in your view will be predicted and will appear more smooth even if they are on a bad network connection. however small prediction errors might appear.
cg_stats "0"	toggles display of client frames in sequence missed frames are not shown
cg_stereoSeparation "0.4"	the amount of stereo separation (for 3D glasses!) You ever take off your glasses at a 3D movie, remember how the images were separated into 3 colors? that's what this does
cg_swingSpeed "0.3"	set speed player model rotates to match position (1 is no delay, 0 will never turn)
cg_teamChatHeight "8"	set number of lines or strings of text that remain on screen in team play chat mode (messagemode2) values are 1 - 8 - LOKi
cg_teamChatsOnly "0"	when this is set to a one only chats from team mates will be displayed
cg_teamChatTime "3000"	set how long messages from teammates are displayed on the screen
cg_temp "0"	
cg_testentities "0"	
cg_thirdPerson "0"	toggle the use of and third person view
cg_thirdPersonAngle "0"	change the angle of perspective you view your player (180 changes view to the front of the model)
cg_thirdPersonRange "40"	change the distance you view your player from when in 3rd person view

cg_timescaleFadeEnd "1"	
cg_timescaleFadeSpeed "0"	
cg_tracerchance "0.4"	set frequency of tracer bullets (1 is all tracers)
cg_tracerlength "100"	set length of tracer bullets
cg_tracerwidth "1"	set width of tracer bullets
cg_trueLightning "0"	settings of the new shaft style. from the OSP readme...specifies the "lag" imposed on the rendering of the lightning gun shaft.
cg_viewsize "100"	changes view port size 30 - 100 (you probably wouldn't want less than 100)
cg_zoomfov "22.5"	what the zoomed in field of view will be any thing more than 30 would not be sniper friendly
cg_waveamplitude "1"	
cg_wavefrequency1 "0.4"	

D.3 Variables de les opcions del videojoc del client

cl_allowDownload "1"	toggle automatic downloading of maps, models, sounds, and textures
cl_anglespeedkey "1.5"	set the speed that the direction keys (not mouse) change the view angle
cl_anonymous "0"	possibly to toggle anonymous connection to a server
cl_avidemo "0"	toggle recording of a slideshow of screenshots records into the snapshot folder and appears to have overwritten some snapshots I had in there…)c:
cl_cdkey "123456789"	variable to hold the CD key number to prevent bootleg/warez
cl_currentServerAddress	variable holds the IP address of the currently server

cl_conXOffset "0"	offset the console message display 0 - top left 999 - extreme top right (off the page)
cl_debugMove "0"	used for debugging cl_debugmove [1/2] from John Carmack's plan file
cl_downloadName ""	variable holds filename of file currently downloading
cl_forceavidemo "0"	
cl_freelook "1"	toggle the use of freelook with the mouse (your ability to look up and down)
cl_freezeDemo "0"	stops a demo play back and freeze on one frame
cl_guid ""	holds a code generated by the game that uniquely identifies your client. part of the punk buster integration
cl_maxpackets "30"	set the transmission packet size or how many packets are sent to client
cl_maxPing "800"	controls which servers are displayed in the in-game server browser - ata
cl_motd "1"	toggle the display of "Message of the day" When Quake 3 Arena starts a map up, it sends the GL_RENDERER string to the Message Of The Day server at id. This responds back with a message of the day to the client. If you wish to switch this option off, set CL_MOTD to 0.
cl_motdString ""	possibly a MOTD from id's master server it is a read only variable
cl_mouseAccel "0"	toggle the use of mouse acceleration the mouse speeds up or becomes more sensitive as it continues in one direction
cl_nodelta "0"	disable delta compression (slows net performance, only use if net errors happen otherwise not recommended)
cl_noprint "0"	printout messages to your screen or to the console (tired of all the chatter?)

cl_packetdup "1"	default was 2 but changed to 1 since version 1.09
cl_paused "0"	variable holds the status of the paused flag on the client side
cl_pitchspeed "140"	set the pitch rate when +lookup and/or +lookdown are active
cl_punkbuster "0"	enable the punkbuster client in the server browser
cl_run "1"	always run...play without it I dare you! (c:
cl_running "1"	variable which shows weather or not a client game is running or weather we are in server/client mode (read only)
cl_serverStatusResendTime "750"	possibly allows the admin to change the rate of the heartbeats to the master server(s)
cl_showmouserate "0"	show the mouse rate of mouse samples per frame (USB 1/per frame)
cl_shownet "0"	display network quality info
cl_showSend "0"	network debugging tool "John Carmack"
cl_showTimeDelta "0"	display time delta between server updates
cl_timeNudge "0"	effectively adds local lag to try to make sure you interpolate instead of extrapolate (try 100 for a really laggy server)
cl_timeout "125"	seconds to wait before you are removed from the server when you lag out.
cl_updateInfoString ""	"challenge\14985\motd\This is used by id when new versions come out"
cl_yawspeed "140"	set the yaw rate when +left and/or +right are active

D.4 Variables de les opcions de l'ordinador del client

com_blood "1"	toggle the blood mist effect in the gib animations. 0 option for no gibs and no blood on hits "John Carmack"
com_buildScript "0"	possibly used for the loading and caching of game data like a list of things to be loaded and caches the data for quicker reloading
com_cameraMode "0"	seems to toggle the view of your player model off and on when in 3D camera view
com_dropsim "0"	for testing simulates packet loss during communication drops
com_hunkMegs "20"	set the amount of memory you want quake3.exe to reserve for game play dedicated server memory optimizations. Tips: com_hunkMegs 4 sv_maxclients 3 bot_enable 0 "John Carmack"
com_introplayed "1"	toggle displaying of intro cinematic once it has been seen this variable keeps it from playing each time, to see it again set this to zero
com_maxfps "100"	set max frames per second you receive from server (maxfps was removed)
com_showtrace "0"	toggle display of packet traces. 0=disables,1=toggles.
com_soundMegs "8"	com_soundmegs and com_zonemegs can be adjusted to provide better performance on systems with more than 64mb of memory. the default configuration is set to allow the game to run on a 64 MB system.
com_speeds "0"	toggle display of frame counter, all, sv, cl, gm, rf, and bk whatever they are
com_zoneMegs "16"	com_soundmegs and com_zonemegs can be adjusted to provide better performance on systems with more than 64mb of memory. the default configuration is set to allow the game to run on a 64 MB system.

D.5 Variables referents al sistema de fitxers del videojoc

fs_basegame ""	allows people to base mods upon mods *syntax to follow*
fs_basepath ""	set base path root C:\Program Files\Quake III Arena for files to be downloaded from this path may change for TC's and MOD's
fs_cdpath ""	possibly a variable to use when the full CD was copied to the HDD
fs_copyfiles "0"	toggle if files can be copied from servers or if client will download
fs_debug "0"	possibly enables file server debug mode for download/uploads or something
fs_game ""	set gamedir set the game folder/dir default is baseq3 (other for MODS)
fs_homepath	possibly for TC's and MODS the default is the path to quake3.exe
fs_openedList ""	variable holds a list of all the pk3 files the client found
fs_referencedList ""	variable holds a list of all the pk3 files the client loaded data from
fs_restrict ""	demoversion if set to 1 restricts game to 4 arenas like the Q3A demo

D.6 Variables de les opcions de les partides del servidor

g_aimTest "0"	removed possibly was a cheat (bot like aiming)
g_allowVote "1"	toggle the use of voting on a server
g_arenaName "0"	possibly toggles the display of the name of the current arena?

g_arenaRank ""	possibly a variable to hold the value for your rank in the current series
g_arenaScores ""	possibly a variable to hold the value of previous arena series scores
g_arenasFile ""	sets the file name to use for map rotation and bot names and game type for each arena default scripts/arenas.txt within the PK3 file
g_banIPs ""	ban specified TCP/IP address from connecting to your server
g_blueTeam ""	set the icon for the blue team (example Pagans)
g_botsFile ""	sets the file name to use for setting up the bots configuration and characters for each bot default scripts/bots.txt within the PK3 file
g_debugAlloc "0"	possibly debugging tool for memory allocation?
g_debugDamage "0"	debugging tool for damage effects?
g_debugMove "0"	debugging tool for brush/entity movements?
g_doWarmup "0"	toggle the use of a warmup period before a match game
g_enableBreath "0"	enable breath in cold maps you can see the players breath - Dekard
g_enableDust "0"	enable dust to be kicked up from feet in areas that have that map entity - Dekard
g_filterBan "1"	toggle the banning of players that match a certain criteria/filter?
g_forcerespawn "10"	set the respawn time in seconds, 0 = don't force respawn
g_friendlyFire "0"	toggle damage caused by friendly fire 1 = can kill or injure teammate
g_gametype "0"	0 - Free For All, 1 – Tournament, 2 - Single Player,3 - Team Deathmatch, 4 - Capture the Flag, 5 - One Flag CTF, 6 – Overload, 7 - Harvester (Team Arena only).
g_gravity "800"	set the gravity level. (this is normally set by a property of the map loaded)

g_inactivity "0"	set the amount of time a player can remain inactive before kicked
g_knockback "1000"	the knockback from a weapon, higher number = greater knockback.
g_listEntity "0"	toggles the display of map entities shows them by number
g_log "1"	toggles logging of game data or statistics John Carmack made g_log a filename instead of a 0/1 in this version
g_logSync "0"	toggle the logging to append to the existing file and not overwrite
g_maxGameClients "0"	set maximum # of players who may join the game the remainder of clients are forced to spectate - Holesinswiss
g_motd ""	set message of the day to "X" (see "cl_motd" to display it)
g_needpass "0"	variable alerts the client that a password is needed to join your server
g_password ""	set the serverside password players use to get on the server
g_podiumDist "80"	sets the draw distance of the podium object player models stand on after a single player bot match - LOKi
g_podiumDrop "70"	sets the height of the podium object player models stand on after a single player bot match - LOKi
g_quadfactor "3"	allows the admin to set the amount of damage the quad damage will do.
g_rankings "0"	
g_redTeam ""	set the team icon for the red team (example Stroggs)
g_restarted "0"	read only variable that is toggled when the game has been restarted in match mode this sets an event trap for if warmup is needed
g_singlePlayer "0"	possibly to allow 3rd party's to make TC's for single player style games?

g_smoothClients "1"	enable players to use the smooth clients option on the server (cg_smoothClients)
g_spAwards ""	variable holds the names of the award icons that have been earned in the tier levels in single player mode
g_speed "320"	how fast you move in Q3Test. The greater the number, the greater the velocity
g_spScores1 ""	holds your scores on skill level 1 in single player games - Dr Qube
g_spScores2 ""	holds your scores on skill level 2 in single player games - Dr Qube
g_spScores3 ""	holds your scores on skill level 3 in single player games - Dr Qube
g_spScores4 ""	holds your scores on skill level 4 in single player games - Dr Qube
g_spScores5 ""	holds your scores on skill level 5 in single player games - Dr Qube
g_spSkill "2"	holds your current skill level for single player 1 = I can win 2 = bring it on 3 = hurt me plenty 4 = hardcore and 5 = nightmare
g_spVideos ""	variable holds the names of the cinematic videos that are unlocked at the end of each tier completion
g_syncronousClients "0"	toggle synching of all client movements (1 required to record server demo) show "snc" on lagometer "John Carmack"
g_teamAutoJoin "0"	toggle the automatic joining of the smallest or loosing team
g_teamForceBalance "0"	toggle the forcing of teams to be as even as possible on a server
g_warmup ""	the warmup time for tournament play is set with g_warmup. A tournament game is implicitly a one on one match, and further players are automatically entered as spectators.

g_weaponrespawn "5"	set time before a picked up weapon will respawn again 0 = weapons stay
g_weaponTeamRespawn "30"	

D.7 Variables de les opcions gràfiques del client

GL_EXT_cull_vertex **GL_EXT_packed_pixels** **GL_EXT_point_parameters**	
GL_EXT_texture_object **GL_EXT_vertex_array** **GL_WIN_swap_hint**	
gl_extensions "GL_EXT_abgr **GL_EXT_bgra** **GL_EXT_compiled_vertex_array**	
GL_KTX_buffer_region "	
gl_pixelformat ""	color(16) depth(16) stencil(8) sets up how many bits for each pixel item 8, 16, or 32 bit?
gl_renderer ""	variable holds the GL Renderer driver information "RIVA 128/RIVA 128 ZX (PCI)"
gl_vendor ""	variable holds the brand of your chipmaker "NVIDIA Corporation"
gl_version ""	variable holds the driver version number "1.1.0"
graphheight "32"	set height, in pixels?, for graph displays
graphscale "1"	set scale multiplier for graph displays
graphshift "0"	set offset for graph displays

D.8 Variables de les opcions dels perifèrics del client

in_debugjoystick "0"	possibly to set the debug level of direct input
in_joyBallScale "0.02"	possibly sets the scale of a joyball rotation to player model rotation?
in_joyBall "0"	possibly to allow support for trackball style joy sticks and orb's
in_joystick "0"	toggle the initialization of the joystick (command line)
in_logitechbug "0"	toggle the use of special code in the game that addresses a bug in the logitech mouse driver software
in_midi "0"	toggle the use of a midi port as an input device r-d-x
in_midichannel "1"	toggle the use of a midi channel as an input device r-d-x
in_mididevice "0"	toggle the use of a midi device as an input device r-d-x
in_midiport "1"	toggle the use of a midi port as an input device r-d-x
in_mouse "1"	toggle initialization of the mouse as an input device (command line)
joy_advanced "0"	applies game controller axis mapping settings < maddog
joy_advaxisr "0"	bind an action to the joystick r axis
joy_advaxisu "0"	bind an action to the joystick u axis
joy_advaxisv "0"	bind an action to the joystick v axis
joy_advaxisx "0"	bind an action to the joystick x axis
joy_advaxisy "0"	bind an action to the joystick y axis
joy_advaxisz "0"	bind an action to the joystick z axis
joy_forwardsensitivity "-1"	set forward/back sensitivity (negative is inverted)
joy_forwardthreshold "0.15"	set forward/back dead zone
joy_name "joystick"	set joystick name
joy_pitchsensitivity "1"	set pitch sensitivity (negative is inverted)
joy_pitchthreshold "0.15"	set pitch dead zone

joy_sidesensitivity "-1"	set side sensitivity (negative is inverted)
joy_sidethreshold "0.15"	set side dead zone
joy_threshold "0.15"	possibly an overall threshold setting all other joy variables removed in 1.08
joy_upsensitivity "-1"	set up/down sensitivity (negative is inverted)
joy_upthreshold "0.15"	set up/down dead zone
joy_yawsensitivity "-1"	set yaw sensitivity (negative is inverted)
joy_yawthreshold "0.15"	set yaw dead zone
m_filter "1"	toggle use of mouse "smoothing"
m_forward "0.25"	set the back and forth movement distance of the player in relation to how much the mouse moves
m_pitch "0.022"	set the up and down movement distance of the player in relation to how much the mouse moves
m_side "0.25"	set the strafe movement distance of the player in relation to how much the mouse moves
m_yaw "0.022"	set the speed at which the player's screen moves left and right while using the mouse
sensitivity "9"	set how far your mouse moves in relation to travel on the mouse pad

D.9 Variables de les opcions de xarxa

net_ip "localhost"	variable holds the IP of the local machine (or the "hosts" name) passed from the OS environment
net_noipx "0"	toggle the use of IPX/SPX network protocol (command line only)
net_noudp "0"	toggle the use of TCP/IP network protocol (command line only)

net_port "27960"	set port number server will use if you want to run more than one instance of Q3A server on the same machine
net_qport "16392"	set internal network port. this allows more than one person to play from behind a NAT router by using only one IP address - Questy
net_socksEnabled "0"	toggle the use of network socks 5 protocol enabling firewall access (only settable at init time from the OS command line) - Graeme Devine
net_socksPassword ""	variable holds password for socks firewall access supports no authentication and username/password authentication method (RFC-1929); it does NOT support GSS-API method (RFC-1961) authentication (only settable at init time from the OS command line) - Graeme Devine
net_socksPort "1080"	set proxy and/or firewall port default is 1080 (only settable at init time from the OS command line) - Graeme Devine
net_socksServer ""	set the address (name or IP number) of the SOCKS server (firewall machine), NOT a Q3ATEST server. (only settable at init time from the OS command line) - Graeme Devine
net_socksUsername ""	variable holds username for socks firewall supports no authentication and username/password authentication method (RFC-1929); it does NOT support GSS-API method (RFC-1961) authentication (only settable at init time from the OS command line) - Graeme Devine

D.10 Variables de les opcions renderització

r_allowExtensions "1"	use all of the OpenGL extensions your card is capable of

r_allowSoftwareGL "0"	toggle the use of the default software OpenGL driver supplied by the Operating System < maddog
r_ambientScale "0.5"	set the scale or intensity of ambient light
r_clear "0"	toggle the clearing of the screen between frames
r_colorbits "16"	set number of bits used for each color from 0 to 32 bit
r_colorMipLevels "0"	"texture visualization tool" John Carmack
r_customaspect "1"	toggle the use of custom screen resolution/sizes
r_customheight "1024"	custom resolution (Height)
r_customwidth "1600"	custom resolution (Width)
r_debuglight "0"	possibly toggle debugging of lighting effects
r_debugSort "0"	possibly toggle debugging of sorting of list like scoreboard
r_debugSurface "0"	possibly used for debugging the curve rendering and possibly for map debugging.
r_debugSurfaceUpdate "1"	if client game code registers a shader after drawsurfaces are generated but before frame is rendered had a one-frame visual glitch (shader indexes messed up)
r_debugSurfaceUpdate "1"	possibly used for debugging the curve rendering and possibly for map debugging.
r_depthbits "16"	set number of bits used for color depth from 0 to 24 bit
r_detailtextures "1"	toggle the use of detailed textures, when disabled every stage of a shader is rendered except those with the keyword "detail". when enabled detail stages are also rendered. in proper use the detail stages are supposed to enhance the texture's visual quality when viewed close up. more information is available in the shader manual included in the GTK Radiant install. - Rroff

r_directedScale "1"	set scale/intensity of light shinning directly upon objects
r_displayRefresh "0"	monitor refresh rate in game (will change desktop settings too in Windows 98 anyway)
r_dlightBacks "1"	"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..."John Carmack
r_drawBuffer "GL_BACK"	set which frame buffer to draw into. basically you draw into a "back" buffer while simultaneously showing a "front" buffer. next frame you "swap" these. the benefit is that you won't "see" the actual painting of the image take place. - Questy/Carl
r_drawentities "1"	toggle display of brush entities
r_drawstrips "1"	toggle triangle strips rendering method
r_drawSun "1"	set to zero if you do not want to render sunlight into the equation of lighting effects
r_drawworld "1"	toggle rendering of map architecture
r_dynamiclight "0"	toggle dynamic lighting (different "dynamic" method of rendering lights)
r_ext_compiled_vertex_array ""	toggle hardware compiled vertex array rendering method default is 1
r_ext_compress_textures "1"	toggle compression of textures
r_ext_compressed_textures "1"	toggle compression of textures (1.27g changed to past tense compressed)
r_ext_gamma_control "1"	enable external gamma control settings
r_ext_multitexture "1"	toggle hardware mutitexturing if set to zero is a direct FPS benefit
r_ext_swapinterval "1"	toggle hardware frame swapping

r_ext_texenv_add "1"	possible duplicate cvar or an extension to the r_ext_texture_add variable
r_ext_texture_env_add "1"	toggle additive blending in multitexturing. If not present, OpenGL limits you to multiplicative blending only, so additive will require an extra pass. - Questy/Carl
r_facePlaneCull "1"	toggle culling of brush faces not in view (0 will slow FPS)
r_fastsky "1"	toggle fast rendering of sky if set to 1 (0 is default and will slow FPS when outdoors 1 will disable your ability to see through portals)...Thanx hacker
r_finish "1"	toggle synchronization of rendered frames (engine will wait for GL calls to finish)
r_fixtjunctions "1"	toggle fixing of a problem with a certain type of vertex in models that can make gaps appear between polygons - Andre Lucas
r_flareFade "7"	set scale of fading of flares in relation to distance
r_flares "0"	toggle projectile flare and lighting effect. the flare effect is a translucent disk that is used to alter the colors around lights with a corona effect
r_flaresSize "40"	set the size of flares? I wish you could make the big balls smaller now those are flares
r_fullbright "0"	toggle textures to full brightness level (is set as a cheat code?) boy who turned on the lights…(c:
r_fullscreen "1"	toggle full screen or play in a window
r_gamma "1"	gamma correction
r_glDriver "opengl32"	used "x" OpenGL driver (Standard OpenGL32 or 3dfxvgl)
r_ignore "0"	possibly ignores hardware driver settings in favor of variable settings
r_ignoreFastPath "0"	possibly to disable the looking outside of the PAK file first feature in case of duplicate file names etc.

r_ignoreGLErrors "1"	ignores OpenGL errors that occur
r_ignorehwgamma "0"	possibly to toggle the use of DirectX gamma correction or video driver gamma correction?
r_ignoreOffset "0"	see r_offsetfactor this will just turn the offset off completely
r_inGameVideo "1"	toggle the display of in game animations on bigscreen map objects that display a camera view of the current game
r_intensity "1"	increase brightness of texture colors (may be like gl_modulate?)
r_lastValidRenderer ""	last known video driver (RIVA 128/RIVA 128 ZX (PCI))
r_lightmap "0"	toggle entire map to full brightness level all textures become blurred with light (is set as a cheat code?)
r_lightningSegmentLength "32"	possibly to set the distance between bends in the lightning bolt of the lightning gun…(c:
r_lockpvs "0"	disable update to PVS table as player moves through map (new areas not rendered) - Randy
r_lockview "0"	possibly was intended to lock a certain Field Of View (FOV) is removed now
r_lodbias "0"	change the geometric level of detail (0 - 2)
r_lodCurveError "250"	another level of detail setting if set to 10000 "don't drop curve rows for a long time" John Carmack (really mean 3D cards only??)
r_lodscale "5"	set scale for level of detail adjustment
r_logFile "0"	possibly toggles logging of rendering errors
r_mapOverBrightBits "2"	set intensity level of lights reflected from textures
r_maskMinidriver "0"	treat the current OpenGL32 driver as an ICD, even if it is in fact a MCD Questy/Zoid
r_maxpolys "600"	
r_maxpolyverts "3000"	

r_measureOverdraw "0"	overdraw' is when the same pixel is written to more than once when rendering a scene. I guess r_measureOverdraw is used to see how much is going on. used for software rendering
r_mode "3"	set video display mode (resolution), use listmodes for list of modes (3 is 640X480)
r_nobind "0"	toggle the binding of textures to triangles
r_nocull "0"	toggle rendering of hidden objects (1=slow performance)
r_nocurves "0"	map diagnostic command toggle the use of curved geometry
r_nolightcalc "0"	disable lighting and shadow calculations…hmm
r_noportals "0"	toggle player view through portals
r_norefresh "0"	toggle the refreshing of the rendered display
r_novis "0"	the VIS tables hold information about which areas should be displayed from other areas.
r_offsetfactor "-1"	control the OpenGL Polygon Offset, If you see lines appearing in decals, or they seem to flick on and off, these variables may help out. - Questy/Andre
r_offsetunits "-2"	see r_offsetfactor
r_overBrightBits "1"	possibly similar to r_mapOverBrightBits (no visible effect on mine)
r_picmip "1"	set maximum texture size (0 - 3, 3=fastest 0=quality)
r_portalOnly "0"	when set to "1" turns off stencil buffering for portals, this allows you to see the entire portal before it's clipped, i.e. more of the room, to get a better feel for who's in there before you jump in.
r_preloadTextures "0"	enable video processor to pre-cache textures

r_primitives "0"	set the rendering method. -1 = skips drawing 0 = uses glDrawElements if compiled vertex arrays are present, or strips of glArrayElement if not present 1 = forces strips 2 = forces drawElements 3 = path for non-vertex array testing "John Carmack"
r_printShaders "0"	possibly toggle the printing on console of the number of shaders used?
r_railCoreWidth "16"	set size of the rail trail's core
r_railSegmentLength "64"	set distance between rail "sun bursts"
r_railWidth "128"	set width of the rail trail
r_roundImagesDown "1"	set rounding down amount (larger = faster, lower quality) - Randy
r_saveFontData "0"	
r_showcluster "0"	toggle the display of clusters by number as the player enters them on the currently loaded map<maddog
r_showImages "0"	toggle displaying a collage of all image files when set to a one...texture use debugging tool
r_shownormals "0"	toggle the drawing of short lines indicating brush and entity polygon vertices, useful when debugging model lighting - Andre Lucas < maddog
r_showsky "0"	enable rendering sky in front of other objects
r_showSmp "0"	toggle display of multi processor (SMP) info on the HUD
r_showtris "0"	map diagnostic command show triangles, pretty cool looking...
r_simpleMipMaps "1"	toggle the use of "simple" mip mapping. used to "dumb-down" resoluiton displays for slower machines - Questy
r_singleShader "0"	possibly toggles use of 1 shader for objects that have multiple shaders

r_skipBackEnd "0"	possibly to toggle the skipping of the backend video buffer
r_smp "0"	toggle the use of multi processor acceleration code
r_speeds "0"	show the rendering info e.g. how many triangles are drawn added r_speeds timing info to cinematic texture uploads "John Carmack"
r_stencilbits "8"	stencil buffer size (0, 8bit, and 16bit)
r_stereo "0"	toggle the use of stereo separation for 3D glasses
r_subdivisions "4"	set maximum level of detail. (an example would be the complexity of curves. 1=highest detail)
r_swapInterval "0"	toggle frame swapping.
r_texturebits "0"	set number of bits used for each texture from 0 to 32 bit
r_textureMode ""	select texture mode. "GL_LINEAR_MIPMAP_NEAREST" (nearest or linear)
r_uiFullScreen "0"	
r_verbose "0"	toggle display of rendering commands as they happen on the console
r_vertexLight "1"	enable vertex lighting (faster, lower quality than lightmap) removes lightmaps, forces every shader to only use a single rendering pass, no layered transparancy, environment mapping, world lighting is completely static, and there is no dynamic lighting when in vertex lighting mode. (recommend dynamiclight 0 and this 1) direct FPS benefit "John Carmack"
r_znear "4"	set how close objects can be to the player before they're clipped out of the scene - Questy/Andre

D.11 Variables de les opcions de so del client

s_2dvolume "0.7"	vortex of sound - has a good description of this A3D variable
s_bloat "2.0"	vortex of sound - has a good description of this A3D variable
s_compression "1"	toggle the use of sound compression
s_distance "100.0"	vortex of sound - has a good description of this A3D variable
s_doppler "1.0"	vortex of sound - has a good description of this A3D variable
s_fogeq "0.8"	vortex of sound - has a good description of this A3D variable
s_geometry "1"	vortex of sound - has a good description of this A3D variable
s_initsound "1"	toggle weather sound is initialized or not (on next game)
s_khz "11"	set the sampling frequency of sounds lower=performance higher=quality
s_leafnum "0"	
s_loadas8bit "1"	load sounds in 8bit mode
s_max_distance "1000.0"	vortex of sound - has a good description of this A3D variable
s_min_distance "3.0"	vortex of sound - has a good description of this A3D variable
s_mixahead "0.2"	set delay before mixing sound samples.
s_mixPreStep "0.05"	possibly to set the prefetching of sound on sound cards that have that power
s_musicvolume "1"	music volume level 0=off
s_numpolys "400"	vortex of sound - has a good description of this A3D variable

s_occ_eq "0.75"	vortex of sound - has a good description of this A3D variable
s_occfactor "0.5"	vortex of sound - has a good description of this A3D variable
s_occlude "0"	vortex of sound - has a good description of this A3D variable
s_polykeep "1000000000"	
s_polyreflectsize "10000000"	
s_polysize "10000000"	
s_refdelay "2.0"	vortex of sound - has a good description of this A3D variable
s_refgain "0.45"	vortex of sound - has a good description of this A3D variable
s_reflect "1"	vortex of sound - has a good description of this A3D variable
s_rolloff "1.0"	vortex of sound - has a good description of this A3D variable
s_separation "0.5"	set separation between left and right sound channels (this one is it)
s_show "0"	toggle display of paths and filenames of all sound files as they are played.
s_testsound "0"	toggle a test tone to test sound system. 0=disables,1=toggles.
s_usingA3D "0"	vortex of sound - has a good description of this A3D variable
s_volume "0.7"	Sound FX Volume
s_watereq "0.2"	vortex of sound - has a good description of this A3D variable

D.12 Variables de les opcions del videojoc del servidor

sv_allowAnonymous "0"	possibly to toggle the allowing of anonymous clients to connect to your server
sv_allowdownload "1"	toggle the ability for clients to download files maps etc. from server. .
sv_cheats "1"	enable cheating commands (give all) (serverside only)
sv_floodProtect "1"	toggle server flood protection to keep players from bringing the server down
sv_fps "20"	set the max frames per second the server sends the client
sv_hostname ""	set the name of the server "Shadowlands"
sv_keywords ""	variable holds the search string entered in the internet connection menu
sv_killserver "0"	if set to a one the server goes down (server console only I hope)
sv_lanForceRate "1"	forces LAN clients to the maximum rate instead of accepting client setting
sv_mapChecksum ""	allows check for client server map to match
sv_mapname ""	display the name of the current map being used on a server
sv_master1 ""	set URL or address to master server "master3.idsoftware.com"
sv_master2 ""	optional master 2
sv_master3 ""	optional master 3
sv_master4 ""	optional master 4
sv_master5 ""	optional master 5
sv_maxclients "8"	maximum number of people allowed to join the server dedicated server memory optimizations. Tips: com_hunkMegs 4 sv_maxclients 3 bot_enable 0 "John Carmack"

sv_maxPing "0"	set the maximum ping aloud on the server to keep HPB out
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. Father John stepping in, in the name of fairness…(c: (ever notice when 3 or so LPB's join a server your PING takes a dump? It's because your slice of the pie got smaller because theirs is so big…die bandwidth suckers)
sv_minPing "0"	set the minimum ping aloud on the server to keep LPB out
sv_nopredict "0"	is it possible that the server is handling some prediction of player location?
sv_pad "0"	
sv_padPackets "0"	possibly toggles the padding of network packets on the server PAD - Packet Assembler/Disassembler
sv_pakNames "antilogic"	variable holds a list of all the pk3 files the server found "antilogic"
sv_paks "182784856 "	variable holds the checksum of all pk3 files
sv_paused "0"	allow the game to be paused from the server console?
sv_privateClients "0"	the number of spots, out of sv_maxclients, reserved for players with the server password (sv_privatePassword) - Holesinswiss
sv_privatePassword ""	set password for private clients to login with
sv_punkbuster "0"	enable the punkbuster server in the server creation menu
sv_pure "1"	disallow native DLL loading if sv_pure, requires clients to only get data from pk3 files the server is using "John Carmack"

sv_reconnectlimit "3"	number of times a disconnected client can come back and reconnect
sv_referencedPakNames ""	variable holds a list of all the pk3 files the server loaded data from. these pk3 files will be autodownloaded by a client if the client does not have them. "baseq3/pak2 baseq3/pak0"
sv_referencedPaks ""	variable holds the checksum of the referenced pk3 files
sv_running "1"	variable flag tells the console weather or not a local server is running
sv_serverid ""	hmm…"8021204"
sv_showloss "0"	toggle sever packet loss display
sv_strictAuth "1"	server side variable to control wether strict CDKEY authentication should be performed with the authentication server this is required if you want reliable cl_guid for the server
sv_timeout "120"	sets the amount of time for the server to wait for a client packet before assuming a disconnected state.
sv_zombietime "2"	the amount of time in minutes before a frozen character is removed from the map.
sv_zone "default"	this is the keyword that clients will search for, server admin's should set this variable to the gametype they have running. free for all, tournament, team deathmatch, and CTF I do not know if you can deviate from the keywords the way Zaphod laid them down in the whatsnew.txt

D.13 Variables de les opcions de l'interfície d'usuari

ui_bigFont "0.4"	
ui_browserGameType "0"	set server search game type in the browser list (see g_gametype)
ui_browserMaster "0"	set server search 0=LAN 1=Mplayer 2=Internet 3=Favorites - WeeJoker
ui_browserShowEmpty "1"	toggle the displaying of empty servers in the browser list
ui_browserShowFull "1"	toggle the displaying of full servers in the browser list
ui_browserSortKey "4"	set the field number to sort by in the browser list 0=Server Name 1=Map Name 2=Open Player Spots 3=Game Type 4=PingTime
ui_cdkeychecked "1"	set to a 1 after the cdkey has been checked so won't ask again
ui_ctf_capturelimit "8"	set the menu default capture limit for single player bot matches
ui_ctf_friendly "0"	toggle team mate damage in single player CTF bot matches
ui_ctf_timelimit "30"	set the menu default CTF time limit for single player bot matches
ui_ffa_fraglimit "20"	set the menu default frag limit for single player FFA bot matches
ui_ffa_timelimit "0"	set the menu default time limit for single player FFA bot matches
ui_master "0"	set server search 0=LAN 1=Mplayer 2=Internet 3=Favorites - WeeJoker
ui_singlePlayerActive "0"	
ui_smallFont "0.25"	
ui_spSelection "2"	set the menu default gametype of single player? 16 = CTF 2 = FFA DM

ui_team_fraglimit "0"	set the menu default frag limit for single player team bot matches
ui_team_friendly "1"	toggle default team mate damage in single player team bot matches
ui_team_timelimit "20"	set the menu default time limit for single player team bot matches
ui_tourney_fraglimit "0"	set the menu default frag limit for single player tourney bot matches
ui_tourney_timelimit "15"	sets the menu default time limit for single player tourney bot matches

D.14 Altres

activeaction ""	variable holds a command to be executed upon connecting to a server
arch "win98"	architecture/operating system
capturelimit "8"	set # of times a team must grab the others flag before the win is declared
cheats "0"	enable cheating commands (give all) (serverside only)
cm_curveClipHack "0"	must have been a cheat!!! removed now
cm_noAreas "0"	toggle the ability of the player bounding box to clip through areas?
cm_noCurves "0"	toggle the ability of the player bounding box to clip through curved surfaces
cm_playerCurveClip "1"	toggles the ability of the player bounding box to respect curved surfaces.
color "1"	rail trail color blue/green/cyan/red/magenta/yellow/white respectively 1/2/3/4/5/6/7

color1 "2"	spiral rail trail color spiral core - special thanks to schiz Jax_Gator Dekard blue/green/cyan/red/magenta/yellow/white respectively 1/2/3/4/5/6/7
color2 "5"	spiral rail trail color spiral ring - special thanks to schiz Jax_Gator Dekard blue/green/cyan/red/magenta/yellow/white respectively 1/2/3/4/5/6/7
con_notifytime "3"	defines how long messages (from players or the system) are on the screen
conback ""	select console background file "gfx/2d/conback.tga"
crosshairhealth "1"	show health by the cross hairs
crosshairsize "24"	crosshair size...incase you have crosshair envy (c:
debuggraph "0"	
dedicated "0"	set console to server only 0 is a listen, 1 is lan, and 2 is internet (command line cvar causes engine not to load 3D game just a server console C:\Q3TEST\quake3.exe +set dedicated 2) - Dekard
developer "0"	enable developer mode (more verbose messages)
dmflags "0"	set deathmatch flags originally I posted the values of Quake 2 dmflags but have since tested them and most of them don't work
disable_<item name>	this command allows the administrator of a server to disable a particular item from the map. as an example: "/set disable_weapon_bfg 1" will make it so that the bfg does not show up. changing the value back to 0 and executing a /map_restart command will bring the disabled item back.
fixedtime "0"	toggle the rendering of every frame the game will wait until each frame is completely rendered before sending the next frame

fov "90"	field of view/vision "90" is default higher numbers give peripheral vision.
fraglimit "20"	set fraglimit on a server (0 is no limit)
Freelook "1"	steer aim and control head movement with the mouse…a must (c:
gamedate ""	Sep 30 2002
gamename "baseq3"	display the game name for TC's basedir would be other than baseq3
gun_frame "0"	turns off weapon animation and displays specified frame in the weapons animation sequence 0=animate 1 and up step through frames...(c:
gun_x "0"	set the x location of the gun model (one is up and down one is side to side)
gun_y "0"	set the y location of the gun model (one is up and down one is side to side)
gun_z "0"	set the z location of the gun model (possibly angle?)
handicap "100"	set player handicap (max health), valid values 1 - 99
headmodel ""	changes only the head of the model to another model Example: If you are playing as the Grunt model, /headmodel "sarge" will stick Sarge's head on Grunt's body selecting a new model will load both the model and its matching head
host_speeds "0"	toggle the display of timing information sv=server cl=client gm=gametime rf=render time all=total time
mapname ""	display the name of the current map being used
memorydump "0"	possibly used for debugging memory allocation/use?
maxfps "0"	set the max frames per second the server should send you
model "visor/blue"	set the model used to represent your player Hey John a 3D Keen model would be nice…(c:
name "Commander Keen"	pick your own be original (no Player)

nextmap ""	variable holds the name of the next map in the server rotation
nextdemo ""	variable holds the name of the next demo in a demo loop
nohealth "0"	toggle the use of health items on next map or do it now from the command line
password ""	set password for entering a password protected server
port "27960"	set port number server will use if you want to run more than one instance of Q3A server on the same machine
paused "0"	possible to allow the game to pause while in single player mode
pmove_fixed "0"	typically the player physics advances in small time steps. when this option is enabled all players will use fixed frequency player physics, the time between two advances of the phsysics will be the same for all players. the actual time between two advances of the player physics can be set with the pmove_msec variable. enabling this option will make the player physics the same for all players independent from their framerate. should do what you want for prediction and should even out the machine dependent rates. - Robert Duffy
pmove_msec "8"	set the time in milliseconds between two advances of the player physics. should do what you want for prediction and should even out the machine dependent rates. - Robert Duffy
protocol "68"	display network protocol version. Useful for backward compatibility with servers with otherwise incompatible versions < maddog read only
qport "59337"	set internal network port. this allows more than one person to play from behind a NAT router by using only one IP address – Questy

rate ""	modem speed/rate of data transfer "4500" (take a zero off the end of your connection speed?)
rcon_password ""	set password for remote console control of the server removed cause dupe
rconAddress ""	variable holds IP address of the server for rcon
rconPassword ""	set password for remote console control of the server
scr_conspeed "3"	set how fast the console goes up and down
server1 ""	holds IP/URL of a servers from the favorite servers list
server2 ""	holds IP/URL of a servers from the favorite servers list
server3 ""	holds IP/URL of a servers from the favorite servers list
server4 ""	holds IP/URL of a servers from the favorite servers list
server5 ""	holds IP/URL of a servers from the favorite servers list
server6 ""	holds IP/URL of a servers from the favorite servers list
server7 ""	holds IP/URL of a servers from the favorite servers list
server8 ""	holds IP/URL of a servers from the favorite servers list
server9 ""	holds IP/URL of a servers from the favorite servers list
server10 ""	holds IP/URL of a servers from the favorite servers list
server11 ""	holds IP/URL of a servers from the favorite servers list
server12 ""	holds IP/URL of a servers from the favorite servers list
server13 ""	holds IP/URL of a servers from the favorite servers list
server14 ""	holds IP/URL of a servers from the favorite servers list
server15 ""	holds IP/URL of a servers from the favorite servers list
server16 ""	holds IP/URL of a servers from the favorite servers list
session "2"	possibly holds the value for the active session number when running multiple addresses and sockets for multiple servers on one machine?
session0 ""0 300 1 0 0 0"	possibly to set up multiple addresses and sockets for multiple servers on one machine (can you say BFServer with multiple processors?)
session1 "0 300 1 0 0 0"	possibly to set up multiple addresses and sockets for multiple servers on one machine (can you say BFServer with multiple processors?)

session2 "0 300 1 0 0 0"	possibly to set up multiple addresses and sockets for multiple servers on one machine (can you say BFServer with multiple processors?)
session3 "0 300 1 0 0 0"	possibly to set up multiple addresses and sockets for multiple servers on one machine (can you say BFServer with multiple processors?)
session4 "0 300 1 0 0 0"	possibly to set up multiple addresses and sockets for multiple servers on one machine (can you say BFServer with multiple processors?)
sex "male"	set gender for model characteristics (sounds, obituary's etc.)
showdrop "0"	toggle display of dropped packets. 0=disables,1=toggles.
showpackets "0"	toggle display of all packets sent and received. 0=disables,1=toggles.
showtrace "0"	toggle display of packet traces. 0=disables,1=toggles.
snaps "20"	set the number of snapshots sever will send to a client (server run at 40Hz, so use 40, 20, or 10) -Randy
snd "visor"	select which model sounds your player uses (mix it up)
sys_cpuid "33"	more snooping into your CPU
sys_cpustring ""	variable holds a string that identifies your processor
team_headmodel ""	set head of team_model to a head that will only be used during team game play
team_model ""	set player model that will only be used during team game play
teamoverlay "0"	toggle the drawing of the colored team overlay on the HUD
teamflags "0"	set flags for team play (probably will be a hex value like deathmatch flags)
teamtask "0"	variable holds the number of the team task you are currently asigned 1- offense 2– defense 3- point/patroll 4- following 5- retrieving 6- escort 7- camping

timedemo "0"	when set to "1" times a demo and returns frames per second like a benchmark
timegraph "0"	toggle the display of the timegraph. .
timelimit "0"	amount of time before new map loads or next match begins
timescale "1"	set the ratio between game time and real time
username "vern"	variable holds your network login id from %username% env variable…hmmm? id hackers!
version ""	Q3 1.32 win-x86 Oct 7 2002
versionNumber ""	"Q3T 1.08"
vid_xpos "30"	x position when windowed
vid_ypos "30"	y position when windowed
viewlog "0"	toggle the display of the startup console window over the game screen
viewsize "100"	changes view port size 0 - 100 (you probably wouldn't want less than 100)
vm_cgame "0"	part of the virtual machine interpreter which allows PC MOD makers to not have to know MAC code and MAC MOD makers to not have to know PC
vm_game "0"	toggle the virtual machine interpreter, cgame can switch between being loaded as a binary .dll or an interpreted .qvm at the change of this cvar
vm_ui "0"	part of the virtual machine interpreter which allows PC MOD makers to not have to know MAC code and MAC MOD makers to not have to know PC
win_hinstance ""	address of the handle instance of quake3 under windows - LOKi
win_wndproc ""	the address of the function that receives messages from windows (4368704) - MeTaL125
zoomfov "22.5"	what the zoomed in field of view will be any thing more than 30 would not be sniper friendly