Per la serie "E se pensate che sia stato facile, cercate il DOS Extender... non lo troverete!".

Kaos v1.0

(per quelli che non devono chiedere... i sorgenti)

(per i duri della programmazione)

(per gli hacker e gli smanettoni)

(per quelli che il Software...)

(ci siete ancora?)

(Beh, facciamo per tutti quelli che apprezzano)

File multimediale da leggere a 1024x768 caratteri grandi (17,6 cm di righello), senza zoom e con le scarpe allacciate a livello d'Internét.

Questo file è stato scritto dall'analista-programmatore di Kaos (Valentini Domenico) per fare un po' di propaganda e presentare le caratteristiche di questo banale gioco, ma non certo semplice programma.
(rileggete pure, ma non credo che il problema sia nella visualizzazione dei caratteri, o in errori di stampa... l'ho proprio scritto così).

Dato che il nome del Kaos non dice molto, proverò a fare un rapido quadro delle sue caratteristiche.
A livello di gioco, Kaos è un misto di azione e terrore 3D e si svolge in un castello medievale popolato da mostri.
I comandi relativi al giocatore sono stati calibrati in modo da rendere l'azione un pò più veloce del normale.
Le caratteristiche globali possono essere così riassunte:

- **programmazione ad oggetti (C++)** per gli elementi fondamentali di gioco, cioè grandi possibilità di estensione del programma;
- grafica texture mapped in VGA 256 colori;
- **pavimenti e soffitti con texture variabili**;
- **sistema di gestione dell'illuminazione in tempo reale**, effettuato direttamente dagli oggetti luminosi;
- muri di diversi tipi e forme (anche in diagonale), porte (anche diagonali) che si aprono in modi diversi, sbarre, ecc.;
- gestione di oggetti di grandezze variabili e regolazione automatica del punto di vista, con numero illimitato di facciate;
- grande varietà e libertà di creazione di oggetti animati (già disponibili mostri con comportamenti intelligenti molto differenti);
- visione esterna nei punti dove non sono presenti pavimenti o soffitti;
- capacità di ridurre o ampliare la finestra di gioco in tempo reale;
- possibilità di guardare in alto e in basso (nel gioco è utilizzato maggiormente per abbassare/alzare lo sguardo quando si viene colpiti);
- punto di vista ad altezza qualsiasi (nel gioco è utilizzato per ottenere la camminata a balzi (attenuata come in Quake) e per saltare dopo le esplosioni;
- **capacità di gioco in MULTIPLAYER (sullo stesso computer!) fino a 3 giocatori contemporaneamente** con visioni in soggettiva;
- **16 canali indipendenti per suoni digitalizzati a 11,025KHz** (usando inizialmente file .WAV);
- **suoni** veramente **3D** (generati con lo sfasamento delle onde);
- efficiente gestore di eventi per lo scambio di messaggi tra gli oggetti e tra oggetti e sistema principale;
- avanzato gestore di memoria (convenzionale ed XMS), con capacità di swapping in memoria e utilizzo dei dati caricati direttamente dal disco (tutto in modalità reale) (utilizza solo il file di sistema HIMEM.SYS);
- sistema di temporizzazione per mantenere costante la velocità di gioco, anche su computer meno potenti.

Comincio con elencare i principali moduli del programma, che poi sono quelli che fanno la differenza (e sono quelli ai quali tengo di più):

- **LGraph** (è insieme delle routines di gestione della modalità video 0x13 (13h) liscia);
  Lo so, lo so è una modalità del cavolo, ma è veloce. Prima c'era la XGraph (è la MIA, ma credo che tutti i programmatori del mondo l'avrebbero chiamata così) per gestire la modalità X "dura e pura" con minimo 4 pagine e poi la 320x400 GRATIS, ma quando ho cominciato a disegnare i pavimenti (in orizzontale) "m'è schiattato er Pentium e 'a scheda video" (e non era colpa mia!);
- **EventManager** (in realtà è fin troppo banale, ma è usato in un modo un po' ambiguo);
- **MemoryManager** (per ora gestisce la memoria XMS e il disco);
- **3D_SoundManager** (qui ciò messo il cuore, il cervello e il ... beh sì è stata dura);

e poi il solito...

- **3D-Ɛngine** (forse ai tempi di Wolf3D si sarebbe detto: "WOW !");

...poi ci sarebbero il gestore della tastiera, della palette (per animazioni, fade in/out, morphing dei colori), degli oggetti, ecc... ma per quelli rimando alla prossima puntata.

La storia del Kaos.

(se, come per me, non vi piace la storia, allora saltate al prossimo paragrafo)

L'inizio della programmazione del Kaos risale al Novembre del 1995 e, come è ovvio in qualsiasi software-story che si rispetti, il nome del programma (allora solo motore grafico) non era Kaos, ma Nic3D e si prefiggeva di raggiungere, utilizzando tecniche risolutive simili a quelle di Ack3D (vedi Internet), la precisione e l'alta velocità di disegno di Wolf3D. Non stupitevi di questo, perchè a quel tempo il potentissimo mezzo di sviluppo disponibile era un 286 a 10MHz e Wolf3D era uno dei giochi più belli e veloci, capaci di funzionare in quelle condizioni!
Non appena il motore grafico è uscito (naturalmente più lento dell'intero Wolf3D), si è pensato (Io & Max) di farci un gioco, e da lì sono cominciati i problemi...
Il prodotto finale doveva essere un "simulatore di barbone", e infatti il nome del gioco sarebbe stato Barboon (bello vero?). Barboon era ambientato in una stazione (stazione Termini di Roma) e prevedeva la visione in soggettiva di un barbone che vagava alla ricerca di cose da prendere, tra le quali anche un carrello! Da qui il gioco divenne "Barboon - the trolley simulator (il simulatore di carrello)", il cui scopo era quello di investire il numero più grande di persone presenti nella stazione, evitando gli attacchi degli altri barboni e della polizia. Il progetto Barboon proseguì, rendendo possibile il miglioramento del 3D engine in modo da utilizzare il ray-casting one-shot che avevo già progettato e che è stato portato (data l'efficienza) anche nell'ultima versione. Il problema più grave di questo motore grafico era dato dal fatto di non poter gestire oggetti sovrapposti (es.: le torce e i mostri), oltre al fatto di non poter visualizzare pavimenti e soffitti e nemmeno poter creare muri di forme diverse.
Dopo una terrificante lotta per la definizione di obiettivi sicuri (le specifiche cambiavano a distanza di 2 giorni), non appena si è potuto beneficiare dei vantaggi dei 32 bit nella fase di programmazione, ma soprattutto all'interno del programma stesso, il risultato ebbe un salto di qualità soprattutto nella precisione di calcolo, che si tradusse in un'ottimo risultato visivo, una maggiore fluidità nei movimenti e, non appena inseriti pavimenti e soffitti, anche in un rallentamento globale del motore grafico: ecco Kaos v0.1ß.
Da qui alla versione shareware c'è stato solo un lavoro di ottimizzazione su tutti i fronti.

Informazioni generali:

Innanzitutto premetto che a me questo tipo di giochi non piace proprio, preferisco qualcosa di un po' più statico e riflessivo (un gioco di ruolo?), ma allo stesso tempo sono affascinato dalle tecniche di risoluzione dei problemi inerenti il 3D (da questo punto di vista Quake è un "grande").
Non sono molto orgoglioso del 3D-Ɛngine, ma è pur sempre un bel pezzo di software; e dire che ad un certo punto volevo mandare tutto al diavolo per gestire un BSP 2D come Doom...

Questo programma è stato scritto in C/C++/Assembler ed alterna fasi di genio e di demenzialità di analisi (sembra quasi il "C") dovute più che altro al fatto che questo è il mio primo programma in C/C++ (e supera le 25K linee di codice !) ed è stato scritto in parte prima e in parte dopo l'esame di Informatica II. Come già avreste potuto leggere, per ottenere l'attuale risultato si è dovuto passare per altre due versioni: la prima non prevedeva pavimenti, suoni 3D, oggetti sovrapponibili, insomma la solita pappa che trovi nella directory "appoggio\shit" di ogni programmatore; la seconda non prevedeva giochetti strani con i muri, oggetti particolari... insomma, due schifezze.
Posso affermare con certezza che questa versione le supera tutte, anche perché è stata l'unica ad oltrepassare la versione beta.

Attenzione! Questo NON è un esempio realizzato con una qualche libreria 3D per principianti, ma un vero e proprio gioco, con codice più o meno ottimizzato, che si rivolge, almeno dal mio punto di vista, ad altri programmatori che vogliono prendere degli spunti, o magari rimetterci le mani sopra e farci qualcosa di migliore! A proposito, colgo l'occasione per dire ai programmatori Pascal che, anche se sono forti, si fanno il DOS Extender da soli, programmano il 70% del codice in ASM, raggiungono i 50fps con un 3D-engine, ecc. è sempre meglio il "C++" ! (E ve lo dice uno che di programmi in Pascal (anche ad oggetti) ne ha fatti a migliaia, e che giravano bene su un 286 con 1Mb di RAM).

Come avete potuto leggere nella prima riga, questo programma funziona in modalità reale... eh, sì
MA NON PER SCELTA ! Se un giorno potrò permettermi il lusso di acquistare un compilatore serio (tutto a 32 bit e con le ottimizzazioni varie) forse perderò più tempo a progettare cose serie, invece di inutili gestori di memoria che tra l'altro sono anche difficili da testare, perché sappiate che, quando non funzionano, distruggono sempre qualcosa di utile in memoria; è una questione di "sfiga": su 640Kb beccano sempre prima quei 4Kb che non devono essere toccati!

MULTIPLAYER
Per ora il gioco non funziona in rete o altro, ma sto sviluppando questa parte e forse ne uscirà qualcosa (premetto che io non ho ancora il modem (eresia!)). Per far vedere che comunque la possibilità c'è, sono state aggiunte delle opzioni che permettono di giocare (anche in tre) sulla stessa tastiera con lo schermo diviso (economico è? Sì, ma sai che ammucchiata!) e, per non creare il Kaos, solo il 1° giocatore potrà usufruire dei suoni 3D (ma questa è stata una mia discutibile scelta... e logicamente non è valida per la modalità "in rete").

Non starò qui a fare la "traduzione in italiano" del programma, ma voglio esporre le capacità e le caratteristiche di ogni singolo modulo:

LGraph
Sono anni ormai che mi fisso sulle routine "low-level" per la gestione delle modalità grafiche, e sono sicuro di aver sfornato delle routine tra le più veloci realizzabili con le istruzioni del 2/386. Qui dentro c'è tutto ciò che mi è servito (e non) per fare il gioco (ma non parlo di 3D !).
Siccome io sono un patito della modalità X, ho anche una XGraph, molto più bella, più veloce nei trasferimenti video, più adatta a gestire le alte risoluzioni, più fornita di funzioni, più generosa dal punto di vista della memoria utilizzata, più lenta a disegnare per linee orizzontali!
In realtà, esistono le modalità standard (VESA) per la gestione delle alte risoluzioni in SVGA in modo lineare (senza X), ma non ho avuto modo di provarle, anche perché dovrò (prima o poi) comprare un compilatore C/C++ a 32 bit (se avete suggerimenti...), e allora cambierebbe davvero tutto.

EventManager
Non c'è assolutamente niente da dire, è il più semplice e completo (per ciò che gli compete) gestore di eventi che io abbia mai fatto. Pur essendo stato progettato per un uso generale, non soffre di problemi di lentezza, e lega bene con il gioco.

The MemoryManager
Da come l'ho presentata dovrebbe essere il nucleo di tutto, ma in realtà è una parte di programma che è nata come appoggio al motore grafico e funziona INSIEME alla normale gestione della memoria, nel senso che quando è opportuno (ah, ah, ah) si utilizzano le funzioni del MemoryManager, e quando non lo è si usano quelle standard (più o meno). Questo gestore di memoria permette di maneggiare dinamicamente dei blocchi di dati, in modo che possano essere sbattuti in memoria XMS quando non sono usati. C'è anche la possibilità di "bloccare dei blocchi" in memoria convenzionale (non i suoni, sono troppo grossi!) e di prendere i dati dal disco nel caso che non ci sia l'XMS necessaria per lo swapping. Il bello di tutto ciò è che non ci sono limitazioni particolari per l'allocazione di questo tipo di blocchi, ed è molto facile "attaccarlo" a qualsiasi altro programma (infatti l'editor dei livelli funziona con questo gestore), e soprattutto non costa dal punto di vista del tempo di

elaborazione, perché è fatto in modo da essere veloce (per quanto è possibile). C'è da notare che la memoria può essere allocata anche in modo standard (statica e in memoria convenzionale) e non ci sono comunque problemi per quanto riguarda l'ordine di allocazione di questi.
Per quanto riguarda l'acquisizione dei dati da disco, il MemoryManager dispone di 2 modalità di funzionamento:
1) Normale: è il programma a stabilire l'allocazione della memoria, richiedendola man mano che vengono preparati i dati;
2) DataFile: se in modalità Normale è stata chiamata la funzione "makedatafile()", tutta la struttura dei blocchi e il loro contenuto sono stati memorizzati in un "Big Data File" (per ora in formato Raw, senza compressione), che verrà appunto utilizzato in questa modalità.
Il file ".BDF" (praticamente enorme) verrà utilizzato come unica sorgente di informazioni, evitando la presenza di numerosi altri files di dati, e sarà possibile utilizzarlo, in caso di mancanza di memoria XMS, per l'acquisizione diretta da disco, perchè conosce già le posizioni dei blocchi all'interno file.
"E' possibile chiedere in qualsiasi momento uno spazio di memoria al MemoryManager, e lui la troverà".

All'interno del MemoryManager troverete The Wizard (er mago) che si occupa di CREARE la memoria che non c'è.

3D_SoundManager

Secondo me questo è il modulo più interessante di tutti, forse perché è stato voluto solo ed esclusivamente da me, in alternativa al solito gestore di suoni stereo. La tecnologia usata si basa praticamente su dati empirici, in altre parole è stata progettata sulle sensazioni di un campione alquanto ristretto di persone, e non fa riferimento ad alcun testo sull'argomento (anche perché non avevo voglia di leggermi disquisizioni su trasformate di Fourier o altre cose devianti anche se interessanti).
Il 3D_SoundManager gestisce fino ad un massimo di 16 suoni contemporaneamente (ma basta cambiare il numero...) e, utilizzando la modalità DMA di una qualsiasi SoundBlaster stereo (Pro, 16, 32, 64, e compatibili), elabora, mixa e poi suona in perfetto background (senza scatti). Per questo gioco ho dovuto aggiungere qualche funzione di sincronizzazione dei suoni con le animazioni dei mostri, ma in linea generale è possibile generare 2 tipi di suoni:
- suoni 3D, per i quali c'è l'aggiornamento dinamico degli effetti (volume e direzione) direttamente in fase di mixing;
- suoni FISSI, per i quali non si ha variazione degli effetti (stabiliti prima di cominciare a suonare), ma rimane pur sempre la possibilità di suono tridimensionale.
È poi possibile aggiungere caratteristiche al suono:
- continuità: il suono viene ripetuto all'infinito finché non si impartisce un comando di "stop";
- autostop: il suono muore non appena muore chi lo ha suonato (Don't worry! Questo non significa che c'è una coesione disperata con il gioco, ma solo che il 3D_SoundManager "ascolta" i messaggi di gioco, ed è predisposto a farne uso)
- autofocus: il suono 3D segue il suo generatore;

Ma se non è stereo che è?
E' semplice, invece di utilizzare solo i volumi (separati) per ottenere la direzionalità del suono, ho provato a sfasare i canali destro e sinistro, ottenendo un ritardo di emissione dell'uno rispetto all'altro, e questo crea (a quanto pare) un effetto 3D. Non che l'abbia scoperto io! Già si sapeva, ed è anche più complicato di come l'ho fatto io, ma provate a tarare il tutto senza avere neanche un numero in mano! E non crediate che sia facile farlo bene anche solo in modo stereo liscio!

Attenzione! Questo NON è un effetto Surround, ma volendo modificare leggermente il codice si ottiene anche quello.

Per mezzo di una velocissima, quanto rozza, routine di ray-checking ('sta parola non esiste, l'ho inventata) il 3D_SoundManager riesce a regolare l'intensità del suono controllando, oltre alla distanza e al processo di desincronizzazione, anche il numero di oggetti incontrati in linea d'aria dal suono; in questo modo è possibile ottenere effetti più realistici durante la chiusura di porte (o passaggi segreti), al passaggio dei mostri davanti alla sorgente sonora, ecc. Con questo metodo è anche realizzabile un'approssimativa routine per l'emissione degli echi, ma non mettiamo troppi bit sul processore!

3D-Ɛngine

Questo è veramente il nocciolo del gioco, anche perchè se non c'è questo → non c'è il gioco.
Nel 3D engine entra un banalissimo giochino stile PAC-MAN e ne esce un bellissimo e realistico gioco di terrore (mmah... troppo facile).
La scelta degli obiettivi del 3D-Ɛngine è stata forse la parte più difficile della progettazione dell'intero gioco, perché si rendeva necessario valutare gli aspetti di velocità/qualità_finale/memoria prima ancora che gli obiettivi fossero ben definiti.
Dall'X-mode, a causa degli ultimi rallentamenti, dovuti principalmente all'introduzione di pavimenti & soffitti, è stato necessario portare il tutto alla modalità grafica più facile e "veloce": 320x200x256col. liscia.

Il risultato finale è un motore grafico con le seguenti caratteristiche:

MURI
Gestisce diversi tipi e forme di elementi verticali:

- Tipi
  - Muro;
  - Porta ad apertura laterale;
  - Porta ad apertura centrale;
  - Grate verticali (passo 4-8);
  - (paesaggio esterno)
- Forme
  - Quadrato;
  - Superficie x;
  - Superficie y;
  - Diagonale 1;
  - Diagonale 2;
  - Superficie auto-orientante (per le porte);

(e se siete forti in matematica potrete anche aggiungere i muri circolari (colonne), le curve, i cancelli...).

PUNTI DI VISTA VARIABILI (ma veramente)
Una possibilità che non è stata sviluppata è data dal fatto di poter animare le texture con dei televisori e poterli usare come punti di vista (come Duke 3D).
Data la generalità delle routine grafiche è possibile anche cambiare le dimensioni della finestra di visualizzazione in tempo reale e modificare persino il range di visione, che nel gioco è fissato a 90° (come Doom)... ma dovreste vedere a 120°! La vecchia versione funzionava a 60° (come Wolf3D, credo) ed era più realistica di questa, ma non permetteva nemmeno di guardare 2 strade di un incrocio contemporaneamente.
C'è anche la possibilità di guardare in alto e in basso, ma anche questa cosa non è molto utile in un gioco che si svolge interamente su un solo piano, tranne che per guardare un mostro gigante in faccia.

Ehi, sto guardando leggermente in alto!
(forse i colori che vedrete non sono quelli originali, ma i muri sono indubbiamente più alti)

Per quanto riguarda i pavimenti, non è possibile (se pò fa, se pò fa) farli di diverse altezze, ma è possibile cambiare le texture una per una (in teoria, più ne metti di diverse nello stesso ambiente e più tempo ci vuole a disegnarle). Se qualcuno vuole provarci, posso anche suggerirgli come mettere i pavimenti ad altezze diverse

(potrei farlo addirittura io nella versione finale...), ma garantisco che il disegno dei piani orizzontali è la parte del gioco che richiede più tempo di elaborazione. Dato che il gioco funziona tutto sullo stesso piano (come Wolf3D, e non oso certamente dire Doom) si può provare a rendere un po' più maestoso l'ambiente raddoppiando l'altezza del soffitto e quindi duplicando i muri (c'è anche questo nei sorgenti...), ed in questo modo si velocizza anche il tutto (sono funzioni alternative), oltre a sembrare più bello (per me). Ma allora perché non è stato fatto per funzionare così? E' semplice, per fare le cose in modo serio, bisognerebbe gestire 2 piani di textures (diverse), quindi 2 raggi (ray-casting), quindi "tempo"! Volendo rinunciare ai 2 raggi diversi, bisognerebbe comunque considerare dei casi particolari... ma a questo punto è meglio fare un Doom, che è pure più veloce e meno rattoppato!

Ho provato ad imitare Quake... ma è solo un sogno

Per quanto riguarda gli oggetti (Objects o Actors che siano), questi possono essere di dimensione qualsiasi, ma l'attuale implementazione garantisce movimenti precisi solo per mostri che non siano più larghi della dimensione di una texture (64 pixel originari) ma comunque di altezza qualsiasi. Per quanto riguarda le collisioni, il processo è gestito da routines ad alto e basso livello, il che permette un'ampia gamma di tecniche di movimento, anche ridefinibili (come è stato fatto per i missili) in modo da utilizzare le routines di ray-casting del 3D-Engine.

Dato che è stata mantenuta una certa compatibilità con le vecchie X-routines, potrei anche diffondere una versione riaggiustata per la modalità X, per raggiungere la risoluzione 320x400 senza dolori (tranne che per il processore). Il problema sono solo i pavimenti e i soffitti, che vengono disegnati per linee orizzontali, e ciò non costituisce una scelta molto felice per questa modalità grafica. Quando troverò una soluzione più veloce di quella attuale che agisca per linee verticali allora potrò tranquillamente accantonare la LGraph e guadagnare anche ~64Kb di memoria convenzionale.

QUALITA'
Le caratteristiche positive sono:
- precisione nella visualizzazione (anche la correzione dell'effetto "occhio di pesce" funziona senza compromessi, ma tanto avevo già trovato un'altra soluzione);
- presenza di pavimenti e soffitti;
- gestione degli effetti luce in modo veloce e non vincolati alla mappa;
- alta velocità di ray-casting (Assembler 386) con riconoscimento diretto degli oggetti visibili (C) tramite informazioni insite negli oggetti stessi (C++). La struttura globale occupa così poca memoria, ed è dinamica;
- alta velocità di shading grazie ad un piccolo e veloce algoritmo (Assembler) per il calcolo del colore. Ciò è stato fatto per evitare l'uso di un'ulteriore tabella (ce ne sono già tante) da 256x15 byte. Questo piccolo algoritmo funziona solo con una Palette ordinata in un certo modo e, considerando i problemi che questo può portare ad una persona che sviluppa grafica anche in 3D-Studio, devo ringraziare Massimo per aver adattato tutto al meglio.
- espandibilità di tipi e forme di muro;
- ecc.

LIMITAZIONI
I limiti, dettati da scelte più o meno storiche, sono questi:
- non è possibile creare mappe più grandi di 64x64 blocchi (Wolf3D);
- non è possibile far sovrapporre più di 15 oggetti (ma questa non è una vera limitazione);
- non ci sono i muri trasparenti o bucherellati (ma niente vieta di modificare la routine con un piccolo buffer...);

- non ci sono gli specchi (e questa è una vera limitazione, perché anche se è facilmente risolvibile per i muri, bisogna sempre considerare pavimenti e soffitti al contrario!);
- c'è un solo piano (veramente piano) di gioco;
- il soffitto è basso;
- ecc.

Qualche nota per gli amanti dell'impossibile.

Vi piacciono le finestre?
Per voi, ma anche per quelli che hanno un P7 a 1400MHz con scheda video ad 1GHz di refresh e ci mettono Windows95™, posso affermare con sperimentale certezza che il gioco gira quasi perfettamente in una finestra DOS di Windows95™, anche se ad una velocità da fa ridere i polli.
Come si imposta:
- far partire il gioco;
- quando si è entrati nella modalità grafica, premere il tasto di Windoze95 (o equivalenti);
- clickare col pulsante sinistro sul Desktop e poi col destro sul rettangolo di Kaos sulla barra delle applicazioni;
- scegliere Proprietà, poi la cartella Schermo e selezionare "Finestra" invece di "Schermo intero".
  (non so perché, ma consiglierei di disabilitare anche l'opzione "Memoria dinamica").

(a proposito, propongo di cancellare i file nascosti "user.*" per sfidare il S.O. ... scherzavo, forse)

...e io non te la do (la memoria).
Come avrete certamente notato, potete avere anche 10000Mb di XMS ma il gioco usa sempre e solo i suoi 4Mb, e niente più. Se provate a "fregare" anche su quei miseri 4Mb, scendendo al di sotto di 2, il gioco potrebbe cominciare ad utilizzare il disco (ma non rallenta poi tanto se le condizioni sono adatte). Se invece provate a privarlo della memoria convenzionale, allora sì che vedrete i limiti del MemoryManager !

Tips, tricks & traps.
La versione del gioco che vedrete è eccessivamente precisa nel disegno delle texture dei muri, tanto che se provate a sbattere con la faccia su uno di essi non noterete nessun effetto di zig-zag verticale, questo perché l'aritmetica usata è a 32 bit, con ben 20 bit per i decimali! Nel programma giace comunque la routine standard a 16 bit (per i decimali) ma, anche se è decisamente più veloce, fa un po' schifo (e io a queste cose ci tengo).
Per il pavimento non ci sono grandi problemi, tranne per il fatto che per determinate angolazioni si vede "qualche puntino" all'attaccatura con i muri (ho provato a toglierlo ma, a quanto pare, non gli ho dedicato il giusto tempo).
La routine che disegna gli oggetti in scala è veloce (ma non troppo) e precisa, ma poteva essere tranquillamente sostituita da un'altra, anch'essa inclusa ma inutilizzata, che è meno precisa (ma questo non si vede) e molto più veloce, anche per il fatto che permette di risparmiare moltiplicazioni a 32 bit in fase di calcolo dei parametri.
Una veramente bella, veloce e precisa è la funzione di disegno dello sfondo, che per mia precisa scelta non dà l'impressione di guardare da una palla di vetro come in DOOM (ma avrei potuto farla uguale, consumando un pò più di memoria); faccio notare che tutte le cose disegnate nel quadro di gioco devono poter essere visualizzate in scala (!), rimanendo così coerenti con la grandezza della finestra, la quale può essere ridotta o allargata in qualsiasi momento.

La temporizzazione.
Questo è il punto dolente, forse. Questo gioco è temporizzato per generare al massimo 36fps (fotogrammi al secondo) il che significa che se avete un Pentium Pro a 200MHz lo vedrete come su un Pentium a 166MHz (a parità di lavoro "concorrente"). Forse il paragone fa un po' schifo (anche per il fatto che il codice non è a 32 bit) ma penso che il messaggio sia stato recepito.
Naturalmente se ad un certo punto ci sono 16 suoni 3D che svolazzano contemporaneamente nel 3D_SoundManager, il sistema potrebbe rallentare, e in questo caso un computer più veloce farebbe la differenza.
(garantisco che il mixer del 3D_SoundManager, anche non essendo scritto interamente in Assembler, è più veloce di quello di molti altri giochi).
C'è da notare che la natura stessa del MemoryManager può provocare rallentamenti nel momento in cui si cambia completamente ambientazione all'interno dello stesso quadro di gioco.

Se al computer "nun ie regge a pompa", allora il sistema interno di regolazione automatica del tempo dedicato all'utilizzo della CPU per operazioni di visualizzazione e operazioni di bassa priorità per eliminazione dei rallentamenti dovuti a cali di prestazioni del sistema (aaahhhh !), praticamente una "if", si preoccupa di evitare la visualizzazione di alcuni fotogrammi, in modo da mantenere costante il ritmo di gioco. L'effetto che si ottiene è sì quello di dedicare lo stesso tempo alle animazioni, ma è anche quello di rendere paranoico il giocatore umano.


Double-click me!

Perché, esistono anche giocatori non umani?

Noi siamo scienza e non fantascienza, per questo posso rispondere: "Sì".

Per la particolare struttura del 3D-Ɛngine e del sistema di gestione degli oggetti, è anche possibile giocare in multiplayer (e questo va bene) con un giocatore computerizzato visto in soggettiva ! E mbè? Beh, io suppongo che chi sta leggendo questo file si appassioni a queste cose, al di là dell'utilità (che sinceramente non c'è). Sto ancora pensando di lasciare attiva l'opzione “/USESLIM” che permette di vedere in soggettiva come agisce lo "Slimer"...

The Millennium group (in ordine αβico)**:**

Andrea Fiorentini (aiuto-grafico)
Domenico Valentini (analista-programmatore)
Massimiliano Teso (grafico, creatore suoni, ideatore & internet public relationer)

Per gli amanti del genere "era meglio quando eravamo in pochi e senza icone...", se non ne avete già abbastanza, potrei scrivere anche un file .TXT molto meno multimediale, ma più serio ed organizzato, in cui descrivere in modo dettagliato alcune parti fondamentali come la gestione degli "oggetti ad oggetti", il processo di ray-casting veloce, il sistema di comunicazione tra gli oggetti, ecc.

Ricordo comunque che i sorgenti completi del Kaos (compresi anche quelli dell'editor dei livelli, ecc.) li possiamo dare solo per la modica (e non scherzo affatto, stavolta) cifra di L.100.000. Così, forse, con la parte che mi spetta, comprerò un vero compilatore a 32 bit ed il prossimo gioco che farò (senza ray-casting e con un vero texure mapping) ve lo darò gratis !

Anzi, adesso vi rispiaccico tutto il listino (ho dei doveri nei confronti del gruppo):

1) Modalità "forse poi mi faranno uno sconto" L. 20.000
(gioco completo, tutti i livelli, i mostri e le capacità MultiPlayer)

2) Modalità "vediamo che può fare" L. 30.000
(<1> + editor dei livelli)

3) Modalità "mi rifaccio un gioco nuovo" L. 50.000
(<2> + file grafici (.LBM) + file sonori (.WAV) + possibilità di cambiare le immagini e i suoni del gioco)

4) Modalità "God Mode" L. 100.000
(Tutti i sorgenti, i files .3DS (3D Studio), immagini, suoni, musiche e la documentazione per un gioco Kaos-like e +)

A proposito, nel momento in cui sto scrivendo questo file, la versione shareware ancora non esite!

: -)

Nico