Claudio Bacchetta

Daniele Ingrosso

7 dicembre 2020

Motore grafico

Indice

Sommario

Obiettivo:

L'obiettivo del lavoro è quello di riuscire a realizzare un software di rendering in linguaggio C++ il più possibile simile al Doom engine.

Metodologia:

Essendo il Doom engine scritto in C utilizzando metodi antiquati abbiamo dovuto apportare delle modifiche al codice, utilizzando delle dll esterne e apportando alcune semplificazioni.

Conclusioni:

Come risultato finale del lavoro abbiamo prodotto esattamente quello che ci eravamo preposti ovvero un motore grafico 2.5d pienamente funzionante.

Descrizione Generale

L'obiettivo principale era quello di creare un engine da zero simile a quello di doom che utilizza un tipo di grafica bidimensionale chiamata 2.5d. La particolarità di questo tipo di grafica bidimensionale è che simula una grafica tridimensionale ma risulta più semplice e snella a livello di codice, nonostante questo comunque abbia delle limitazioni come l'impossibilità di guardare lungo l'asse delle y, oppure l'impossibilità di creare strutture a due piani.

La fase di lavoro si è svolta nel seguente modo: per prima cosa abbiamo analizzato le caratteristiche del doom engine e le sue particolarità e schematizzato le funzioni che ci interessavano da implementare (all'inizio risulta inutile pensare già a implementazioni avanzate, risulta ottimale invece concentrarsi sulle parti più semplici del progetto); le funzioni che interessavano a noi erano due: quella di movimento e quella di caricamento esterno del mondo (in modo da poterlo modificare senza modificare il codice). Inoltre come secondo passo ci siamo concentrati sull'ottimizzazione del codice eliminando quelle parti che risultavano non necessarie o potevano essere ottimizzate all'interno di dll.
Per creare questo engine ci siamo affidati ad una dll chiamata PixelToaster che ci ha permesso di avere input da mouse e tastiera oltre a fornirci la possibilità di avere un rendering grafico.

Descrizione Tecnica

Nel programma il giocatore è definito da tre parametri: posizione nella mappa (pos), vettore direzione (dir) e piano di camera (cam), ognuno di questi si divide in x e y.

La posizione nella mappa stabilisce la posizione del giocatore; il vettore direzione rappresenta la direzione in cui sta guardando il giocatore, il modulo del vettore direzione rappresenta la sua distanza dal piano di camera, più il piano di camera è lontano più il campo visivo (fov) sarà stretto mentre più il campo visivo sarà vicino più il campo visivo sarà ampio (il fov per essere ottimale deve avere un valore di 66°); mentre il piano di camera rappresenta la coordinata dove si trova lo schermo (per schermo si intende il punto di visione dell'osservatore).

Il rendering si occupa di calcolare le prospettive, l'altezza dei muri e i blocchi vuoti o pieni.

La mappa viene rappresentata da un file di testo (txt) dove ad ogni numero corrisponde un blocco, quindi se volessi creare una stanza 10x10 creerei un file di testo con scritto nelle prime due righe 15 (perchè questi due numeri rappresentano la grandezza complessiva della mappa in blocchi all'interno della quale sistemerò la stanza) e poi andrei a creare un quadrato 10x10 di numeri, in base al colore che desidero nel programma il numero 1 è codificato con verde quindi se si vuole una stanza 10x10 verde dovrò creare un quadrato di 10 numeri 1 affiancati.

Il movimento del personaggio si divide in camminata avanti e indietro e rotazione sul posto, per la camminata basta aggiornare la posizione corrente ovvero posX e posY, quindi definito un passo "p" lo spostamento si esegue facendo la somma tra vettore posizione e vettore direzione moltiplicando per p, ovvero: posX=posX+dirX*p e posY=posY+dirY*p mentre per il movimento all'indietro la formula è la stessa solo che si sottrae al posto di sommare. Per quanto invece riguarda la rotazione si fa uso di seno e coseno, per girare bisogna far girare il vettore direzione dirX e dirY, avendo un angolo "r" che ci servirà come passo di rotazione in radianti la rotazione avviene moltiplicando i due vettori direzione per coseno e seno sottraendoli ovvero:
dirX=dirX*cos(r)-dirY*sen(r) e dirY=dirX*sen(r)-dirY*cos(r).

Conclusioni

L’obiettivo era quello di creare un motore grafico simil-doom ed è stato un pieno successo, per mancanza di tempo però non siamo riusciti ad implementare tutte le funzionalità che volevamo implementare di base, come: l’aggiunta di una texture per differenti tipi di muro, soffitti e pavimento; la funzione di salvataggio che permette di salvare i parametri di un giocatore per ricominciare da dove aveva lasciato il gioco; e la possibilità di guardare lungo l’asse y (anche se non si può propriamente realizzare essendo un motore grafico bidimensionale, però se muovendo l’asse y si spostasse la linea dell'orizzonte questo darebbe l’illusione di un movimento verticale).

Consigli:

All’inizio abbiamo avuto problemi con l’ambiente di sviluppo per questo consiglio di utilizzare una macchina virtuale con sistema operativo windows 7 e visual studio 2019 con i pacchetti di sviluppo per il c++, consiglio inoltre la lettura di “Borland c++ 3.1 programmazione ad oggetti per dos e windows” che fornisce un pratico aiuto per chi non è molto esperto nel linguaggio c++, inoltre consiglio di approfondire la visione della dll PixelToaster che contiene alcune funzioni già fatte che possono essere implementate facilmente nel codice.