GRIso_game

Luigi Russo, Matteo Salvino

November 18, 2018

Progetto per il corso di Sistemi Operativi - La Sapienza - Roma

1 What

Abbiamo lavorato all'implementazione di un videogioco distribuito multiplayer, partendo dalla proposta di progetto **Videogame** che giá includeva la gestione grafica del videogioco.

2 How

Il videogioco sfrutta l'architettura client-server e due protocolli di comunicazione a livello di trasporto: TCP e UDP.

2.1 Server

Il server é responsabile del corretto funzionamento del gioco: tiene traccia di tutti i clients (utenti) connessi, assegna loro un identificativo globale univoco, aggiorna le coordinate (x, y, theta) di ciascun veicolo e gestisce l'eventuale disconnessione da parte di un client. Il server mantiene tutte le informazioni relative ai clients connessi in una struttura ad hoc WorldServer.

2.1.1 Multithread

Il server accetta connessioni multiple, in particolare é un server multithread in quanto a ciascuna connessione viene associato un thread che si occupa dello scambio di pacchetti con il protocollo TCP.

A ciascun client é assegnato un identificativo globale. Per garantire l'unicitá degli identificativi abbiamo sfruttato un semaforo nella struttura di dati RandomId.

2.1.2 Gli aggiornamenti di stato

Il server riceve continuamente informazioni sulla posizione e sulle forze di ciascun veicolo e ogni 500 ms integra queste informazioni creando un pacchetto con lo stato del "mondo" e lo inoltra agli utenti connessi. Questo scambio di pacchetti avviene utilizzando il protocollo UDP, quindi alcuni pacchetti potrebbero non essere consegnati.

Per evitare conflitti e race conditions abbiamo aggiunto un mutex a questa

struttura: quando il server vuole creare il pacchetto di stato deve prima acquisire il lock sulla struttura. Eventuali aggiornamenti ricevuti in quegli istanti verranno applicati successivamente.

2.1.3 Client offline

La disconnessione di un client viene notificata a tutti gli altri utenti tramite apposito pacchetto, sfruttando il protocollo TCP, che a differenza di UDP garantisce maggiore affidabilitá.

2.2 Client

Il client si connette a un indirizzo e una porta noti, richiede un identificativo al server, manda la texture del proprio veicolo (visibile a tutti gli altri giocatori) e riceve dal server le immagini della mappa e di tutti i veicoli connessi.

2.2.1 Aggiornamenti

Ogni 300 ms il client manda un pacchetto UDP contenente la sua posizione (coordinate x, y e theta). Un thread ad hoc attende il pacchetto di aggiornamento di stato da parte del server e ne applica il contenuto al "mondo" locale del client.

2.2.2 Disconnessione

Quando un utente preme il tasto ESC viene inviato un pacchetto di disconnessione al server e a quel punto il gioco termina, avendo cura di liberare tutte le risorse allocate e terminare i threads in esecuzione.

3 How to run

Prima di tutto bisogna compilare i file, tramite comando **make**.

Per lanciare il server é sufficiente eseguire da shell lo script **server.sh**.
Per il client si puó eseguire lo script **client.sh**.

Per interrompere il gioco é sufficiente premere il tasto **ESC**.