Relazione Progetto - Sistemi Operativi - 2017/18

Nicola Iommazzo 1693395
Francesco Mastrostefano 1672787

Professore: G. Grisetti
Tutor: I. Aloise

Che cos'è

Il progetto sviluppato riguarda la realizzazione di un videogioco distribuito con paradigma client-server. Il videogioco consiste nel guidare un veicolo all'interno di una mappa. Ogni client, connettendosi al server, entra a far parte del gioco e può quindi visualizzare sulla mappa il proprio veicolo e quello dei giocatori connessi.

Come funziona

Strutture dati

Le strutture dati all'interno del videogioco sono:

Vehicle: I veicoli sono identificati tramite un id, assegnato loro dal server al momento della connessione al server, e una texture scelta dall'utente. Ogni veicolo possiede le proprie forze di movimento, che variano in base ai comandi dati dall'utente.

World: Il mondo tiene conto di tutti i veicoli che ne fanno parte, tramite una lista collegata. Periodicamente il mondo viene aggiornato, in modo da tramutare gli aggiornamenti delle forze in spostamenti sulla mappa.

ClientVehicle: Struttura di appoggio per il server, grazie al quale tiene traccia di tutte le informazioni necessarie per comunicare con il singolo client. In particolare qui risiedono i descrittori per i socket TCP, UDP e l'id del veicolo associato. Il server fa uso di una lista di ClientVehicle.

Connessione

Nell'applicazione vengono utilizzati sia socket TCP che UDP. I client inizializzano una connessione TCP verso il server per ricevere un ID, inviare la propria *vehicle texture*, e fare richiesta delle *elevation texture* e *surface texture*. La chiusura del socket TCP viene posticipata fino alla disconnessione del client dal server, per agevolare il trasferimento delle texture di tutti i veicoli in ogni momento del gioco. Viene invece utilizzata una connessione UDP per lo scambio dei pacchetti di gioco: i client comunicano le forze del proprio veicolo al server; il server comunica le posizioni aggiornate di tutti i veicoli appartenenti al mondo a tutti i client attualmente connessi.

Server

Il server gestisce il mondo di gioco, con all'interno i veicoli, e la lista di client attualmente connessi. All'avvio del server viene inizializzato il mondo di gioco e il socket UDP responsabile della comunicazione con i client. Successivamente vengono avviati due thread (*listener_thread* e *sender_thread*), mentre il *main_thread* viene dedicato alla gestione delle connessioni TCP entranti.

- **main_thread:** Il main_thread rimane sempre attivo fino alla pressione dei tasti CTRL+C. Esso si occupa di accettare i nuovi client che si connettono e prepararli al gioco. Al momento della connessione di un nuovo client viene creato un nuovo thread e lasciato eseguire in backgroung in cui si effettua l'assegnazione dell'id univoco al client, la trasmissione della mappa di gioco, e la ricezione della texture del veicolo del client, mentre il main_thread si rimette in ascolto di un nuovo client in ingresso. Terminata la procedura di setup il client viene aggiunto alla lista dei client attualmente connnessi ed è pronto a scambiare i pacchetti di gioco.

- **listener_thread:** Questo thread si occupa esclusivamente della ricezione dei pacchetti sul socket UDP. Alla ricezione di tale pacchetto il suo compito differisce in base al tipo di quest'ultimo (*Disconnection, GetTexture, VehicleUpdate*).

  Disconnection: Il compito del listener_thread è quello di rimuovere il client mittente di tale pacchetto dalla lista dei client attualmente collegati, rimuovere il veicolo associato dal mondo di gioco ed in seguito notificare la sua disconnessione a tutti gli altri client.

  GetTexture: Se il listener_thread riceve un pacchetto di tipo GetTexture allora un client sta richiedendo al texture di un veicolo che ancora non possiede. Semplicemente viene cercata la texture richiesta e spedita al client tramite un pacchetto PostTexture utilizzando il socket TCP.

  VehicleUpdate: Nel caso di ricezione di un pacchetto di tipo VehicleUpdate, il listener_thread non fa altro che estrapolare le forze del veicolo dal pacchetto e inserirle nel proprio mondo. Soltanto successivamente viene effettuato un world update, in modo da tramutare tali forze in cambiamenti di posizione.

- **sender_thread:** Lo scopo di tale thread è quello di inviare costantemente gli aggiornamenti del mondo a tutti i client connessi. Di fatto viene costruito un *WorldUpdatePacket* contenenti tutte le posizioni dei veicoli nel mondo, e inviati a tutti i client connessi tramite la connessione UDP. L'intervallo di tempo tra l'invio di un WorldUpdatePacket ed il successivo è impostato a 30 ms.

Client

Il client gestisce la visualizzazione della mappa di gioco e si impegna a reindirizzare i comandi dell'utente al server sotto forma di forze del veicolo. All'avvio del client viene subito inizializzato il socket TCP per permettere la fase di setup con il server, quindi ottenendo un id unico nel gioco, comunicando la texture del suo veicolo, ed ottenere la mappa dal server. Solo al termine di questa fase vengono creati due thread (*listener_thread* e *sender_thread*) mentre il *main_thread* avvia la finestra di gioco mostrando a schermo la mappa con i veicoli.

- **listener_thread:** Tale thread si occupa della ricezione dei pacchetti di gioco sul socket UDP, differenziando il suo comportamento in base al tipo di pacchetto ricevuto (*Disconnection, WorldUpdate*).

  Disconnection: Se il client riceve un pacchetto di tipo Disconnection, allora è stata notificata la disconnessione di un client dal mondo. Di fatto avviene la rimozione di tale veicolo dal mondo.

  WorldUpdate: Nel caso di ricezione di un pacchetto di tipo WorldUpdate, allora il client estrapola tutte le informazioni relative alle posizioni dei veicoli nel mondo. Se il veicolo è gia presente nella copia del mondo all'interno del client, semplicemente viene aggiornata la posizione del veicolo all'interno della corrispondente struttura Vehicle. Se al contrario il veicolo in questione non è presente all'interno della copia del mondo del client, si provvedde all'inizializzazione di tale veicolo, facendo richiesta della texture al server. Infine si aggiorna il mondo per permettere la corretta visualizzazione dei veicoli nel mondo.

- **sender_thread:** Lo scopo di tale thread è quello di inviare costantemente i comandi inseriti dall'utente, tramutati in forze, al server di gioco. Viene infatti costruito un *VehicleUpdatePacket* contenente le forze del veicolo in questione. L'intervallo di tempo tra l'invio di un VehicleUpdatePacket ed il successivo è impostato a 30 ms.

Il client inoltre implementa un gestore dell'interruzione di HangUp. La chiusura della finestra di gioco tramite il tasto 'ESC' comporta l'invio di un segnale SIGUP a se stesso in modo da attivare la routine di servizio. Tale routine prevede l'invio del pacchetto di Disconnection al server, la chiusura del socket TCP e la distruzione del mondo di gioco.

How To Run

Cloning Repository

È possibile ottenere una copia del software sul proprio pc semplicemente clonando il repository tramite: git clone https://github.com/NicolaIomm/progetto_so_game_1718

Compiling server

Per compilare il server digitare:

$ make so_game_server

Compiling client

Per compilare il client digitare:

$ make so_game_client

Running server

Per avviare il server digitare:

$ ./so_game_server *<elevation_texture>* *<surface_texture>*

Running client

Per avviare il client digitare:

$ ./so_game_client *<ip>*:*<port>* *<vehicle_texture>*

Nota Bene: Per fare in modo che la disconnessione del client avvenga in maniera corretta, è necessario chiudere il gioco premendo il tasto ESC dalla finestra di gioco.