Server Client Videogame

Pierfrancesco Cifra, Giuseppe Costantino

May 15, 2018

Realizzazione di un videogame server client per il progetto del corso di Sistemi Operativi.

Server

Il server all'avvio carica l'elevation e la surface texture, dopodichè imposta il mondo, i vari socket TCP e UDP e la Player list per memorizzare i dati sui giocatori connessi. Vengono quindi lanciati tre thread:

- **Update sender:** Ogni 30ms invia dei World Update contenente le posizioni dei vari giocatori sul socket UDP, in modo che ogni client riceva gli aggiornamenti.
- **Update reciver:** Legge in continuazione dal socket UDP dei pacchetti di tipo Vehicle Update, ed aggiorna le forze del giocatore che le ha mandate, ed esegue un World Update.
- **Player delete:** Ogni 2 secondi facendo uso di un timestamp controlla che ogni giocatore abbia mandato almeno un pacchetto negli ultimi 1200 ricevuti. Se così non fosse viene rimosso dalla partita corrente.

Successivamente si cicla in attesa che un client si connetta al server. Viene quindi calcolato un ID da assegnare al giocatore, e viene dedicato un socket TCP a quel client. Viene quindi lanciato un thread **Player Handler** e si torna in ascolto di nuovi giocatori. Il thread appena lanciato assegna un ID al client tramite un ID Packet, dopodiche riceve la texture del giocatore ed aggiunge il veicolo al mondo ed invia elevation e surface texture al client. Da questo momento in poi il thread è dedicato all'invio delle texture dei nuovi giocatori a tutti quelli connessi precedentemente.

Client

Il client all'avvio carica la propria texture, dopodichè imposta i socket per la comunicazione TCP e UDP. Anche il client fa uso di una Player list per memorizzare i giocatori presenti. Il primo pacchetto che manda è un IdPacket per notificare al server la sua intenzione di connettersi, e aspetta quindi l'assegnazione dell'ID, che viene gestita dal server. Immediatamente dopo invia al server la sua texture, per poterla mostrare agli altri giocatori, e facendo uso della funzione recv packet TCP, riceve surface ed elevation texture. Il gioco può quindi iniziare, e vengono lanciati tre thread:

- **Reciver:** Legge in continuazione dal socket UDP dei pacchetti di tipo WorldUpdate. La prima cosa che fa è un controllo sul numero di giocatori, per vedere se sono diminuiti ed eventualmente eliminarli, successivamente aggiorna le posizioni di ogni veicolo ed esegue un World Update.
- **New player listener:** Legge in continuazione dal socket TCP tramite la funzione recv packet TCP. Ogni pacchetto ricevuto corrisponde ad un nuovo giocatore connesso, viene quindi ricevuta la texture ed aggiunto il veicolo al mondo.
- **Update sender:** Invia ogni 30ms le proprie forze al server tramite dei VehicleUpdatePacket

Una volta che i thread sono stati creati, viene creato anche il mondo ed il veicolo del giocatore. Vengono usati dei semafori per sincronizzare la partenza dei thread con la creazione del mondo.

Altro

- **Network function:** Vi è presente la funzione recv packet TCP usata per ricevere pacchetti. Poichè non è nota a priori la dimensine del paccketto, vengono prima letti 8 byte, che sarebbe la dimansione di un Packet Header. Viene poi letto un numero di byte pari al numero memorizzato nella size.
- **Player list:** Vi è implementata una lista collegata che serve a mantenere informazioni sui giocatori presenti nel mondo.
- **Common:** Contiene alcuni error helper e alcune costanti comuni utilizzate dal server e dal client

How to run

Il tutto deve prima essere compilato tramite il makefile. Per eseguire:

- **Server:** *so_game_server <path to elevation texture> <path to surface texture>*
- **Client:** *so_game_client <server IP addres> <path to texture>*