Relazione progetto Programmazione di reti
Traccia 3- CHATGAME

Francesco Padovani
Matricola n. 0000921464

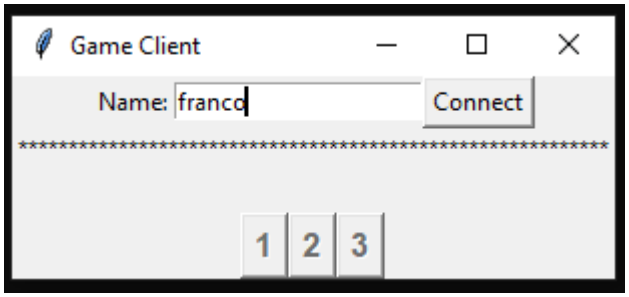
1. Introduzione e descrizione del gioco

Il progetto ha lo scopo di realizzare un gioco multiplayer testuale in cui i due giocatori dovranno rispondere a dei quesiti per accumulare punti partendo dal codice visto a lezione. In una partita vi è un limite temporale allo scadere del quale sarà computato il punteggio finale e decretato un vincitore (o un pareggio). I giocatori sono liberi di rispondere a quante più domande riescono prima dello scadere del tempo, ed ogni risposta corretta incrementerà il loro punteggio di 1 mentre ogni risposta errata equivale ad un decremento di 1 (il punteggio di un giocatore può essere negativo). Per far comparire una domanda il giocatore dovrà selezionare uno dei bottoni presenti nella finestra di gioco e numerati da 1 a 3. Ad ogni scelta due dei bottoni faranno comparire una domanda mentre il terzo contiene la trappola. La scelta della trappola porta al termine della partita con risultante sconfitta del giocatore che l'ha selezionata e vittoria dell'altro. Se invece la scelta del bottone corrisponde ad una domanda essa apparirà a schermo insieme alle 3 possibili risposte. Per poter scegliere una risposta basterà premere il bottone con la numerazione corrispondente. Dopo aver risposto ad una domanda i bottoni torneranno ad avere la funzione di "scoperta" della prossima domanda o della trappola.

2. Descrizione

Il gioco è stato implementato utilizzando un'architettura client-server che si avvale del protocollo TCP. A questo scopo sono stati realizzati due programmi in Python:

- chatgame_server.py come implementazione lato server
- chatgame_client.py come implementazione del lato client



All'avvio di chatgame_client.py apparirà questa schermata dove il giocatore dovrà inserire il proprio username e connettersi al server tramite il bottone connect. Quando due client saranno connessi il gioco partirà.

All'avvio del gioco il timer di entrambi i client partirà (in questo caso impostato a 30 secondi) ed i giocatori potranno rispondere alle domande che vengono loro presentate. Ogni risposta selezionata aggiornerà il punteggio del giocatore sul proprio client e verrà inviata al server che a sua volta la invierà all'avversario per permettere di mantenere consistenti in tempo reale i punteggi di entrambi i giocatori su entrambi i client.

Allo scadere del tempo apparirà a schermo il risultato finale.

Se alla scelta della domanda un giocatore seleziona la trappola apparirà il messaggio “You chose the trap!” ed il client mostrerà un messaggio di sconfitta. Inoltre, verrà mandato un messaggio al server che informerà l’altro client della fine della partita (e della sua vittoria a prescindere dal punteggio).

In questo progetto il server svolge fondamentalmente il compito di mettere in comunicazione i due client. Ha infatti il compito di accogliere i client e di “passare” ad un giocatore le informazioni riguardanti le scelte dell’avversario.

3. Dettagli implementativi e Thread

Il server viene abilitato a ricevere client tramite il metodo start_server() invocato alla pressione del bottone start dell’interfaccia grafica del server. Start_server() avvia un thread che svolge la funzione accept_clients() finchè non sono connessi due utenti. Quando un utente viene ricevuto dal server esso avvia un thread con la funzione send_receive_client_message() che invia dei messagi preliminari al client. Quando entrambi i client sono connessi il server si mette in ascolto dentro ad un ciclo while true. Quando il server riceve un messaggio da un client ne identifica l’indice tramite la funzione get_client_index() ed invia il messaggio al secondo client. Il server avrà dunque in esecuzione tre thread principali durante il gioco: il thread per la propria gui, ed un thread per client per l’invio/ricezione di messaggi.

Lato client il programma permette di connettersi tramite il metodo connect() che permette l’inserimento del proprio username. Dopo l’inserimento del nome connect() invoca il metodo connect_to_server() che crea il socket TCP, si connette se possibile al server e in caso di successo fa partire un thread con il metodo receive_message_from_server() che mette il client in ascolto dentro ad un ciclo while true. Quando il client riceve dal server conferma che due giocatori sono connessi, esso aggiorna la gui ed avvia un thread con il metodo count_down() che fa partire il timer di gioco. I thread attivi lato client durante la partita sono quindi tre: un thread per la gui, un thread per il timer ed un thread per ricevere messaggi dal server. La meccanica principale di gioco è implementata nel metodo choice() che viene chiamato alla pressione di uno dei tre pulsanti sullo schermo. Il client può trovarsi in due diverse “modalità”: scelta o domanda. Quando choice() viene chiamata in modalità scelta viene controllato se è stata selezionata la trappola altrimenti viene

mostrata una domanda e la modalità viene impostata in “domanda”. Quando choice() viene invocata in modalità domanda invece si controlla se la risposta sia corretta o sbagliata, il client aggiorna il proprio punteggio e manda un messaggio al server contenente se la scelta sia giusta o sbagliata. Dopo aver fatto ciò la modalità viene reimpostata a “scelta”. Le domanda mostrata viene selezionata casualmente tramite il metodo getquestions(). Le domande sono rappresentate come quintuple di stringhe contenti in ordine domanda, le tre risposte possibili e l’indice della risposta corretta.

Librerie utilizzate

Sono state utilizzate le seguenti librerie:

- tkinter con cui è stata realizzata la gui sfruttando quella vista a lezione per il rock paper scissors
- socket
- threading
- time
- random