Progetto di "Programmazione di Reti" Chat Game

Davide Merli, Ryan Perrina

14 luglio 2021

Indice

Capitolo 1

Descrizione generale

Il software, realizzato come elaborato del corso di "Programmazione di Reti" del corso di laurea in Ingegneria e Scienze Informatiche dell'università di Bologna, anno accademico 2020-2021, si pone come obiettivo la realizzazione di una implementazione della traccia n. 3 proposta dal professore Andrea Pirrodi:

"Sfruttando il principio della CHAT vista a lezione implementate un'architettura client-server per il supporto di un Multiplayer Playing Game testuale. I giocatori che accedono alla stanza sono accolti dal Master (server) che assegna loro un ruolo e propone loro un menu con tre opzioni, due delle quali celano una domanda mentre la terza è l'opzione trabocchetto. Se sceglie l'opzione trabocchetto si viene eliminati dal gioco e quindi si esce dalla chat. Se si seleziona invece una delle domande e si risponde correttamente al quesito si acquisisce un punto, in caso contrario si perde un punto. Il gioco ha una durata temporale finita; il giocatore che al termine del tempo ha acquisito più punti è il vincitore."

1.1 Istruzioni di gioco

Innanzitutto, il server deve essere eseguito, in modo da aprire le connessioni ai client. Quando un client vuole partecipare al gioco, gli viene mostrata una schermata contenente un'area per lo scambio di messaggi e tre pulsanti: "Invia", "Pronto" ed "Esci". La prima cosa richiesta al client è di inviare un messaggio contenente il suo nome: se il nome è già presente verrà chiesto di reinserire un nome diverso, altrimenti il client viene ammesso e potrà partecipare alla partita. Successivamente, al client viene assegnato uno di sei ruoli disponibili, i quali hanno effetti sul punteggio ottenuto dalle domande. Quando un giocatore è entrato, può cliccare sul pulsante "Pronto"; quando

tutti i giocatori presenti hanno dichiarato di essere pronti il gioco comincia, e non potranno entrare in gioco altri client.

All'inizio del gioco appare una seconda area di testo in cui verranno visualizzate le domande, seguita dal pulsante "Rispondi" e da tre pulsanti etichettati con "A", "B" e "C". Per ricevere una domanda bisogna cliccare su uno di questi tre pulsanti, ma uno dei tre contiene un tranello: premendo tale pulsante, la finestra di gioco verrà chiusa, quindi il giocatore avrà perso la partita. Per rispondere alla domanda bisogna scrivere la risposta in un apposito spazio e cliccare sul pulsante "Rispondi": a quel punto il giocatore verrà informato se ha risposto in modo corretto o errato, e verrà visualizzato il suo punteggio complessivo. All'inzio del gioco, inoltre, viene avviato un timer di durata 2 minuti: allo scadere del tempo il giocatore con più punti è il vincitore.

1.2 Domande

Ogni domanda rientra sotto una delle sei materie scelte (scienze, geografia, storia, sport, spettacolo, arte): tali materie sono ispirate a quelle del gioco da tavolo "Trivial Pursuit", prodotto dalla Hasbro e dalla Horn Abbot. Le domande stesse, ad onor del vero, sono state selezionate dall'edizione del 2016 di Trivial Pursuit.

1.3 Ruoli dei giocatori

Normalmente, quando un giocatore risponde ad una domanda, gli viene aggiunto un punto se la risposta è corretta, o gliene viene sottratto uno se la risposta è sbagliata. Lo scopo dei ruoli è di rendere questa meccanica più "interessante", facendo sì che alcune domande abbiano un valore diverso a seconda del giocatore che vi risponde, e del ruolo assegnatogli. In seguito una veloce spiegazione dei ruoli:

- Scienziato: guadagna due punti se risponde correttamente ad una domanda di scienze, ma perde due punti se sbaglia una domanda di scienze.
- Geografo: guadagna due punti se risponde correttamente ad una domanda di geografia, ma perde due punti se sbaglia una domanda di geografia.
- Storico: guadagna due punti se risponde correttamente ad una domanda di storia, ma perde due punti se sbaglia una domanda di storia.

- Atleta: guadagna due punti se risponde correttamente ad una domanda di sport, ma perde due punti se sbaglia una domanda di sport.
- Attore: guadagna due punti se risponde correttamente ad una domanda di spettacolo, ma perde due punti se sbaglia una domanda di spettacolo.
- Artista: guadagna due punti se risponde correttamente ad una domanda di arte, ma perde due punti se sbaglia una domanda di arte.

Capitolo 2

Strutture dati utilizzate

Segue successivamente un breve descrizione delle strutture dati utilizzate. Si fa presente che non sono stati sfruttati set e tuple poiché non ritenuti necessari alle operazioni da svolgere all'interno di questo progetto.

2.1 Dizionari

Innanzitutto sono stati utilizzati i dizionari, un tipo built-in, mutabile e non ordinato che contiene elementi formati da una chiave (key) e un valore (value). Una volta che il dizionario è creato viene popolato con un insieme di coppie (chiave:valore), e si può usare la chiave (che deve essere univoca) per ottenere il valore corrispondente. E' proprio per queste caratteristiche che si è scelto di utilizzare dei dizionari per implementare la memorizzazione degli utenti che accedono al server e associare il ruolo alla materia corrispondente.

2.2 Liste

Una lista è una collezione mutabile di elementi indicizzata. Abbiamo usato questa collezione per memorizzare le domande del gioco, per mantenere l'insieme dei giocatori, ed una per tenere traccia dei possibili ruoli da assegnare ai giocatori stessi. La lista contenente le domande viene popolata utilizzando un metodo della libreria json per leggere dal file "questions.json" le domande stesse, la seconda lista serve per tenere traccia delle informazioni di ogni giocatore (nome, punteggio, ruolo), mentre la terza è utile al momento dell'assegnamento dei ruoli: quando un nuovo player entra nel gioco viene estratto un elemento casuale dalla questa lista.

Capitolo 3

Thread attivi

3.1 Server

Nel server vengono gestiti due thread: il primo si occupa di accettare le connessioni in entrata, che grazie al ciclo while rimangono in attesa occupandosi di immagazzinare tutti i client arrivati; il secondo, invece, ha il compito di gestire i client e tutta la logica di gioco. Il suo unico parametro è il client che deve gestire in modo tale da poter ottenere i messaggi che invia. A seconda del messaggio che l'utente invierà verrà eseguita una diversa azione. Per far sì che il server non si chiuda subito dopo essere partito, ovvero dopo l'accettazione delle connessioni, si utilizza il metodo join che aspetta il completamento della funzione senza passare alla riga successiva. Facendo diversamente causeremmo la chiusura del server.

3.2 Client

Per quanto riguarda il client viene utilizzato un unico thread che si occupa di gestire la ricezione dei messaggi andando, poi ad aggiungerli alla vista che si occupa di mostrarli, realizzata tramite la libreria tkinter.

3.2.1 Timer

All'inizio del gioco ogni client inizializza un timer, realizzato come un thread separato: ogni secondo questo thread modifica una label (componente della libreria grafica tkinter) apposita che tiene traccia del restante tempo di gioco. Alla fine del conto alla rovescia, il thread termina e la label da esso modificata, così come una parte della GUI destinata alla visualizzazione di domande e risposte, scompaiono dalla schermata di gioco.

Capitolo 4

Struttura del progetto

E' stato scelto di strutturare in parte il progetto "ad oggetti" per integrare le conoscenze acquisite nel corso "Programmazione ad Oggetti" e sfruttare al massimo le potenzialità del linguaggio.

4.1 RoadMap

- La classe Timer implementa il concetto di un timer che restituisce, attraverso il metodo "convert", un tempo in formato minuti:secondi.
- La classe Player non presenta metodi in sé per sé ma solo dei campi per memorizzare le informazioni del giocatore (nome, ruolo e punteggio).
- La classe Question, similmente alla classe Player, non presenta metodi ma solo dei campi per memorizzare le informazioni relative alla domanda (domanda, materia e risposta).

Capitolo 5

Istruzioni per l'esecuzione

5.1 Eseguire il server

Spostarsi all'interno della cartella "src" ed eseguire il comando python3 server.py. Il server si metterà in ascolto sull'indirizzo IP "127.0.0.1" e sulla porta "53000". Per modificare questi valori occorre modificare direttamente il file.

5.2 Eseguire un client

Spostarsi all'interno della cartella src ed eseguire il comando python3 client.py Il client proverà automaticamente a connettersi all'indirizzo IP "127.0.0.1" e alla porta "53000". Per modificare questi valori occorre modificare direttamente il file.