Chat Game

davi.merli, ryan perrina

July 2021

Indice

Capitolo 1

Analisi

1.1 Requisisti

Questo progetto si riferisce alla terza traccia data dal professore Andrea Pirrodi, si tratta di un gioco a cui due o piu giocatori possono partecipare rispondendo a delle domande di vario genere. Durante il gioco è possibile scambiare messaggi tra giocatori, però solamente nei momenti in cui non si è sottoposti a interrogazione. Prima di poter ricevere una domanda viene richiesto al giocatore di scegliere uno dei tre bottoni (A,B,C) che compariranno, uno dei tre è il trabochetto che ti fara uscire dalla partita. Quando una risposta è considerata giusta il giocatore acquisisce uno o piu punti a seconda della tipologia della domanda. Allo scadere del tempo vince chi ha accumulato piu punti.

1.2 Requisisti funzionali

- All'inizio della partita ogni giocatre sceglie un nome univoco, e successivamente gli viene assegnato un ruolo .
- Per potere iniziare il gioco c'è bisogno di due o piu giocatori.
- Per iniziare il gioco tutti i giocatori dovranno cliccare sul pulsante "Pronto", e solo quando tutti lo avranno fatto si inizierà il gioco.
- Una volta iniziato il gioco nessun altro giocatore potrà unirsi alla partita corrente.

Capitolo 2

Strutture dati utilizzate

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, si può usare la chiave (che deve essere univoca) per ottenere il valore corrispondente. E' proprio per queste caratteristiche che sono stati utilizzati per implementare la memorizzazione degli utenti che accedono al server e associare il ruolo alla materia corrispondente.

2.2 Liste

Una lista è una collezzione mutabile di elementi indicizzata. Abbiamo usato questa collezzione per memorizzare le domande del gioco, per mantere l'insieme dei giocatori attualmente in gioco, ed una per tenere in memoria tutti i ruoli. Nella prima lista che tiene traccia delle domande, si utilizza un metodo della libreria json per leggere da file le domande ed inserirle in essa, la seconda lista servere per tenere traccia delle informazioni di ogni giocatore(nome,punteggio,ruolo), è utile all'aseegnamento dei ruoli: quando un nuovo player entra nel gioco viene estratto un elemento casuale dalla questa lista.

Non sono stati usati set e tuple perchè non ritenuti necessari alle operazioni all'interno di questo codice.

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 invierà l'utente verrà eseguita una diversa azione. Per far si che il server non si chiuda subito dopo essere partito, ovvero dopo l'accettazione delle connessioni, utilizziamo il metodo join che aspetta per 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 separat: 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 il progetto "a 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 implenta il concetto di un timer che resistuisce 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 (punti, ruolo, e nome).
- La classe Question non presenta metodi in sé per sé ma solo dei campi per memorizzare le informazioni della domanda (domanda, risposta e materia).