Peer-Review 2: Network

Stefania Raibaldi, Veronica Viceconti, Yana Siao, Ishmeet Singh
Gruppo GC09

Valutazione dell'implementazione della network del gruppo GC56.

Lati positivi

L'idea di separare Controller in due componenti è molto buona in quanto questo semplifica la leggibilità e la manutenzione del codice.

È molto interessante come avete pensato di implementare la AF riguardo la connessione e disconnessione del client. In particolare il metodo saveGame per salvare la "view" del giocatore e il metodo join a cui viene passato l'id della partita in modo da poter essere riconnesso.

Lati negativi

Alcuni aspetti su cui vi invitiamo a ragionare:

1. Per quanto riguarda l'attributo serverID presente in client vi suggeriamo di inserirlo (nel caso si abbia bisogno di un ID per il server) all'interno della classe server, in quanto il server sarà solo uno per ogni partita di conseguenza il client "avrà" un unico server.
2. Sempre in client suggeriamo di eliminare l'attributo gameController in quanto il client non deve avere la possibilità di chiamare i metodi del controller, ma deve fare riferimento sempre al server.
3. Per quanto riguarda l'implementazione del protocollo socket potrebbe essere utile implementare un'enumerazione con i vari messaggi scambiati tra client e server.

4. Suggeriamo di porre attenzione al metodo connectToGame in quanto da specifiche un giocatore non può connettersi a una partita a sua scelta ma partecipare (join) a una partita disponibile
5. Il metodo nel client receiveUpdate potrebbe essere un pò generico, suggeriamo di creare una classe o un'interfaccia che distingue i vari tipi di aggiornamenti es. Punti, carte, eventualmente gli aggiornamenti della chat.
6. In VirtualServer potreste cercare di accorpare i metodi di pesca carta dal tavolo; inoltre potreste pensare di estrarre gli obiettivi comuni in modo casuale quando la partita inizia, senza che questa risulti un'azione effettuata da un client.
7. Suggeriamo di collegare il listener al model e alla view invece che al controller per evitare di dover gestire tutti i possibili errori, essendo che in caso di errore nel model il controller propaga l'errore anche al client.

Confronto tra le architetture

Le architetture risultano essere molto simili. Entrambi abbiamo pensato di creare un unico server che implementa RMI o socket in base alla scelta dell'utente. Inoltre per ogni partita viene allocato un Controller che andrà a gestire la singola partita evitando colli di bottiglia con molteplici richieste. Per la nostra architettura come suggerito dal professore durante un laboratorio abbiamo pensato di implementare RMI e socket entrambi asincroni così facendo le richieste al server da parte del client dovrebbero essere gestite in modo più veloce senza tempi di attesa.