Peer-Review 2: CONTROLLER, VIEW, RETE

<Rubin Ronchieri Byrd>, <Gabriele Proverbio>, <Alessio Tagab>, <Andrea Pazienza>

Gruppo <38>

Valutazione del diagramma UML delle classi del controller e della view, del sequence diagram di rete del gruppo <28>.

Lati positivi

Partendo dalla view, buona la scelta di una enum a rappresentare lo stato della partita per distinguere le varie fasi di gioco, scelta che anche il nostro gruppo ha adottato nell'implementazione. Interessante anche il metodo per visualizzare ad ogni turno gli obiettivi raggiunti dal player nel turno stesso.
Passando al controller, si può notare come tutte le fasi possibili di gioco siano coperte, compreso anche l'endgame, come si nota dal metodo getRanking() per la visualizzazione del punteggio.
Nel complesso, sicuramente un'implementazione strutturate, che copre tutte le situazioni possibili e da cui emerge una visione ben chiara di come si svolgerà la partita nel complesso.
Passando alla rete, viene proposto il sequence diagram che rappresenta le azioni del giocatore nel turno, ben spiegato all'interno della relazione mandata insieme al diagramma; ottima copertura degli errori, come si nota dalle eccezioni che vengono tirate in caso di errore. Anche qui, emerge la chiarezza di fondo e la comprensione di tutte le possibili fasi di gioco.
Buona l'idea di tenere traccia in ogni turno del momentaneo punteggio, tramite il rankMessage.
Per quanto riguarda la serializzazione degli oggetti, il nostro gruppo ha avuto una visione simile, con due tipologie possibili di oggetto, una turnview che viaggia dal server ai client, e una choice che viaggia dai player al server e da li al modello.

Lati negativi

Manca una relazione che accompagni UML del controller e della view, che permetta una comprensione più immediata del diagramma. Manca un UML riassuntivo della parte di rete, che mostri le interfacce di client e server e i metodi di entrambi.
Non avendo questo UML, non è possibile avere un'idea sul funzionamento complessivo della rete. Dal sequence diagram, però, sembra che il server comunichi direttamente con i player e quindi con la view, senza un passaggio intermedio con il client (ipotesi sulla base del diagramma, non sappiamo se vero), e a nostro parere è necessario che venga chiarito meglio questo.
L'utilizzo del pattern observer/observable è sicuramente una buona scelta, ma non la migliore per le notifiche degli eventi: suggeriamo di usare dei listener, che faciliteranno ancora di più il lavoro nel momento in cui entrerà in gioco la GUI.

Confronto tra le architetture

Apprezzabile sicuramente la coerenza di fondo del progetto, bene la schematizzazione delle fasi di gioco. Implementazione che dal UML sembra essere molto "user friendly", e permetterà sicuramente una buona esperienza di gioco all'utente (vedi il mostrare a schermo turno per turno la classifica momentanea e gli obiettivi raggiunti in quel momento).
Probabilmente questo è il campo da cui il nostro gruppo può maggiormente prendere spunto. Sembra essere buona la gestione degli eventi e il passaggio degli oggetti, punto su cui troviamo le maggiori similitudini fra i due gruppi, anche se l'utilizzo dei listener permetterebbe una maggior flessibilità e modularità, per cui questo è il nostro suggerimento principale.