Peer-Review 1: UML

Giulia Rossi, Andrea Rondi, Xin Ye, Samuele Russo

Gruppo GC41

Valutazione del diagramma UML delle classi del gruppo GC31

Lati positivi

I lati positivi che abbiamo riscontrato nell'UML del progetto del gruppo 31 sono:

- La definizione di una classe apposita per la Bag, che consente una gestione efficiente delle tessere in gioco.
- L'attenzione alla flessibilità del codice nella maggior parte delle classi.
- La predisposizione delle classi 'Chat' e 'Message'.
- L'Enum 'TILETYPE' per la definizione delle posizioni giocabili della plancia a seconda del numero di giocatori.

Lati negativi

I lati negativi che abbiamo riscontrato nell'UML del progetto del gruppo 31 sono:

- La separazione di alcuni metodi tra le classi del controller e quelle del model. Consigliamo di lasciare getter e setter nel modello, che si occupa di mantenere lo stato attuale della partita e di spostare i metodi che lo modificano (come il refill() della Dashboard) nel controller.
- Il valore 'BLANK' nell'enum 'COLOR' si distacca dalle specifiche del gioco. Sia logicamente che graficamente esistono infatti solo 6 tipi di Tiles, la cui assenza dalla Dashboard può essere verificata senza la necessità di definire un ulteriore tipo di tessera. Consigliamo inoltre di utilizzare un enum che faccia riferimento ai tipi delle tessere (CAT, BOOK…) e non ai colori, sempre per rispecchiare meglio il gioco fisico.
- Non è chiaro come l'attribuzione dei punteggi per i CommonGoals tenga conto del numero di giocatori e dell'ordine in cui sono stati conseguiti tali obiettivi. Secondo il regolamento, infatti, i punti da assegnare ad un giocatore che abbia completato l'obiettivo dipendono da quante persone sono riuscite a farlo prima di lui. Si potrebbe conservare questa informazione in una struttura dati o alternativamente creare una classe apposita per gli ScoringTokens.
- L'utilizzo di un'ArrayList di matrici come parametro nelle funzioni di 'CONTROLLER_GAME'. Riteniamo sia possibile implementare più efficientemente le stesse funzionalità con delle strutture dati più semplici.
- La dipendenza del calcolo degli adiacenti dalla colonna scelta per l'inserimento delle tessere nella Shelf. Consigliamo di effettuare il check sulle adjacentTiles in un unico metodo e di fare a meno del pattern Command e della relativa suddivisione della logica in tre concreteCommands.
- L'assenza delle cardinalità sulle associazioni, la scelta dei tipi di relazione, l'uso improprio della sintassi e la mancata suddivisione esplicita in packages, che rischiano di portare ad un'erronea interpretazione del vostro progetto.

Confronto tra le architetture

Una delle differenze più evidenti nel confronto tra le architetture si nota sicuramente nella rappresentazione della Plancia Soggiorno: noi l'abbiamo implementata come una matrice di oggetti 'BoardBox', utili per effettuare dei controlli sull'occupabilità, mentre il gruppo 31 l'ha gestita come una matrice di Tiles, sfruttando direttamente l'enum TYPE per tali controlli.
Abbiamo trovato interessante l'utilizzo di una classe Chat che permette di salvare lo stato della stessa anche in seguito ad un'eventuale perdita di connessione.
Abbiamo apprezzato inoltre l'introduzione di una classe Bag per la gestione della Tiles in gioco, che rende la struttura più in linea con il gioco fisico.
Trarremo sicuramente spunto da queste osservazioni per migliorare la nostra architettura.