Peer-Review 1: UML

Francesco Tarantino, Carlo Prestifilippo, Davide Vinci, Alessandro Scibilia

Gruppo 2

Valutazione del diagramma UML delle classi del gruppo 42.

Lati positivi

Il progetto è costituito da molte classi, risulta dunque modulare, garantendo una facile manutenibilità del codice. Inoltre, l'elevata varietà e quantità dei metodi rende evidente il funzionamento e l'evoluzione di ogni componente del gioco, mantenendo esplicita ogni regola nel modello.

Un altro elemento positivo del progetto valutato è il numero di classi che ereditano da *CommonGoalChecker*, che sono minori rispetto al numero delle Common Goal Card presenti in gioco: ciò permette di riutilizzare il codice per tutte le carte che effettuano controlli simili sulla matrice della *Bookshelf*.

Lati negativi

Il diagramma UML risulta a tratti confusionario, soprattutto nella parte centrale: potrebbe essere migliorato evitando di aggiungere frecce uscenti in direzione opposta da una classe.

Inoltre, si osserva che la Board è implementata come un *ArrayList*, e non come una matrice di *Tile*; ciò potrebbe rendere l'implementazione dell'algoritmo di creazione della Board più complessa, ed il relativo ripristino più elaborato. A tal proposito, si nota come la classe *Box* non abbia corrispondenza con alcun elemento specifico del gioco, ma sia stata aggiunta a causa dell'implementazione della Board.

In aggiunta, si nota la scelta di utilizzare due vettori di interi e non una matrice per rappresentare le posizioni utili al controllo del raggiungimento di un obiettivo personale.

Infine, si rileva che *l'ArrayList* players presente nella classe *Game* sembrerebbe essere di tipo pubblico dal diagramma UML, nonostante siano presenti anche i relativi metodi getter e setter.

Confronto tra le architetture

L'implementazione della "mano" del giocatore e della "mossa" possono risultare convenienti per mantenere l'informazione aggiornata su cosa succede alle tessere in gioco. Nella nostra implementazione, invece, la "mossa" non viene registrata in memoria sul server come oggetto, in quanto le tessere scelte dalla board andranno obbligatoriamente inserite nella Bookshelf del giocatore corrente: abbiamo quindi deciso di memorizzare lo stadio temporaneo di "*in mano*" soltanto nella vista.

L'implementazione delle carte di obiettivo comune è sostanzialmente diversa tra le due architetture. La scelta di mantenere nel controller la logica di controllo del raggiungimento degli obiettivi comuni è comprensibile, ma questo potrebbe renderne più complessa la gestione, in quanto in ogni momento del gioco deve essere garantita la corrispondenza tra il *checker* e la relativa

CommonGoalCard presente nel modello. Nello sviluppo della nostra architettura si è scelto di implementare i controlli nel modello, in quanto elementi costitutivi della carta stessa: ogni classe che rappresenta un gruppo di obiettivi comuni simili eredita dalla classe astratta *CommonGoalCard*, implementando il metodo di check, che invece si occupa della distribuzione dei punti, essendo questa un'operazione comune ad ogni carta. In una ipotetica espansione del gioco, sarà quindi più semplice integrare nuove carte di obiettivi.
La suddivisione in due classi distinte del *controller* per cui una implementa il setup del gioco e l'altra si occupa di gestire una partita già in corso può essere utile per suddividere i metodi e le istruzioni relative alle due fasi del gioco, mentre il nostro unico controller dovrà occuparsi di gestirle per intero.
La revisione del progetto assegnato ha permesso, infine, di verificare l'assenza delle tessere punteggio dai *Player* dal progetto del gruppo revisore, presente invece nel progetto revisionato. Si prevede di implementare una soluzione simile per mantenere in memoria ogni singola tessera punteggio presa dal giocatore, anche per facilitarne la visualizzazione di questa da parte della vista.