Peer-Review 1: UML

Gruppo 01

Zanca Federico, Zhuge ZhengHao Michele, Wu JiaHui, Zheng Fabio

Valutazione del diagramma UML delle classi del gruppo 41.

Lati positivi

È stato apprezzabile l'adozione di 4 controller invece che affidarne l'intera gestione ad uno unico, rendendo più chiara la suddivisione delle chiamate ai metodi del model.

È interessante implementazione dei metodi per il calcolo dei gruppi adiacenti all'interno della bookshelf, tramite l'impiego di un algoritmo di depth first search.

L'uso di uno strategy pattern per definire le varie Common Goal Card è un utilizzo da manuale di questo tipo di design pattern, che è pensato proprio per consentire di selezionare a runtime l'algoritmo da impiegare.

L'utilizzo dei metodi relativi a *freeEdges* è una buona scelta per poter valutare velocemente se ogni data tessera sia prendibile o meno.

Lati negativi

L'attributo *firstPlayer*, i metodi relativi e *setNextPlayer* potrebbero essere superflui, l'informazione sull'ordine dei giocatori è possibile esplicitarla all'interno dell'ArrayList di giocatore.

Anche l'attributo *occupied* di *BoardBox* e *BookshelfBox* risulterebbe superfluo e tale funzione potrebbe essere realizzata tramite un controllo su *ItemTile*.

Inoltre, ritengo che i controller potrebbero beneficiare di una riduzione dei metodi al loro interno, spostando alcuni metodi come *createCommonGoalCard*, *CreatePersonalGoalCard*, *CheckPersonalGoal*, ecc.

all'interno del *Model*.

Come nota conclusiva, potrebbe essere presa in considerazione l'idea di compattare i vari metodi e attributi relativi alle coordinate di *BoardBox*, creando delle classi aggiuntive e di includere i *set* e *get* di *PlayerChoice* in un unico metodo di *ItemChoose*, per rendere il tutto più leggibile e più semplice da invocare dall'esterno.

Confronto tra le architetture

Dopo aver analizzato l'architettura del gruppo 41, abbiamo realizzato che alcuni aspetti architetturali del nostro progetto potrebbero essere migliorate come ad esempio un migliore utilizzo di un sistema di *BooshelfBox* e *BoardBox* che potrebbe essere utile anche per una futura implementazione della *View*.

Inoltre, l'attributo *freeEdges*, sempre relativo a *BoardBox* è un'altra soluzione interessante adottata nel loro progetto, che potrebbe semplificare gli algoritmi posti al calcolo delle tessere che possono essere scelte dal giocatore in fase di gioco.

Dall’altro canto, ritengo che una soluzione che includa nell’enumerazione di *Item Type* i tipi *Empty* e *Forbidden*, presenti nella nostra architettura, potrebbero rendere la definizione di concetti come casella proibita o casella vuota/non occupata, più semplici da implementare e snellire la struttura della *Board* e della *Bookshelf* e relative *Box*.