Peer-Review 1: UML

Lorenzo Luisi, Luca Parsani, Davide Simoncini, Lorenzo Zanardi

Gruppo 44

Valutazione del diagramma UML del model del gruppo 34.

# Lati positivi

Avere EndGameCard nel PlayerModel è comoda, altrimenti è necessario appoggiarsi ai punti dei giocatori per capire chi l’ha ottenuta.

L’array di liste di ObjectCard è molto comodo, quando si inserisce una tessera è sufficiente fare un append.

Le tessere degli obiettivi comuni sono gestite in modo molto diverso dal nostro, dove abbiamo semplicemente inserito degli attributi all’interno della classe CommonGoal invece di creare una classe PointCard. È un implementazione molto chiara e diretta.

# Lati negativi

Per come sono configurati i PlayerModel, diventa complesso risalire alla posizione dei giocatori (che può servire in ambiti più specifici come la persistenza), poiché sono necessari più confronti. Una soluzione semplice è aggiungere un banalissimo attributo posizione.

La funzione getPlayerAmount è superflua, basta chiamare la funzione size dell’array di PlayerModel.

Il funzionamento di isGameOver e setGameOver è poco chiaro. Sono due metodi di set, uno generico e uno condizionato, e manca un getter? In tal caso è molto più comodo accorparli in un metodo solo e definire un getter.

Per come e’ definito getCardsFromBoard, diventa macchinoso lo svolgimento del turno, poiché non è possibile riordinare le tessere una volta prelevate.

# Confronto tra le architetture

La differenza principale tra quest’implementazione e la nostra è come sono stati gestiti model e controller, in questo caso il model fa quasi tutto mentre nel nostro caso le funzioni di logica applicativa sono state delegate al controller.

La tessera da un punto nella plancia è gestita in modo molto simile alla nostra implementazione.

La funzione nextPlayer direttamente nel model è sicuramente più comoda della nostra generica funzione setCurrentPlayer, a cui il controller deve passare come attributo il giocatore successivo.

Nell’implementazione i punti vengono calcolati man mano, in contrasto con il nostro calcolare alla fine. Ciò permette di mostrare dinamicamente i punti durante le fasi di gioco, anche se non è richiesto.

A parer nostro la plancia è gestita in modo macchinoso, con i metodi validTile e validMove, al contrario della nostra in cui tra i tipi di tessere ne esiste uno che identifica le caselle non valide.

L’obiettivo personale è gestito in modo diverso, qui è implementato un insieme di carte mentre nella nostra implementazione l’obiettivo personale è una simil-libreria da confrontare con la libreria vera e propria.

Le carte obiettivo comune sono state gestite con un’ereditarietà, mentre nel nostro caso è stata gestita con un pattern Strategy.