Peer-Review 1: UML

Mirko Scigliano, Davide Villani, Francesco Virgulti, Gabriele Scorrano
GC30

Valutazione del diagramma UML delle classi del gruppo GC20

Lati positivi

- Enumerazioni semplici e concise, riescono ad esplicare correttamente la suddivisione di carte e valori presenti nei vari angoli senza introdurre ridondanza.
- Buona coerenza fra Carta e Corner, molto bene l'attributo link che permette di risolvere i collegamenti fra carte in maniera semplice soprattutto in virtù del calcolo dei punteggi sulle carte speciali.
- Distinzione di carte oro in carte con bonus dati dall'angolo o dal numero di obiettivi molto efficiente in vista degli algoritmi per il calcolo dei punteggi, avremmo tuttavia preferito anche una distinzione tra le classi carta - carta risorsa a causa delle differenze tra carte iniziali e risorsa, tuttavia è solo una questione di comodità.
- Molto utili i metodi di getBelow, getAbove presenti in table che semplificano molto il controllo delle carte intorno.
- Molto buono l'utilizzo del Factory Method per quanto riguarda la gestione dei Goal, snellisce la gestione del codice e aumenta la scalabilità del gioco; tuttavia, lo Strategy Pattern è più adatto, in quanto accresce ancor di più la scalabilità e la flessibilità.

Lati negativi

- Organizzazione della carta in linea generale coerente e strutturata, tuttavia avremmo optato più alla creazione di una classe che rappresenta il lato della carta (contenente i vari angoli e risorse centrali), istanziando poi due lati all'interno di una carta e non un array, che potrebbe appesantire la scrittura del codice cercando di dereferenziare il puntatore ad ogni utilizzo.
- Il metodo per il calcolo del punteggio (che da quanto si comprende è relativo al punteggio delle carte gold con obiettivi) è forse preferibile inserirlo all'interno della classe Player o, meglio ancora, nel controller.
- Una piccola imprecisione relativa al consiglio nel primo punto è l'inserimento della posizione all'interno della carta, preferibile forse inserire il riferimento della carta all'interno della posizione nel campo da gioco per alleggerire.
- La gestione dei Deck tramite classi apposite, invece che come attributi della classe MatchGesture, semplificherebbe l'implementazione e la flessibilità del codice; inoltre, sarebbe meglio istanziare gli stessi come pile.

- Nella classe Player, potrebbe essere utile aggiungere un attributo per rappresentare lo stato di gioco del giocatore, al fine di gestire al meglio le dinamiche della partita.

Confronto tra le architetture

Tra i punti che potremmo prendere in considerazione per il nostro miglioramento possono esserci i metodi presenti in table per il get delle carte intorno alla card data.

La gestione delle carte nei due UML presenta sostanziali differenze, poichè nel nostro caso ogni singola tipologia di carta costituisce una espansione della classe astratta PlayCard, mentre nell'altro UML non viene distinta una sottoclasse differente per la StartingCard, nè vengono gestite le ResourceCards come espansione di una classe astratta.
Inoltre, è ottimale l'espansione della GoldCard in due sottoclassi divise a seconda del constraint, nonchè la gestione dei constraint stessi tramite passaggio di un ArrayList di Symbol e non tramite Enumeration; tuttavia, è poco chiaro l'override del metodo pointsCalculator.

Infine, la differenza più evidente riguarda la gestione del GameField, nel nostro caso composto da celle contenenti ognuna un angolo di ogni carta piazzata nel campo di gioco.