Peer-Review 1: UML del Model

Beniamino Perri, Nicolò Ravasio, Paolo Vanotti, Davide Roccuzzo
GC47

24 giugno 2024

Valutazione del diagramma UML delle classi del gruppo GC37.

1 Lati positivi

- Sono presenti classi per implementare alcune funzionalità aggiuntive, come Game e Chat;
- le enumerazioni gameState e playerState sono utili per la gestione della partita;
- si fa un uso intelligente di alcune strutture dati (come symbolCount con HashMap).

2 Lati negativi

Sono stati individuati i seguenti lati negativi:

- L'HashMap coordinate-carta non ha reali vantaggi rispetto ad una matrice (array bidimensionale), a parte il fatto di essere dinamica invece che statica. Implementazione poco intuitiva e non particolarmente efficiente quando bisogna eseguire i controlli per giocare una carta, oppure quando bisogna verificare un obiettivo raggiunto;
- non è ben chiaro cosa sia gestito dal Model e cosa dal Controller, in quanto nel Model mancano vari metodi fondamentali per giocare una partita completa. Probabilmente, gran parte della logica di gioco è lasciata al Controller;

- l'enumerazione di simboli ha più senso dividerla in sottogruppi: negli obiettivi infatti alcuni tipi di simboli non possono essere presenti;
- nella classe Card, utilizzare l'attributo retro e currentSide di tipo CardBack e Card ha poco senso e genera confusione (riferimenti ricorsivi);
- gli obiettivi a "L" e quelli diagonali sono difficili da distinguere tra di loro (orientamento della L per esempio), perché l'unica cosa che li differenzia è l'attributo color. Se le carte del gioco dovessero cambiare, questa ipotesi non sarebbe neanche più valida;
- i metodi nell'UML sono sprovvisti di parametri e ciò rende difficile capirne bene il funzionamento: essi non hanno parametri neanche nel codice? Oppure è una dimenticanza di rappresentazione nell'UML?

3 Confronto tra le architetture

Nel confronto svolto tra il nostro diagramma UML e quello ricevuto, abbiamo notato qualche differenza nelle scelte di design. Senza ombra di dubbio la sezione che più differisce dal nostro modello delle classi è quella relativa alla gestione delle carte, nella fattispecie per quanto riguarda la classe Corner e la gestione di Risorse/Oggetti caratteristici del gioco. Mentre il nostro gruppo ha optato per un utilizzo più "generico" di questi ultimi, rappresentando entrambi con due differenti enumerazioni ad hoc, nell'UML ricevuto la scelta presa è stata quella di accorpare questi due elementi di gioco in una singola classe, assieme ad altri elementi utili per classificare le proprietà di un dato angolo.

Sebbene questa scelta abbia certamente qualche vantaggio per la gestione della classe Corner, le cui caratteristiche sono così subito identificate, riteniamo che nel complesso questa non sia una strategia ottimale: è infatti sufficiente classificare alcune delle informazioni sull'angolo semplicemente utilizzando attributi booleani (e.g. "isCovered" per indicare se sia già coperto da qualche altra carta). Inoltre, questa scelta non è ideale per la classe Achievement: gli obiettivi relativi agli Item sono caratterizzati dall'enumerazione Symbols, che include anche elementi (e.g. NOCORNER, INSECT...) che non devono mai apparire su tale carta.

L'altra differenza sostanziale tra le due architetture è la presenza di una classe Game, contenente gli attributi e i metodi cruciali per il corretto svolgimento della partita. Analizzando tale classe si può intuire che la logica applicativa sembra essere stata delegata al controller, essendo la maggior parte dei metodi presenti metodi getter/setter. Al contrario, la scelta del nostro gruppo è ricaduta su un'architettura più "Model-oriented", ovvero in cui i metodi che permettono il flow di gioco sono presenti nel Model stesso, venendo solamente invocati dal controller. Riteniamo che l'implementazione di una classe Game abbia particolarmente senso se ci sia l'intenzione di aggiungere, come funzionalità aggiuntiva, la possibilità di gestire più partite simultaneamente. Dall'UML è facile intuire che l'altra funzionalità aggiuntiva presa in considerazione sia la Chat. La presenza già nell'UML inziale delle classi relative alle funzionalità aggiuntive è certamente una nota di merito da cui prendere spunto per l'UML del nostro progetto.

Esiste infine una discrepanza tra le due architetture per quanto riguarda la gestione della PlayerBoard, nello specifico per quanto concerne il piazzamento delle carte. Il gruppo in analisi ha infatti scelto di ricorrere a una hashmap <coordinate (x, y) - carta piazzata>. Riteniamo che questa scelta sia inutilmente complessa: usare una rappresentazione matriciale è più che sufficiente per gestire il piazzamento delle carte per ogni giocatore, dal momento che la Board difficilmente può raggiungere grosse dimensioni, dato il limitato numero di carte. Per uno sviluppatore diventa anche meno intuitivo e veloce utilizzare tale struttura dati per fare le dovute verifiche sugli obiettivi raggiunti.

Per quanto riguarda gli obiettivi, le due architetture si comportano in modo pressoché identico, con l'unica differenza che nell'UML in analisi "l'obiettivo generico" è rappresentato mediante interfaccia, mentre nella nostra proposta esso è una classe astratta con relativo metodo di calcolo punteggio anch'esso astratto. Entrambe le rappresentazioni sono sensate, in quanto garantiscono l'utilizzo del polimorfismo.