Peer-Review 1: UML

Lorenzo Veronese, Giada Silvestrini, William Stucchi

Gruppo GC49

Valutazione del diagramma UML delle classi del gruppo GC59.

Lati positivi

Dall'analisi del modello presentato, sono stati riscontrati i seguenti aspetti positivi:

- Sono state rappresentate le principali entità del gioco: il modello risulta così complessivamente chiaro e la struttura del gioco è ben rappresentata al suo interno. Si può notare un'organizzazione concettualmente sviluppata attorno a tre gruppi principali: il campo da gioco (Playground, Cloud, Island), i giocatori (Player, School, Student, Professor, Tower, Bag, Assistant) e le carte personaggio (Character). Ciò segue i principi dell'Object Orientation, aumentando la modularità, scalabilità e leggibilità del modello.
- Una classe particolarmente evidente è Student, il cui collegamento a numerosi altri oggetti e i metodi correlati (addStudent(), addStudentToDiningRoom(), getStudent() ecc..) ne evidenzia la mobilità all'interno del campo da gioco. Questa classe condivide con Professor il colore, come giustamente sottolineato dall'ereditarietà su Pawn.
- Ottimo l'utilizzo delle Enum, che facilitano la gestione di alcuni tipi di dato e la comprensibilità dell'UML.
- Si ritiene corretto l'utilizzo dello strategy pattern sulle carte personaggio: questo, infatti, potrebbe aiutare a snellire il codice, evitando di definire in modo ridondante ciascuna carta.

Lati negativi

- Il primo problema riscontrato è quello riguardante la gestione delle varie modalità di gioco. È stata infatti definita una sola classe Playground che comprende l'implementazione delle funzioni di tutte le diverse modalità (due, tre o quattro giocatori) e difficoltà (avanzata o semplice), lasciando al Controller il compito di gestirle. Quest'ultimo potrebbe risultare così eccessivamente complesso, dovendosi occupare non solo di controllare l'integrità degli input provenienti dalla View, ma anche di rispettare la logica di ogni modalità di gioco.
- Inoltre, il Controller può far uso solo di funzioni che eseguono azioni estremamente semplici: questo lo obbliga ad agire su più oggetti, chiamando in sequenza numerosi metodi che avrebbero potuto invece essere incapsulati in uno solo, più ad alto livello, riducendo la possibilità di manipolare in modo errato il Model.
- Il punto precedente risalta particolarmente nel caso delle carte personaggio: non avendo un riferimento alle scuole e alla bag, esse non possono modificarle direttamente. Pertanto, il Controller dovrà contenere parte dell'implementazione dell'effetto della carta pur non essendo l'oggetto adibito a tale scopo. È incoerente il fatto che la carta possieda metodi

come addStudentToDiningRoom: secondo la logica seguita nel resto dell'UML, dovrebbe essere il Controller a farsi carico anche di questo aspetto.

- L'UML non sempre è sufficientemente esplicativo. Data la mancanza delle cardinalità sulle frecce dell'UML, non ci è chiaro se ai giocatori venga associato un Team in ogni caso, oppure solo nell'eventualità di una partita a quattro giocatori. Il fatto che il Controller abbia un ruolo così centrale rende difficile capire come verranno utilizzati alcuni metodi.

Confronto tra le architetture

Le scelte che più di tutte hanno portato il nostro modello a differenziarsi da quello dei nostri compagni sono state quelle relative alla gestione delle varie modalità di gioco e la diversa interpretazione dei compiti del Controller. Noi abbiamo scelto di esplicitare le partite a due, tre, quattro giocatori (BoardTwo, BoardThree, BoardFour) e le diverse difficoltà (decorator BoardAdvanced) tramite classi apposite. Questo è stato reso possibile anche dalla centralità che la classe Board assume all'interno del nostro modello: da essa si può risalire ai riferimenti di tutte le entità di gioco. La conseguenza è che il Controller dovrebbe risultare più snello, dovendo utilizzare soltanto il proprio riferimento alla Board per compiere tutte le azioni necessarie e svolgendo operazioni maggiormente ad alto livello. Nell'UML dei nostri colleghi invece la struttura è diversa: il compito di coordinare le entità è svolto da più classi (Playground, School, Character), pertanto sarà il Controller a svolgere la funzione di mediatore tra di esse.
Altre differenze strutturali sono la gestione delle isole, la cui possibile unione in arcipelaghi noi abbiamo deciso di rappresentarla esplicitamente tramite la classe Archipelago, e quella delle monete, che nel modello nei nostri colleghi sono degli int, mentre nel nostro sono definite da una classe Coin apposita. Questi aspetti rendono, a nostro parere, il nostro modello più esplicativo, ma non impattano le possibili azioni da svolgere.
La scelta di raggruppare le varie carte personaggio tramite il pattern strategy aumenta l'espressività del modello, chiarendo il fatto che tali carte sono talvolta accomunate da effetti simili, il che non si nota nel nostro UML.