Peer Review #1

Samuele Pischedda, Angelo Prete, Gabriele Raveggi, Andrea Sanvito
GC11

3 Aprile 2024

Review dell'UML Class Diagram del gruppo GC01.

1 Lati positivi

- Implementazione della classe `Corner` semplice ed efficace.
- Implementazione dello strategy pattern per calcolare i punti delle `ObjectiveCard`.
- Separazione di carte che possono essere giocate (`PlayableCard`) da quelle con scopo diverso (`ObjectiveCard`).
- Utilizzo delle enumerazioni per una migliore chiarezza e manutenibilità; interfaccia `PlayerResources` comune all'enum `Resource` e all'enum `Item`.

2 Lati Negativi

2.1 Generali

- Tutti gli attributi nell'UML sono pubblici, nonostante siano stati aggiunti dei getter.
- Sintassi dell'UML a volte poco chiara (e.g. non è esplicito se `Card` e `Deck` siano classi astratte o meno).
- Non sempre si è fedeli alla nomenclatura di gioco usata nel rulebook (e.g. RadixCards [StarterCards], Wolf [Animal], Mushroom [Fungi], etc).
- Mancanza di alcuni attributi e metodi essenziali al funzionamento delle classi (e.g. non si può risalire alle visible cards pescabili in `Room`, però la classe ha il metodo `getDrawableCards()`).

2.2 Class-Specific

- La classe `GoldenCard`, sottoclasse di ResourceCard, non rispetta il principio di sostituzione di Liskov (dovrebbe invece essere sottoclasse di `PlayableCard`).
- Implementare il "back" di una carta `RadixCard` come sottoclasse di `RadixCard` non è molto chiaro.
- `RadixCard` è implementata come una carta giocabile, ma viene assegnata al giocatore a inizio partita.
- L'implementazione attraverso un Set di posizioni della board di gioco (`Field`) può risultare molto scomoda. Un'implementazione con una matrice o una `Map<Coordinate, PlayableCard>` potrebbe semplificare notevolmente la logica.
- L'interfaccia `CardResources` risulta ambigua a causa di valori sovrapposti, può contenere infatti sia una risorsa (o item) ma anche il valore `FULL`.

3 Confronto tra le architetture

- L'implementazione della classe `Corner` è molto più semplice della nostra, che invece prevede delle sottoclassi in base al tipo di angolo (`VisibleCorner`, `HiddenCorner` e `CoveredCorner`).
- L'interfaccia comune per gli enum `Resource` e `Item` è interessante perché permette di eliminare ridondanza.