Peer-Review 1: UML

Riccardo Corà, Marco Devecchi, Valeria Fasciana

Gruppo AM02

Valutazione del diagramma UML delle classi del gruppo AM32

# Lati positivi

Il modello astrae fondamentalmente la logica di una partita reale:

* La decentralizzazione delle regole del gioco tra Game e Player permette di tener separati i metodi “pubblici” a tutti i giocatori, da quelli che sono invece decisioni del singolo giocatore, facilitandone anche la comprensione a livello visivo.
* Le classi con i loro metodi sono scritti chiaramente e non vi sono ambiguità di nomi.
* La scelta di creare un’interfaccia per ogni effetto facilita l’implementazione dei poteri delle carte Character e permette una futura modifica o aggiunta di altri effetti.

# Lati negativi

* Poteva risultare utile generalizzare le classi contenitrici di studenti, quali Cloud, Island, Board e SchoolBoard, per evitare di ripetere gli stessi metodi della modifica del numero di studenti.
* Non c’è modo di passare attributi agli effetti delle varie carte, e non è chiaro il modo in cui avviene la scelta della carta Personaggio. Il metodo nella classe Game “useCharacterEffect” riceve un oggetto Character, ma quest’ultimo non ha metodi per poter effettivamente applicare l’effetto.
* L’interfaccia EffectHandler sembrerebbe il fulcro del modello, perché collegata alle varie carte, ma non vi è un collegamento diretto per poter essere utilizzata dal Player. Sarebbe più corretto se fosse la classe Player ad avere il metodo, per utilizzare la carta in qualsiasi fase del turno, e in quanto scelta attiva.
* Non è specificato in che modo e in che fase del gioco si calcoli l’influenza. L’attributo colorsInfluenceMultiplier non ha metodi che lo modificano, e l’unico oggetto contenente metodi che la calcolano è l’interfaccia della carta MooshroomMan, che, come detto sopra, non ha metodi per poter essere chiamato. Ma anche a livello di modello sembrerebbe sbagliato, dato che si dovrebbe poter calcolare l’influenza indipendentemente da che si sia chiamata o no la carta durante il turno, soprattutto se la modalità di gioco è quella semplice, perché la classe MooshroomMan non esisterebbe neanche.
* Non c’è una separazione tra le due fasi di gioco, Planning Phase e Action Phase, quindi non è chiaro in che modo si svolga il turno del giocatore, dato che la action phase del singolo player dovrebbe iniziare dopo che tutti i giocatori hanno finito con la planning phase. E non è presente neanche un metodo che imposti l’ordine dei giocatori una volta scelta la carta assistente.

# Confronto tra le architetture

Nel modello del gruppo AM32, la scelta della modalità di gioco va a modificare un singolo attributo della classe Game, expertMode, e parrebbe essere più efficace rispetto all’avere un’enumerazione con sole due modalità, come invece abbiamo fatto noi.

La rete di implementazione dei Character, collegata agli specifici ambiti, è implementata in modo che non si debbano fare molte chiamate ad oggetti differenti, ma si parte sempre dal nucleo, l’interfaccia EffectHandler. Noi invece abbiamo fatto in modo che tutti i metodi di modifica del gioco stessero nell’interfaccia GameAvailableAction, che risulta più confusionario perché si responsabilizza una sola classe generica, rispetto al modificare solo tramite classi dell’ambito preso in considerazione tramite la carta scelta.