Peer-Review 1: UML

Roberto Cialini, Umberto Colangelo, Vittorio La Ferla

Gruppo AM29

Valutazione del diagramma UML delle classi del gruppo AM02.

Lati positivi

- In generale, la logica di gioco descritta nell'UML è fedele alla logica di gioco reale. Classi e metodi sono di immediata comprensione; ciò permette di afferrare senza difficoltà la struttura del gioco, favorendo la cooperazione tra più sviluppatori.

- Le interfacce GameAvailableActions e PlayerAvailableActions permettono di filtrare efficientemente i metodi che possono essere invocati dal controller rispetto a quelli presenti in Game e Player, non esposti e pensati per la gestione di ulteriori dinamiche di gioco.

- La scelta di decentralizzare la logica di gioco tra Game e Player permette di non mantenere quasi tutte le informazioni all'interno un'unica classe, rendendo più agevole una successiva modifica concettuale del gioco.

Lati negativi

- A primo impatto visivo, la disposizione di classi e interfacce nel class diagram non indica chiaramente la gerarchia della logica di gioco; infatti, numerose classi o enumerazioni sono collegate direttamente con le classi principali (Game e Player) senza la presenza di interfacce, classi astratte o modalità di filtraggio dei metodi disponibili e delle possibili modalità di gioco (come invece fatto per le carte esperto con l'interfaccia StrategyEffects).

- Non considerare gli studenti come una vera e propria classe, ma come "interi", sicuramente semplifica notevolmente le dinamiche di gioco, come ad esempio nella gestione della posizione delle pedine, ma è un approccio di tipo procedurale che potrebbe limitare una futura estensione del gioco (come, ad esempio, dare il nome agli studenti o aggiungere un effetto speciale diverso per ogni colore).

- La scelta di includere e di gestire nella classe IsleGroup anche le isole singole, attraverso il parametro size, potrebbe anche sembrare un modo di ottimizzare il modello, ma non risulta chiaro in che modo avvenga la gestione delle informazioni relative alle specifiche isole nel momento in cui vi sono delle aggregazioni che modificano la size del gruppo di isole.

Confronto tra le architetture

L'uso di Player e Game per decentralizzare la logica di gioco è interessante in quanto permette di rendere il gioco più facilmente estendibile e di rispettare i parametri di manutenibilità tipici della programmazione Object Oriented; questa scelta è stata valutata anche dal nostro gruppo durante la stesura dell'UML. L'utilizzo delle enumerazioni Phase e PlayerState, per gestire le fasi del gioco e del player, è sicuramente da tenere in considerazione per rendere più agevole ed ottimizzata la gestione di informazioni per logica del gioco e per facilitare il controllo delle azioni disponibili in una determinata fase della partita.