Peer-Review 1: UML

Rigat, Schiavoni, Avarino

Gruppo GC18

Valutazione del diagramma UML delle classi del gruppo GC28.

Lati positivi

Aver fatto delle sottoclassi per la modalità esperti potrebbe aver del potenziale, non sappiamo come è stato implementato ed avendo preso una strada diversa non possiamo darne un giudizio completo.

Inoltre l'idea alla base della fusione delle isole è una cosa che condividiamo.

Lati negativi

Innanzi tutto l'UML è poco chiaro e non completo:

- Ci sono attributi istanziati come classi ma queste non sono presenti nell'UML, come Game_Phase, Game_Checker.
- Alcune classi non sono ben collegate tra loro, ad esempio come fa Student_sack a non essere collegata a nulla? E visto che avete creato questa classe perché in Game_Manager_Normal l'attributo student_sack è istanziato come arraylist?
- A molti metodi manca cosa ritornano, rende difficile capirne il funzionamento solo dal nome.
- Alcuni attributi sono istanziati in maniera dubbia: oltre al caso di student_sack mostrato prima, come fa Coin_Reserve ad avere un attributo void? Cosa serve?
- Check_Base serve a gestire le tre carte personaggio giusto? Perché non è collegato a Character_Card in qualche modo? Come avviene la gestione delle carte personaggio? Come viene passato l'effetto delle tre carte personaggio in Check_Effect_1/2/3? se viene fatto.
- Perché la gestione dei colori delle torri non viene gestita anch'essa da una Enum? Non sarebbe più semplice?

Fatta questa precisazione su alcuni aspetti tecnici, ora vediamo alcune cose a noi poco chiare sull'impostazione e l'implementazione dell'UML:

- Avere la gameboard integrata nel controller è sconveniente a nostro avviso, nel controller ci dovrebbero essere solo i metodi che vanno ad interagire con il model e la gameboard fa parte del model.
- Scrivere un metodo per ogni diverso numero di giocatori è poco produttivo, non ha senso riscrivere due metodi che fanno la stessa cosa ma che variano per un parametro… ad esempio Student_Sack_To_Cloud_2() e Student_Sack_To_Cloud_3(), non ha più senso fare un metodo unico che va bene per ogni modalità di gioco e in base a questa modalità fa determinate cose?

- Per la gestione degli studenti nella schoolboard è utile tenere traccia degli “free_students_disk”? inoltre, va bene aggiungere un’altra classe a schoolboard per la gestione degli studenti per ogni colore, ma sono necessarie due classi?

Confronto tra le architetture

Il nostro approccio allo sviluppo del gioco è molto diverso da questo UML, implementare qualcosa che hanno fatto loro stravolgerebbe il nostro lavoro.

Proprio per questo ultimo motivo, nella valutazione siamo stati un po’ duri; avendo implementato molte cose in modo diverso, alcune cose ci sembrano, senza vedere il codice, poco chiare. Non abbiamo tuttavia scritto tutte le “differenze” con il nostro perché poi una volta implementate possono essere giuste le vostre intuizioni.

Vi lasciamo infine con una riflessione che un tutor ci ha fatto: se un domani dovesse aggiungere nuove funzionalità al gioco, riuscireste ad implementarle senza stravolgere completamente il gioco? Pensate in ottica di dover andare a ritoccare il codice il meno possibile.
Un suggerimento in quest’ottica che sentiamo di darvi è creare una classe dove mettete dentro tutte le variabili globali… in questo modo chiamando per esempio NomeClasse.NumeroPlayers potete accedere al numero di giocatori da qualsiasi punto del gioco.

Ps: A seguito di uno scambio di email il gruppo GC28 ci ha fatto sapere che sono state aggiornate alcune cose, ma a noi è stato fornito questo materiale e non c’era più tempo per una nuova analisi.