Peer -Review 1: UML

Andrea Albergo, David Loredan Barb, Edoardo Bozzini

Gruppo AM48

Valutazione del diagramma UML del gruppo AM21.

1. Aspetti negativi:

a) **Classe “GameManager”:**
La classe di per se è ottima, ha una struttura simile alla nostra per quanto riguardano gli attributi e l’interazione con il controller. Pero risulta il problema di avere una classe ampia con molte linee di codice e a ciò consiglierei una maggior modularizzazione creando una classe al posto del metodo “setupBoard()”, da utilizzare all’inizio del gioco prima che il gioco abbia inizio, e al posto di metodi come “setupRound()” e simili creare una classe che gestisca il turno generale del gioco e il turno in cui ogni singolo giocatore avrà il proprio display illuminato con comandi disponibili da cliccare per svolgere le azioni possibili e rimaste.

b) Non è chiaro il metodo “equals()” in classe Player.

c) Non è chiaro il procedimento di scelta tra modalità semplice e avanzata del gioco da parte del giocatore host: è dentro “SetupBoard()”? Se sì, guardate punto a).

d) Classe “IslandGroup”: Non è chiaro come gestite il conto delle torri in caso di isole unificate, cioè manca una sorta di metodo “howManyTowers()” per il calcolo delle influenze.

2. Aspetti positivi:

a) Frecce d’associazione corrette e organizzate bene;

b) Apprezzatissima il pattern state dell’interface “InfluenceCalculator” in caso di carte personaggio attive e molto probabilmente prenderemo spunto da esso;

c) Ottima la distinzione tra Board e Board Manager, perchè lasciate al BoardManager i metodi da utilizzare con il controller mentre il Board è il gioco in se, la parte visuale.

d) Interessante l’utilizzo della classe “Professor Ownership” con la relativa interfaccia per l’assegnamento e gestione dei professori.

e) Abbiamo provato a simulare una partita di Eryantis con modalità semplice e avanzata, non è C-style, va tutto bene solo da sistemare qualche classe elencata in sezione “Aspetti Negativi”.

3. Confronto con la nostra architettura:

La scelta del BoardManager è come la nostra, infatti abbiamo fatto in modo da interagire con il controller e nel gestire il gioco. Da BoardManager noi abbiamo implementato una classe Round con cui gestire I turni di ogni singolo giocatore e si passa al successivo una volta che tutte le nuvole siano vuote.
Anche noi abbiamo utilizzato la classe astratta CharacterCard per la modalita esperti. Stesse enum, non abbiamo classi come ProfessorOwnership e InfluenceCalculator ma prenderemo spunto da esse per modificare le nostre classi di SchoolBoard e gestione influenze. Inoltre noi abbiamo implementato una classe associata al BoardManager, detta StartGame, con cui scegliamo nomi giocatori, il wizard, colore, scelta modalità di gioco e setup del tavolo da gioco. Tale classe interagirà con il controller attendendo i giocatori nella lobby pre-partita.