Descrizione dell'UML

Marco Paradina, Samuele Peri, Davide Palmiotti
Gruppo 22

4 maggio 2022

Descrizione del diagramma UML delle classi del gruppo 22.

1 Model

Lo stato è principalmente contenuto nel model. I nomi di classi, attributi e metodi sono autoesplicativi. Di seguito una breve descrizione delle classi il cui comportamento potrebbe non risultare evidente dal nome:

Hand rappresenta le carte che ha in mano un giocatore

IslandWrapper centralizza lo stato di tutte le isole e permette di effettuare operazioni su di esse, come verificare la posizione di madre natura e unire più isole

CloudWrapper svolge una funzione simile a IslandWrapper ma per le nuvole

Shop gestisce i pagamenti in monete che si devono fare per poter usare le carte personaggio nel gioco per esperti

2 Controller

Game è il controller centrale: riceve dalla RemoteView diversi tipi di eventi e dispone di diversi metodi HandleEvent in overload per gestirli. Il controller è decentralizzato su più componenti interni, ognuno dei quali svolge il ruolo di controller su aspetti diversi del gioco. Ad esempio, CloudController svolge

il ruolo di controller sulle Cloud. In questo modo l'implementazione risulta più pulita e ordinata, e se in futuro si volessero apportare modifiche al gioco esse sarebbero più facili da implementare. I nomi di classi, attributi e metodi sono autoesplicativi. Di seguito una breve descrizione delle classi il cui comportamento potrebbe non risultare evidente dal nome:

HandleEvent gestisce gli eventi che arrivano dalla view (si veda paragrafo View) e come essi modificano il model. Per ogni tipo di evento c'è un overload del metodo HandleEvent.

Game.playersOrder cambia ad ogni turno e rappresenta l'ordine in cui i giocatori devono giocare

ExpertGame ha per metodi gli override dei metodi di game per poter applicare le regole del gioco per esperti

3 View

Le interazioni dell'utente con la UI genera diversi tipi di evento. Questi eventi sono instanziati con un factory method e quindi serializzati con Gson, inviati attraverso la rete, ricevuti dalla RemoteView e per ultimo dal controller, dove le informazioni contenute dal singolo evento verranno utilizzate per modificare il model.

Conclusione

In generale il progetto segue il pattern MVC facendo uso di Observer/Observable come visibile nel diagramma UML. Nel Sequence Diagram (1) allegato è riportata la sequenza di interazioni tra i componenti dell'applicazione per gestire le interazioni dell'utente in base alla logica di gioco. Il Sequence Diagram (2) mostra invece come una generica partita viene inizializzata, chiarendo l'utilizzo delle classi inserite nella parte di Network dell'UML.