Breve descrizione UML model

GC51

marzo 2022

1 Model

Il modello è rimasto pressoché invariato rispetto all'ultima volta, se non per l'aggiunta di qualche funzione di utility e la rimozione delle seguenti classi

- Student
- Tower
- LockCard

2 Controller

Il paradigma scelto per l'implementazione del controller è quello ad eventi: il controller riceve da *qualcuno* un evento di update, lo processa e aggiorna il model di conseguenza. L'evento di update è gestito dalla classe **Action**, che è customizzabile facilmente per poter accogliere qualsiasi tipo di evento relativo al gioco (dalla selezione di un'isola all'utilizzo di una carta personaggio, per intenderci)

3 Network

Il carico di lavoro client-server è estremamente sbilanciato verso il server, che gestisce il model, il controller e parte della view. Il client, ovviamente thin, gestisce la restante parte della view e gli input dell'utente (con un controllo minimale su di essi, in modo tale da poter rifiutare quantomeno quelli evidentemente errati - esempio: *seleziono l'isola numero 15*).
Il server può gestire più partire in contemporanea. L'intera comunicazione client-server si basa dapprima sui client che indicano il proprio username e il numero di giocatori con cui vogliono competere; effettuato il matching e validate le condizioni necessarie per iniziare la partita, inizia un flusso continuo di comunicazione tra client e server in cui

- Il server rende note al client che deve giocare (per ordine di turno) la fasi di gioco che esso può giocare
- Il client si occupa di creare un oggetto *Action* chiedendo all'utente sia la fase di gioco in cui egli vuole giocare (usando, ovviamente, la lista di fasi accettabili ottenuta dal server) e compilandola con i parametri necessari (ad esempio, quali studenti spostare in quale posto)

- Il client invia la *Action* al server
- Il server processa la *Action* e
  - Se i parametri indicati sono incompatibili con le regole del gioco, notifica il client con un messaggio dettagliato di errore e viene ripetuto tutto fino all'ottenimento di dati sensati
  - Se i parametri sono sensati, il server notifica il client della corretta esecuzione della mossa e invia una rappresentazione testuale del modello (che il client stamperà) e si passa quindi alla prossima fase di gioco, ripetendo il procedimento qui illustrato

Di fatto quindi il server gestisce ogni singolo aspetto del gioco, effettuando anche i dovuti controlli, mentre il client si limita a inviare le richieste relative alla corrente fase di gioco e mostrandone l'output all'utente

4 Allegati

In allegato trovate, oltre a questo documento, l'UML (model, controller, network) e il sequence diagram del protocollo