Documentazione relativa all'UML del gioco Codex Naturalis per Peer Review 2

Gruppo 32: Bergagna Matteo, Brun Antony, Chen Jie, Chiroli Lorenzo

All'interno del nostro progetto abbiamo deciso di implementare sia RMI sia comunicazione via Socket TCP: entrambe le tecnologie vengono generalizzate per essere impiegate nel trasferire "messaggi" tra client e server. Nel caso di RMI, questi messaggi saranno oggetti Java puri mentre, nel caso di comunicazione Socket, quest'ultimi dovranno essere prima serializzati per poi essere inviati sul canale.

Il cliente, sia che usi RMI, o che usi Socket, manderà quindi degli oggetti di tipo Message, al server, che, eseguendo l'unico metodo associato a tale messaggio, stimolerà il controller nella maniera opportuna (a modo di Strategy Pattern).

Tutti i messaggi implementano, in maniera differente, a seconda dello stimolo da produrre, una delle seguenti interfacce:

- CtoSMessage: rappresentano i messaggi inviati dal client al server nella fase di gioco;
- StoCMessage: rappresentano le risposte inviate dal server al client;
- CtoSLobbyMessage: rappresentano i messaggi inviati dal client al server nella fase di lobby.

LobbyMessage differisce da CtoSMessage per il fatto che i primi stimolano il GamesManager, la classe dedita alla gestione delle partite multiple, mentre i secondi interagiscono con il GameController, proprio di ogni singolo match.

La disconnessione dei giocatori viene rilevata mediante la tecnica dell'"heartbeat": il Player manda periodicamente un ping al server se ciò non avviene l'host, che deve rispondere con un "pong" a queste richieste, considera il giocatore come disconnesso.

In allegato a questo documento, si trova la lista dei messaggi che costituiscono il protocollo di comunicazione classificati in base alla coppia mittente-destinatario. Il protocollo è stato pensato per essere resiliente alle disconnessioni e permettere a un singolo server di gestire partite multiple: assieme alla chat queste sono le tre funzionalità avanzate che intendiamo implementare.

Inoltre, alleghiamo anche i vari sequence diagrams che contestualizzano questi messa ges rendendo esplicita la comunicazione e l'invocazione dei metodi che questi produco no.

Si prega di fare attenzione che in questi diagrammi sequenziali notazioni come "CtoSM essage" non fanno riferimento a una specifica classe ma a una particolare istanza di un determinato messaggio; alla stessa maniera "PersonalVirtualView" si riferisce alla Virtu alView del singolo giocatore mentre "AllVirtualView" si riferisce alle virtualview di tutti i

player. Per un maggiore dettaglio delle classi riferirsi all'UML di controller&network forn ito.

In alcuni diagrammi può sembrare che le VirtualView lancino metodi autonomamente ma ciò è possibile grazie al fatto che queste sono eseguente da thread separati seguen do il pattern "observer".