Protocollo di comunicazione

Architettura del server e struttura delle connessioni
Il paradigma MVC distribuito implementato è molto simile a quello visto in classe.
Esso presenta una classe Server che gestisce la lobby iniziale di gioco e che presenta il server socket per la connessione iniziale. Quando un client si connette al server, viene creata un'istanza della classe ClientSocketConnection che gestisce la comunicazione tra lo specifico client e il server.
Il patter MVC implementato presenta un model che contiene la maggior parte della logica (in particolare, le classi Game e Round gestiscono lo svolgersi della partita), una classe Controller che rappresenta il controller e una classe RemoteView. Come da pattern, si ha che, all'avvio della partita, tali classi vengono istanziate. Il controller sarà osservatore delle view e le view saranno osservatrici del model. Lo svolgimento e aggiornamento della partita avverrà grazie a questo pattern e allo scambio di appositi messaggi, scatenato da eventi lato client, o modifiche lato server.

Login e inizio della partita
La fase iniziale di setup serve a scegliere la tipologia di partita che si vuole giocare (numero di giocatori e modalità) e a inserire il proprio nome di gioco. Tale fase prevede le scambio di messaggi di setup (classe SetupMessage). Il server invia la richiesta di un messaggio specificando il tipo di risposta che attende, tale informazione è specificata mediante l'enumerazione ConnectionState.
Le fasi di setup sono 3:

1. richiesta/invio del nome utente;
2. richiesta/invio della modalità di gioco (normal o expert);
3. richiesta/invio del numero di giocatori della partita ricercata.

Dopo aver inserito le informazioni correttamente, il giocatore viene inserito in una partita e viene atteso il collegamento del numero necessario di giocatori per avviarla. Raggiunto tale numero la partita inizia.

Abbiamo deciso di non implementare, nella fase di setup, la scelta dei maghi in quanto, all'interno nostra implementazione del progetto, non cambierebbe alcun aspetto del gioco.

Tipi di messaggio di setup
Facendo riferimento alle fasi di setup precedentemente citate, i connection state scambiati sono i seguenti:

1. ConnectionState.LOGIN;
2. ConnectionState.MATCHMODE;
3. ConnectionState.NUMBEROFPLAYERS.

Messaggi inviati dal giocatore
I messaggi di gioco inviati dai client sono sottoclassi di PlayerMessage. Essi ridefiniscono il metodo performMove(Game game) della loro sopraclasse. Ogni sottoclasse eseguirà una specifica azione sul modello e il nome del messaggio è esplicativo della funzione svolta.

Tipi di messaggio inviati dal giocatore
I messaggi inviati dai client durante il gioco sono:

- PlayAssistantMessage
- AddStudentOnIslandMessage
- AddStudentOnTableMessage
- ChangeMotherNaturePositionMessage
- GetStudentsFromCloudMessage

I messaggi vengono inviati dal client e raggiungono la classe ClientSocketConnection. Tale classe notifica la classe MessageReceiver (sua observer) della RemoteView. Viene allora chiamato il metodo handleMove(PlayerMessage message) della RemoteView che notifica l'arrivo di un messaggio al controller (controller) che è suo osservatore.

All’arrivo di un messaggio la classe Controller esegue il metodo performMove(PlayerMessage message) che a sua volta esegue performMove(Game model) del messaggio sul modello. Quest’ultimo metodo permette di effettuare la modifica richiesta aggiornandone lo stato di gioco.

A ogni aggiornamento dello stato viene invitato un messaggio al Client tramite la RemoteView.

Messaggi relativi ai characters
In aggiunta ai messaggi standard ci sono due messaggi relativi alla modalità “expert”. Per attivare i character (classe Character) la logica è analoga a quella degli altri messaggi con l’eccezione che il messaggio ActivateEffectMessage può essere mandato in qualsiasi istante della “action phase” del player che è di turno. Nel caso il character richieda ulteriori input essi verranno richiesti e successivamente inviati con il messaggio DoYourJobMessage. Tale messaggio è gestito sempre in modo analogo agli altri.

Messaggi inviati dal server
I messaggi di gioco inviati dal server sono sottoclassi di GameMessage, o la classe GameMessage stessa. Tale classe contiene tutte le informazioni necessare per renderizzare il campo di gioco, nel caso della GUI, oppure per effettuare le dovute stampe, nell’opzione di gioco CLI.
Nel caso si giochi da CLI i messaggi scambiati sono di tipo GameMessage. Le classi relative al client stampano a video il modello richiamando i metodi presenti all’interno di tale messaggio.
Nel caso si giochi da GUI i messaggi scambiati sono sottotipi di GameMessage. Ogni sottotipo richiede il rendering di diversi aspetti grafici, a seconda di cosa è stato modificato dalla precedente richiesta dei client. Per fare ciò, tali messaggi ridefinisco il metodo renderWhatNeeded(BoardController controller) della loro sopraclasse. Ogni sottoclasse eseguirà alcuni specifici metodi della classe che controlla il rendering della scena (BoardController). I messaggi relativi ai Character renderizzano nuovamente l’intera grafica, per questo il rendering è più lento (non abbiamo avuto sufficiente tempo per correggere tale problema).

Tipi di messaggio inviati dal server
I messaggi inviati dal server durante la partita sono:
- PostPlayAssistantMessage
- PostAddStudentOnIslandMessage
- PostAddStudentOnTableMessage
- PostChangeMotherNaturePositionMessage
- PostGetStudentsFromCloudMessage

I nomi sono autoesplicativi per quanto riguarda il motivo per cui vengono inviati dal server. Ognuno viene inviato in risposta a una specifica richiesta di azione di gioco da parte dei client (nome del messaggio a cui si toglie il prefisso ”Post”).

Messaggio TerminatorMessage
Il messaggio di tipoTerminatorMessage serve a segnalare due cose:
- se inviato dal client, serve a notificare che il client si sta disconnettendo dalla partita;
- se inviato dal server, serve a notificare ai client che un giocatore si è disconnesso, se esso era in una partita, il server notifica agli altri giocatori che essa è terminata (non è stata implementata la resilienza alle disconnessioni).

Tale messaggio può arrivare in qualsiasi istante, sia nella fase di ingresso in un match che durante lo stesso.