Design del protocollo di rete

Gruppo 14 (Arbasino, Amadori, Cassenti)

Server e client comunicano attraverso messaggi resi tramite una classe astratta Message estesa da classi messaggio più specifiche.

I messaggi una volta compilati nei loro parametri verranno codificati e inviati sottoforma di file Json.

La classe Message possiede 3 attributi chiave:

1. MessageMain : enumerazione per la gestione dei controlli tra i sottotipi di controller
2. MessageSecondary : enumerazione per la gestione interna ai singoli controller
3. PlayerId : invia al player corrispondente all’id, se -1 invia broadcast

## Divisione controller

Ogni messaggio viene accettato e processato dal MainController che , dopo aver controllato che il player da cui il messaggio è partito sia il giocatore corrente (in caso contrario manda un messaggio di errore) e che tale messaggio sia mandato in un momento del turno consono all’azione svolta mandando un messaggio di errore in caso contrario, tramite uno switch invoca un metodo del Controller relativo alla gestione della suddetta azione che la gestisce e modifica il model.

## Messaggi da client a server

Ogni 5 secondi viene mandato in autonomia dal client al server un messaggio di ping che verifica la connessione, in caso il ping non arrivasse entro 10 secondi il gioco viene automaticamente interrotto tramite un messagio di disconnessione e tutti i client vengono chiusi

### Fase di login

1. Il client connettendosi manda un messaggio con il suo username settato.
2. Al primo player connesso viene mandato un messaggio per il set dei parametri di gioco (modalità e numero di player)
3. Si procede ad accettare gli altri giocatori tramite il messaggio di nickname, che in caso sia gia presente nella partita lo comunica al nuovo utente
4. Il gioco inizia autonomamente una volta raggiunto il numero di giocatori richiesto e settati i nickname

### Fase di gioco

Durante ogni fase di gioco al giocatore verranno fornite informazioni sullo stato del gioco e sulla sua prossima azione da fare tramite l invio di ClientResponse con una frase settata che verrà stampata a video

### Fase planning

1. Al primo giocatore viene chiesto di giocare un assistante, egli compilera un PlayMessage che verrà gestito dal PlayController(in caso di assistant gia giocato viene notificato il giocatore in quanto l’input non è valido)
2. Il giocatore successivo riceve un messaggio quando è il suo turno di giocare un assistente
3. Dopo che tutti i giocatori hanno giocato gli assistanti verrà notificato il primo giocatore

### Fase action

1. Move student per un numero di volte pari al numero di studentOnCloudTile (gestito dal MoveController) svolto tramite un MoveMessage
2. Move mother nature (gestito dal MoveController) svolto tramite un MoveMessage
3. Move cloud to entrance (gestito dal MoveController) svolto tramite un MoveMessage

Il messaggio di move incorpora queste 3 azioni modificando i parametri al suo interno a seconda di cosa il player vuole fare.

Durante ogni azione della fase di action può arrivare un messaggio di Play (nel caso di un character) che verra accettato e processato dal PlayController.

In caso il PlayMessage non contenga tutti i campi di cui il Character ha bisogno per essere giocato si provvede a notificare il client con un’apposita ClientResponse.

### End phase

Dopo aver preso la cloud tile se il round non è ancora terminato verrà inviato un ClientResponse al prossimo giocare altrimenti si notifica il player che ha appena terminato il turno che deve giocare l’assistant per dare inizio a un nuovo round

### Victory

Se una delle condizioni di vittoria viene raggiunta tutti i giocatori vengono notificati tramite un VictoryMessage che racchiude l’Id del team vincitore e i nickname dei player che lo compongono