COMMUNICATION PROTOCOL

Each client initially tries to login as a player: if the is operation is successful the server replies with a success message, otherwise the server sends an error message.

Starting from when the first client successfully connects, a ping message is sent from each client to the server, to which the server responds with a Pong message, at fixed intervals in order to check whether all the connections are still active: if that's not the case, the game ends and the server notifies each client (or, if the server is the one whose connection was shut down, each client takes notice and exits the game).
(To be modified if 'resilience' advanced function is implemented)
Each message exchanged between client and server will be JSON objects.

Upon a change in the model, a specific message regarding such change is sent to the client, so that the client can update the locally stored version of the model.

The communication can be divided into 7 phases, each accepting different message types:

1) Login
2) Prepare
3) Plan
4) Move student
5) Move Mother Nature
6) Pick cloud
7) End game

There are different types of messages that can be exchanged:

UTILITY MESSAGES

- PingMessage()
- PongMessage()

CLIENT -> SERVER

- LoginRequestMessage(nickname, numberOfPlayers, gameMode)
- WizardRequestMessage(wizardName)
- AssistantRequestMessage(assistantID)
- MoveStudentMessage(color, destination)
- MoveMotherNatureMessage(numberOfSteps)
- PickCloudMessage(cloudID)
- CharacterInfoRequestMessage(characterID)
- CharacterMessage(characterID)
- CharacterColorMessage(characterID, color)
- CharacterDestinationMessage(characterID, destination)
- CharacterDoubleColorMessage(characterID, firstColor, secondColor)
- CharacterColorDestinationMessage(characterID, color, destination)

SERVER -> CLIENT

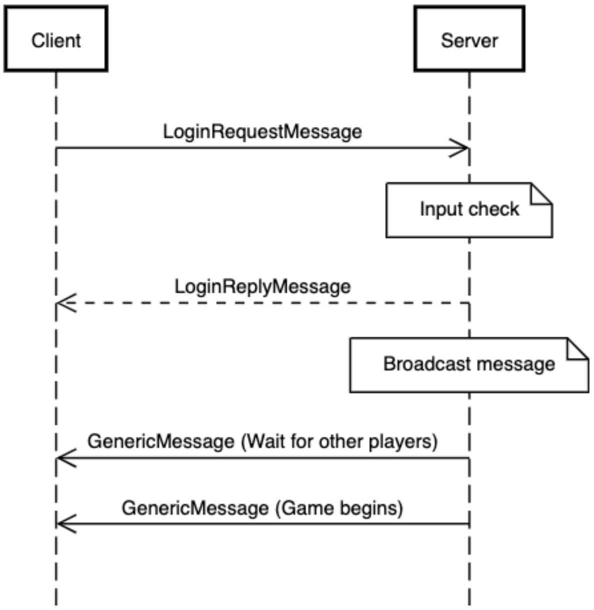
- LoginReplyMessage(connectionAccepted, nickname accepted)
- WizardsAvailableMessage(wizardNames)
- AssistantsMessage(assistants)
- BoardMessage(board)
- CharacterInfoMessage(characterID, description)
- CharactersDrawnMessage(characters)
- CloudsMessage(clouds)
- ColorsAvailableMessage(colors)
- CurrentPhaseMessage(currentPhase)
- CurrentPlayerMessage(currentPlayerNickname)
- DisconnectionMessage(nicknameDisconnected, messageDisconnection)
- ErrorMessage(error)
- GenericMessage(message)
- IslandGroups(islandGroups)
- IslandMessage(island)
- MatchInfoMessage(players, characters, boards, islandGroups, currentPlayer)
- CoinMessage(coins)
- WinnerMessage(winnerNickname)

Sequence diagrams

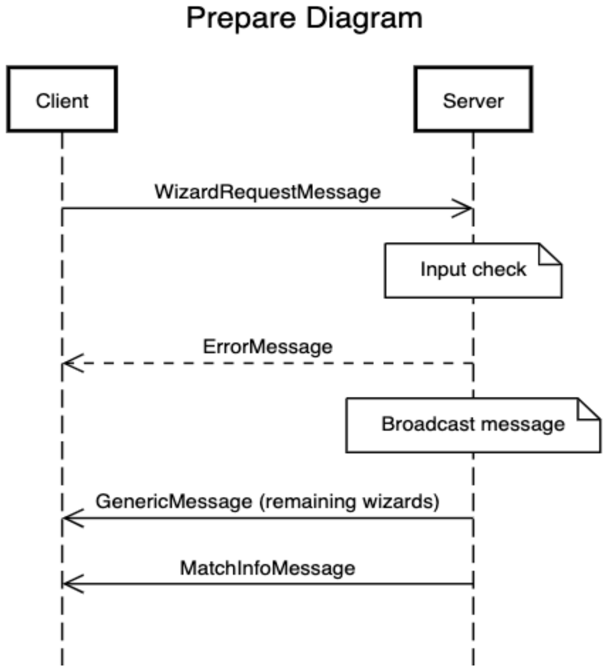
LOGIN

The client sends a message containing their nickname and preferences, such as number of players and game mode. If the connection is successful and the nickname is unique, the server broadcasts a message to all the players, that varies depending on whether the desired number of players has been reached.

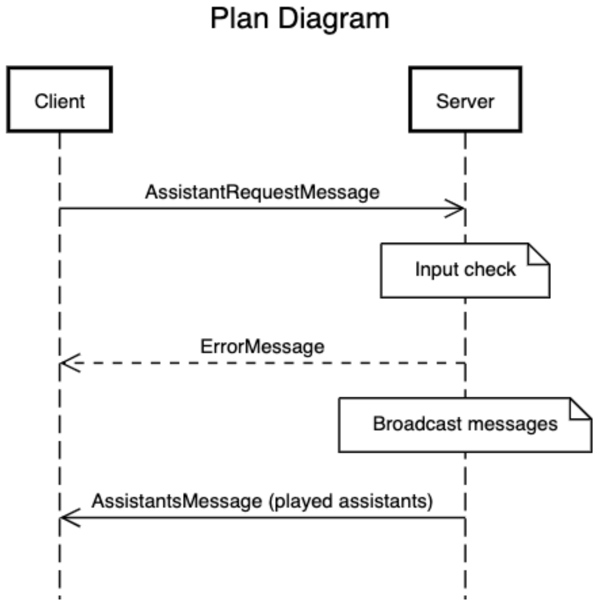
Login Diagram



PREPARE
The client chooses a wizard. When every player has chosen a wizard, the game broadcasts a message with all the necessary information for the clients to locally store the model.

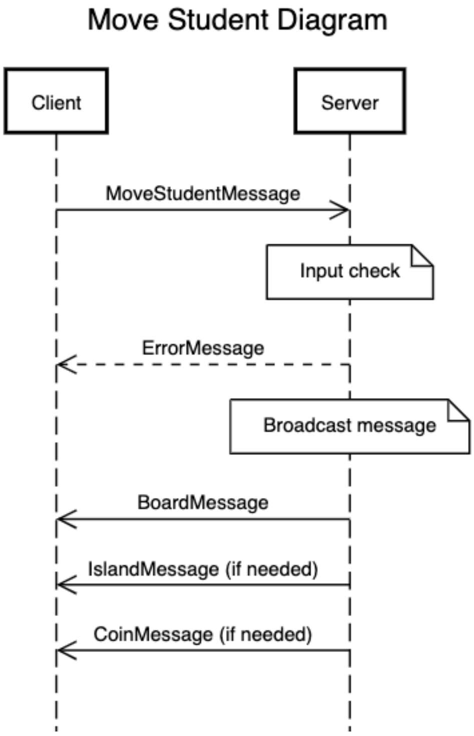


PLAN
The client chooses an assistant to play. The server then broadcasts a message to inform every player of the assistant played.



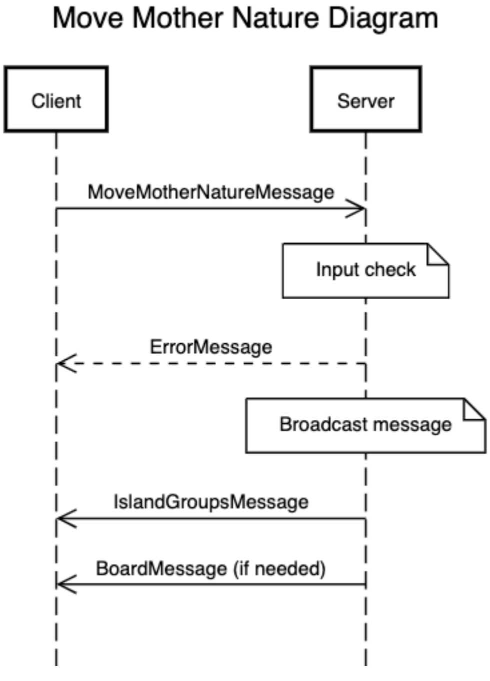
MOVE STUDENT

The client sends a message containing the desired move (student color and its destination). The server responds with specific messages to update the board and, eventually, the island to which the student was moved. An additional message informs the client if a new coin is gained.



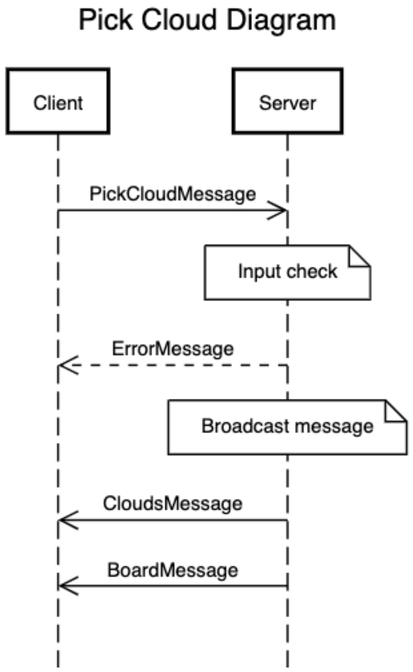
MOVE MOTHER NATURE

The client sends a message containing the number of steps Mother Nature has to take. The server responds in order to update every player's version of the model. An additional message informs the players if an island group was conquered, updating the boards as well since a change of towers occurred.



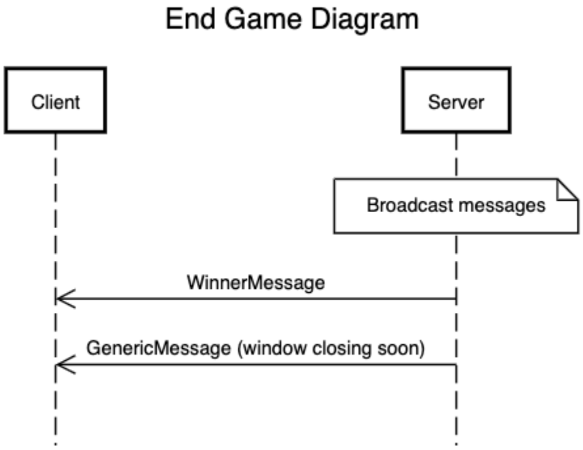
PICK CLOUD

The client chooses a cloud to get the students from. The server sends an update to every player.



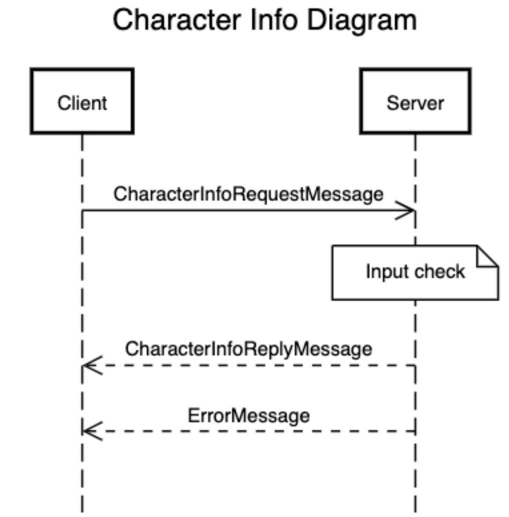
END GAME

The server broadcasts the nickname of the winning player.



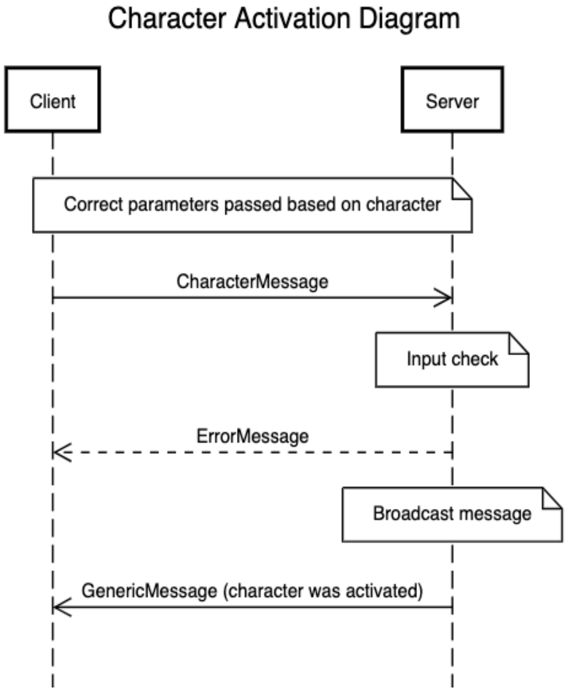
CHARACTER INFO

The client requests information about a certain character. The server responds to the client by providing such information.



CHARACTER

The client sends a message containing the ID of the character to activate, together with other parameters eventually required by the desired character (such as color or destination). The server broadcasts the activation of the character, together with its effect.


Character Activation Diagram
Client
Server
Correct parameters passed based on character
CharacterMessage
Input check
ErrorMessage
Broadcast message
GenericMessage (character was activated)