Explicative Text sequence diagrams

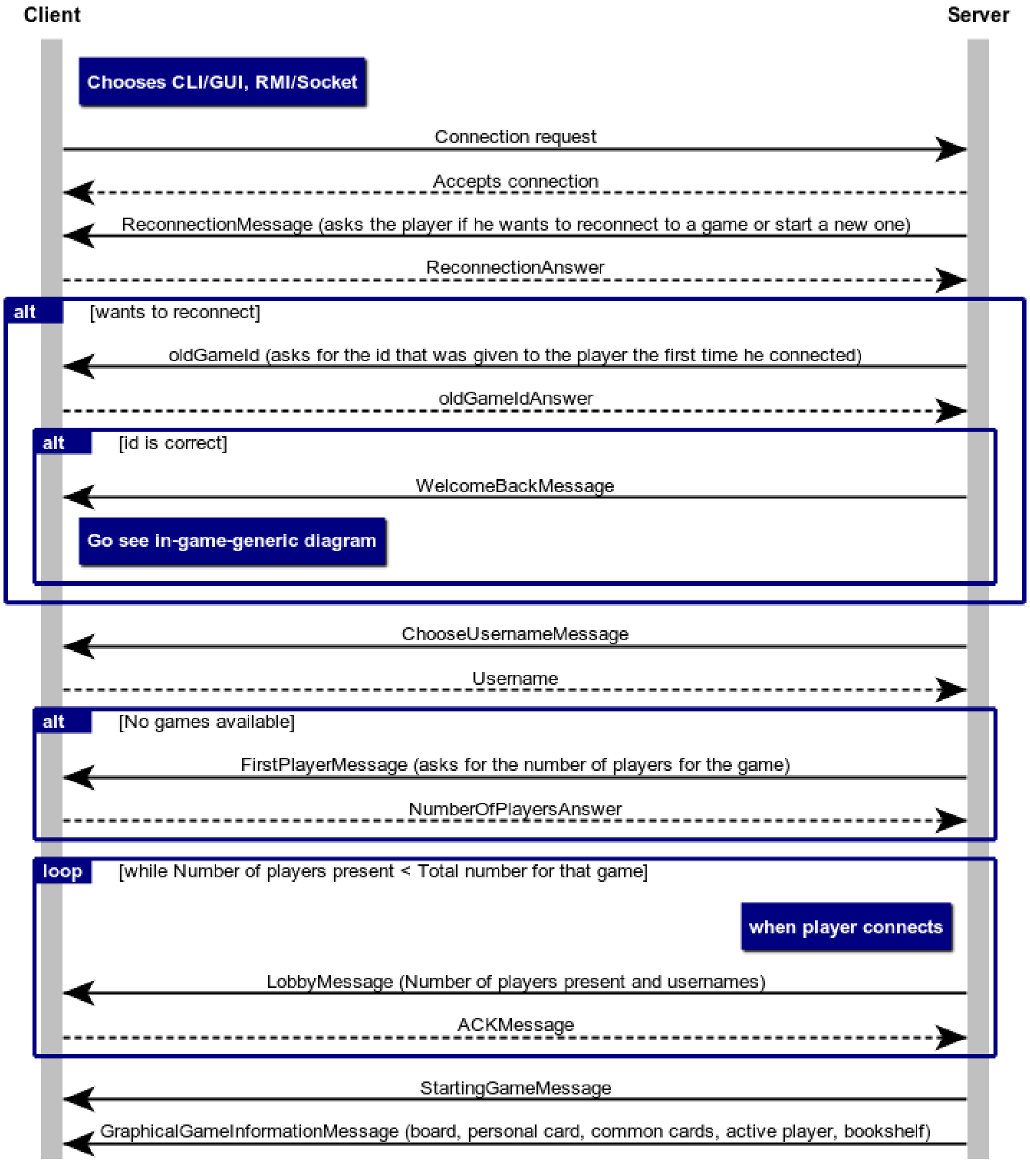
Samuele Pietro Galli, Alessandro Fornara, Donato Fiore, Edoardo Gennaretti

29 giugno 2023

1 Player Login

- **Connection request**: Client sends a first connection request to join the server and starts waiting for an answer from the server.
- **Accepted connection**: Server responds to the client.
- **ReconnectionMessage**: Ask if a player wants to reconnect
- **ReconnectionAnswer**: Message sent from the client
- **(WANTS TO RECONNECT) OldGameId**: asks the player the previous game id
- **(WANTS TO RECONNECT) oldGameIdAnswer**: sent from the client
- **(CORRECT ID) WelcomeBackMessage**: reference to in-game-generic diagram
- **ChooseUsernameMessage**: Client sends the username to the server.
- **Username**: sent from the client
- **(NO AVAILABLE GAMES) FirstPlayerMessage**: Server asks for the number.
- **(NO AVAILABLE GAMES) NumberOfPlayersAnswer**: Clients gives the number of players.
- **(LOOP) LobbyMessage**: Server sends a message with the number of players in the game and their respective usernames. Loop ends when the total number of players has been reached.
- **ACKMessage**: Server starts the match.
- **StartingGameMessage**: Server sends to the client all the information for the match (personal card, board, common cards).
- **GraphicalGameInformationMessage**: board, personal card, common, active player, bookshelves.

Player login


Client
Server
Chooses CLI/GUI, RMI/Socket
Connection request
Accepts connection
ReconnectionMessage (asks the player if he wants to reconnect to a game or start a new one)
ReconnectionAnswer
alt
[wants to reconnect]
oldGameId (asks for the id that was given to the player the first time he connected)
oldGameIdAnswer
alt
[id is correct]
WelcomeBackMessage
Go see in-game-generic diagram
ChooseUsernameMessage
Username
alt
[No games available]
FirstPlayerMessage (asks for the number of players for the game)
NumberOfPlayersAnswer
loop
[while Number of players present < Total number for that game]
when player connects
LobbyMessage (Number of players present and usernames)
ACKMessage
StartingGameMessage
GraphicalGameInformationMessage (board, personal card, common cards, active player, bookshelf)

2 IN-GAME GENERIC

2.1 General vision of how a game works split between Model, View, Controller

- **Create Game**: Controller send a message to the Model in order to create a new game. startGame() in ServerManager class, saves the game in a file and sends the graphical info to the clients

- **Game information**: Once the game is created Model sends all the information to the Controller. Contains board, common cards, shelves, personal card and active player

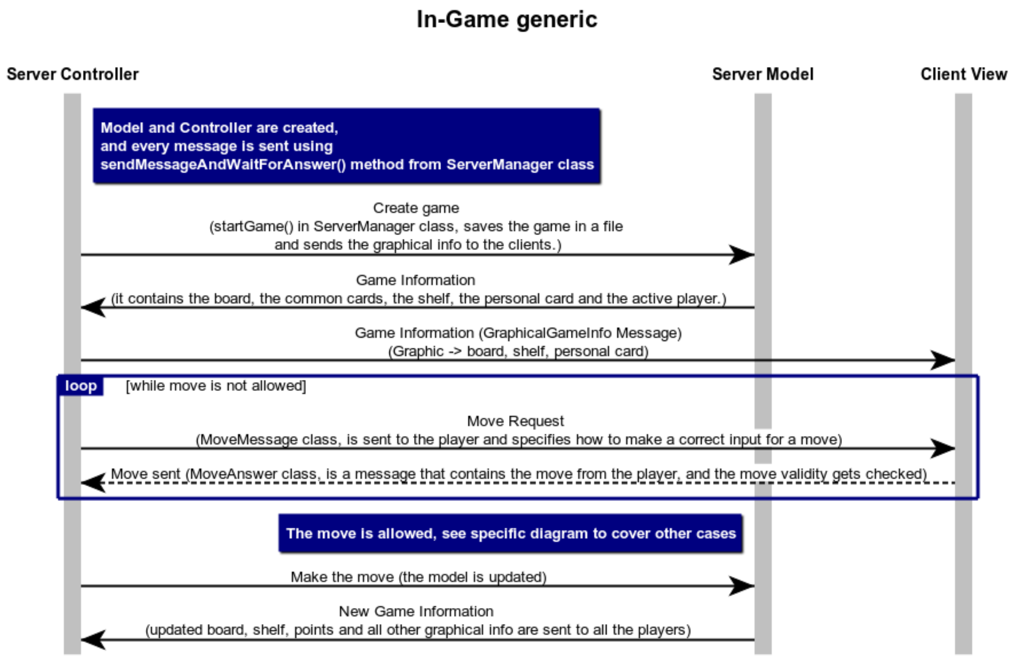
- **Game information**: (GraphicalGameInfo Message)(Graphic -> board, shelf, personal card).

- **(LOOP) Move request**: MoveMessage class, is sent to the player and specifies how to make a correct input for a move.

- **(LOOP) Move Sent**: MoveAnswer class, is a message that contains the move from the player, and the move validity gets checked

- **Make the move**: Once the move is approved by the Controller, a message is sent to the Model, and the move gets completed (Check the next paragraph for a more detailed view).

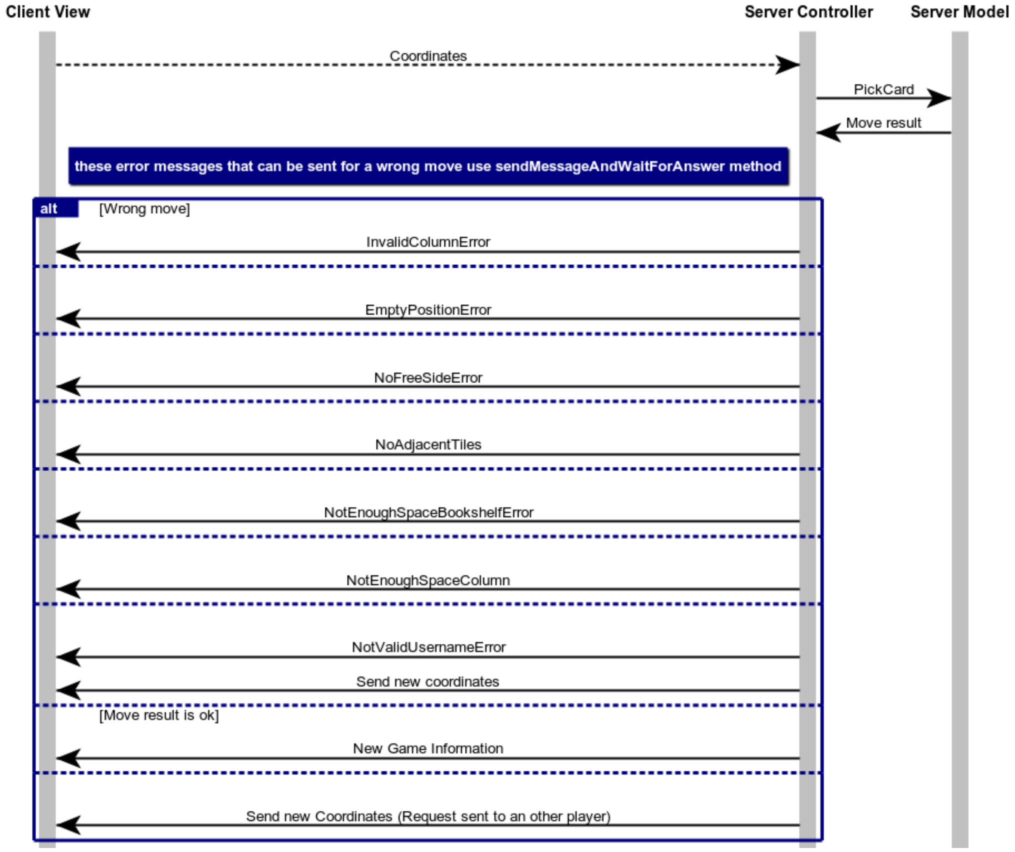
- **New Game Information**: Model sends to the Controller all updated information, and everything gets repeated, based on the latest situation.


In-Game generic
Server Controller
Server Model
Client View
Model and Controller are created,
and every message is sent using
sendMessageAndWaitForAnswer() method from ServerManager class
Create game
(startGame() in ServerManager class, saves the game in a file
and sends the graphical info to the clients.)
Game Information
(it contains the board, the common cards, the shelf, the personal card and the active player.)
Game Information (GraphicalGameInfo Message)
(Graphic -> board, shelf, personal card)
loop
[while move is not allowed]
Move Request
(MoveMessage class, is sent to the player and specifies how to make a correct input for a move)
Move sent (MoveAnswer class, is a message that contains the move from the player, and the move validity gets checked)
The move is allowed, see specific diagram to cover other cases
Make the move (the model is updated)
New Game Information
(updated board, shelf, points and all other graphical info are sent to all the players)

2.2 Move

- **Coordinates**: input of the move
- **PickCard**: The controller sends the coordinates to the model and asks it to remove the tiles from the board and put them in the bookshelf of the player who sent them, it also checks if the player's column in the bookshelf has enough space to put them.
- **Move result:** The model sends the result of the move to the controller. It will be positive if the coordinates of the tiles don't break the rules of the game, negative otherwise.
- **(Wrong move) InvalidColumnError** :
- **(Wrong move) EmptyPositionError :**
- **(Wrong move) NoFreeSideError** :
- **(Wrong move) NoAdjacentTiles** :
- **(Wrong move) NotEnoughSpaceBookshelfError**
- **(Wrong move) NotEnoughSpaceColumn** :
- **(Wrong move) NotValidUsernameError** :
- **(Wrong move) Send new coordinates** :
- **(Move Result OK)New Game Information** :
- **Send new coordinates**: This request is sent to another player.

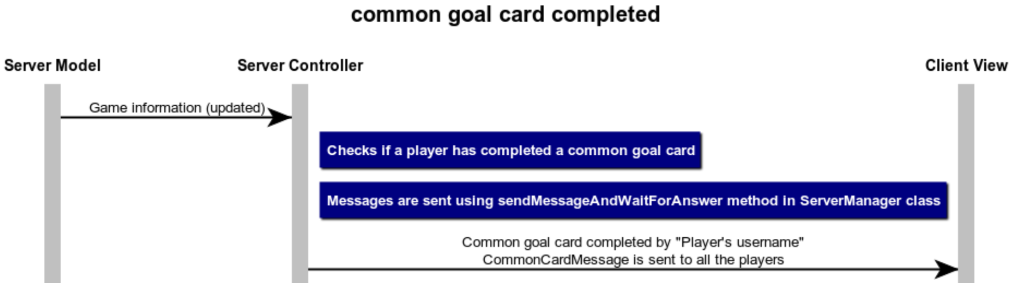
Move



these error messages that can be sent for a wrong move use sendMessageAndWaitForAnswer method

2.3 Common Goal Card Completed

- **Game Information updated**: Once a move in completed, Model sends the updated information of the game to the Controller.
- **Checks if a player has completed a common card goal, and gives points for it**
- **Common goal card completed**: CommonCardMessage is sent to all the players.


common goal card completed
Server Model
Server Controller
Client View
Game information (updated)
Checks if a player has completed a common goal card
Messages are sent using sendMessageAndWaitForAnswer method in ServerManager class
Common goal card completed by "Player's username"
CommonCardMessage is sent to all the players

NB: The MODEL uses the method checkbookshelf to see if a card has been completed, awards the points and then sends the result in a message. The CONTROLLER will only check a BOOLEAN value and notify the VIEW.