A player selects some object cards and put them in his library

SERVER SIDE

The Model modifies the game data when a player makes a move. Each move is received by the GameController, which starts a TurnController thread to handle the turn. TurnController is the class that interacts with the Game model. When the TurnController.run() method is called, it means a move has been received from the client. First, the method checks if the message is from the correct player; otherwise, the message is discarded, and the thread terminates. If the message is from the correct player, its content is forwarded to the Model through the manageTurn() method, which can return an informative message about the move. The Model performs the following checks: - On the grid: extraction of up to three consecutive tiles with an empty border. - On the library: There must be space in the chosen column. The informative message returned from the Game contains information about the outcome of the move: NEGATIVE case: If the TurnController is notified that the move is not feasible, it invokes "sendMessage()" on the GameController to request a new move. TurnController then terminates. POSITIVE case: If the TurnController is notified that the move was successful, it extracts the additional information contained in the message. This information can include reaching common goals, completing a library, and updates to the central grid and player's library. The players are then notified of this information. Before the TurnController process ends, a message is sent to request the next player's move, and the variable for the current player is updated.

CLIENT SIDE

The client controller receives the "MY_MOVE_REQUEST" message and notifies the View to request an input for drawing cards (askMove). The View receives the message and only checks that the number of selected cards is between 1 and 3. If the move meets the condition, it is returned to the ClientController, which forwards it to the server (MessageMove). If the server rejects the move, the client receives a new "MY_MOVE_REQUEST" message, and the move needs to be sent again. On the other hand, if the move is accepted, the client receives the updated game data and simply prints it.

A player achieves a common goal

The process is entirely managed on the server side and simply transmitted to the clients in the game.

The procedure begins when a client makes a move, the server, through the TurnController class, carries out the requested move by the user on the model.

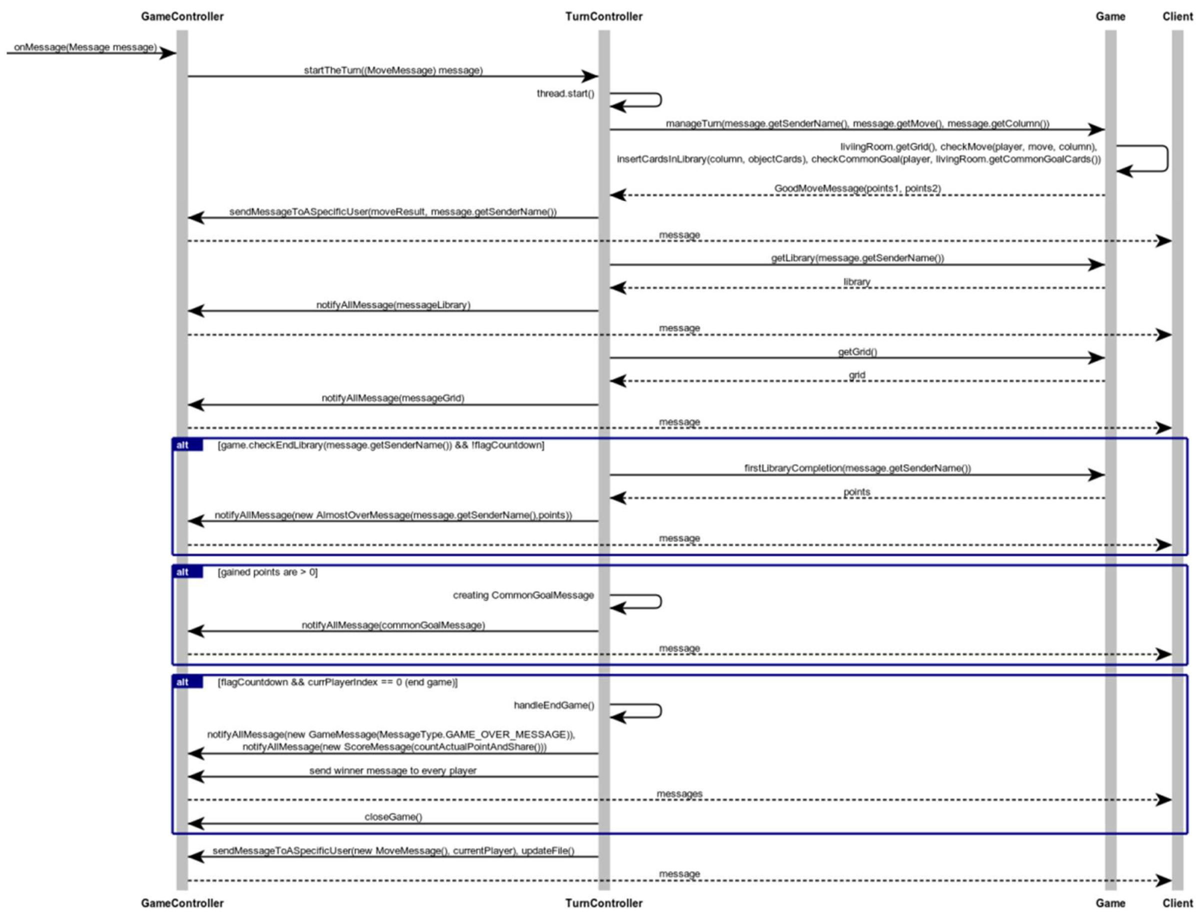
The model detects any satisfied common goals during the move checks (checkCommonGoal), updating the scores on the model side as well as the central grid and the player's library.

The model also composes a message containing, among other information, the number of points gained from the common cards with the just performed move.

This message is returned to the TurnController, which checks any points collected in the message and forwards them to the clients with a CommonGoal message.

The clients will then display these new points on the screen.

A turn management, whit the achievement of a common goal


GameController
TurnController
Game
Client
onMessage(Message message)
startTheTurn((MoveMessage) message)
thread.start()
manageTurn(message.getSenderName(), message.getMove(), message.getColumn())
livingRoom.getGrid(), checkMove(player, move, column),
insertCardsInLibrary(column, objectCards), checkCommonGoal(player, livingRoom.getCommonGoalCards())
GoodMoveMessage(points1, points2)
sendMessageToASpecificUser(moveResult, message.getSenderName())
message
getLibrary(message.getSenderName())
library
notifyAllMessage(messageLibrary)
message
getGrid()
grid
notifyAllMessage(messageGrid)
message
alt
[game.checkEndLibrary(message.getSenderName()) && !flagCountdown]
firstLibraryCompletion(message.getSenderName())
points
notifyAllMessage(new AlmostOverMessage(message.getSenderName(),points))
message
alt
[gained points are > 0]
creating CommonGoalMessage
notifyAllMessage(commonGoalMessage)
message
alt
[flagCountdown && currPlayerIndex == 0 (end game)]
handleEndGame()
notifyAllMessage(new GameMessage(MessageType.GAME_OVER_MESSAGE)),
notifyAllMessage(new ScoreMessage(countActualPointAndShare()))
send winner message to every player
messages
closeGame()
sendMessageToASpecificUser(new MoveMessage(), currentPlayer), updateFile()
message
GameController
TurnController
Game
Client