Eryantis Protocol Documentation

Moretti Lorenzo, Mormile Matteo, Orofino Antonino
Group 6

June 24, 2022

Contents

1 Messages **2**

1.1 NotifyPlayerIdMessage 2
1.2 NewGameMessage 2
1.3 AskNewGameName 2
1.4 AskGameListMessage 3
1.5 GameListMessage 3
1.6 SelectGameMessage 3
1.7 AskNewGameChoice 3
1.8 AskNickname 4
1.9 SetNickname 4
1.10 AskTowerColor 4
1.11 SetTowerColor 5
1.12 AckTowerColor 5
1.13 GameStart 5
1.14 AskAssistantCard 6
1.15 SetAssistantCard 6
1.16 UpdateGameBoard 6
1.17 AskAction 6
1.18 SetAction 7
1.19 ShowLastRound 7
1.20 ShowEndGame 7
1.21 ShowDisconnection 7
1.22 Heartbeat 8

2 Scenarios **9**

2.1 Game setup 9
2.2 Planning phase 10
2.3 Action phase 11

1 Messages

The messages are classified depending on the recipients and the sender. The possible kinds are the following:

- `CVMessage`: sent by the game controller to the view;
- `CVMessage`: sent by the view to the game controller;
- `SYSMessage`: sent by the view to the server controller.

1.1 NotifyPlayerIdMessage

This message is sent from the server to the client (`CVMessage`) after accepting its connection request. It only sends the id assigned to it by the server

Arguments

1. `playerID`: player ID generated by the server.

Possible responses

- `AskGameListMessage`:requests the list of available games from the server;
- `NewGameMessage`: message containing the new game settings.

1.2 NewGameMessage

This message, which belongs to the `SYSMessages` class, is sent by the player to provide the new game configuration.

Arguments

1. `playerID`: the ID of the message's sender;
2. `gameInfo`: cointains the new game name, the number of players (2..3) and the game mode (Basic, Expert).

Possible responses

- `AskNewGameName`: ask the player to provide a new name of the game to create;
- `AskNickname`: ask the player to provide a nickname.

1.3 AskNewGameName

This message is sent from the server to the client (`CVMessage`) to ask the player to provide a new name of the game to create.

Arguments

This message has no arguments.

Possible responses

- `AskNewGameName`: ask the player to provide a new name of the game to create;
- `AskNickname`: ask the player to provide a nickname.

1.4 AskGameListMessage

This message, which belongs to the `SYSMessages` class, is sent by the client to ask for the list of available games.

Arguments

1. `playerID`: the ID of the message's sender;

Possible responses

- `GameListMessage`: returns the list of available games.

1.5 GameListMessage

This message is sent from the server to the client (`CVMessage`) to show the list of available games.

Arguments

1. `gameList`: the list of available games;

Possible responses

- `SelectGameMessage`: ask the player to provide the name of the game to join.

1.6 SelectGameMessage

`SYSMessage` sent to the server with the chosen game's name.

Arguments

1. `playerID`: the ID of the message's sender;
2. `gameName`: the name of the game you want to join.

Possible responses

- `AskNewGameChoice`: ask the player to provide a new name of the game to join;
- `AskNickname`: ask the player to provide a nickname.

1.7 AskNewGameChoice

This message is sent from the server to the client (`CVMessage`) to ask the player to provide a new name of the game to join

Arguments

This message has no arguments.

Possible responses

- `AskNewGameChoice`: ask the player to provide a new name of the game to join;
- `AskNickname`: ask the player to provide a nickname.

1.8 AskNickname

This message is sent from the server to the client (`CVMessage`) to ask the player to provide a nickname.

Arguments

1. `isFirstRequest`: true if it is the first request.

Possible responses

- `SetNickname`: message sent to set the player nickname.

1.9 SetNickname

`VCMessage` sent to the server with the chosen nickname.

Arguments

1. `playerID`: the ID of the message's sender;
2. `nickname`: the nickname chosen by the player.

Possible responses

- `AskTowerColor`: condition in which the nickname provide is available and it was assigned to the player;
- `AskNickname`: the message has been received but the nickname was unavailable and another choice should be sent.

1.10 AskTowerColor

This message is sent from the server to the client (`CVMessage`) to ask the player to choose the color of the towers.

Arguments

1. `possibleColors`: the available colors;
2. `isFirstRequest`: true if it is the first request.

Possible responses

- `SetTowerColor`: message sent to set the color.

1.11 SetTowerColor

`VCMessage` sent to the server with the chosen color.

Arguments

1. `playerID`: the ID of the message's sender;
2. `color`: the color chosen by the player.

Possible responses

- `AckTowerColor`: condition in which the color provide is available and it was assigned to the player;
- `AskTowerColor`: the message has been received but the color was unavailable and another choice should be sent.

1.12 AckTowerColor

This message is sent from the server to the client when the color chosen by the player is valid. It is a `CVMessage`

Arguments

This message has no arguments.

Possible responses

This message has no responses.

1.13 GameStart

This message is a `CVMessage` and it is sent to every client to inform that the game is ready to start after two or three players have been matched. Moreover it specifies the first player chosen randomly.

Arguments

1. `game`: the current state of the game board;
2. `firstPlayer`: the nickname of the first player.

Possible responses

This message has no responses.

1.14 AskAssistantCard

This message is sent from the server to the client (`CVMessage`) to ask the player to choose an assistant card.

Arguments

1. `game`: the current state of the game board;
2. `possibleCards`: list of possible cards that can be chosen by the player.

Possible responses

- `SetAssistantCard`: message sent to set the chosen card.

1.15 SetAssistantCard

`VCMessage` sent to the server to set the assistant card.

Arguments

1. `nickname`: the nickname of the message's sender;
2. `assistantCard`: the card chosen by the player.

Possible responses

This message has no responses.

1.16 UpdateGameBoard

This message is sent from the server (`CVMessage`) to the client when the state of the game's board has been updated.

Arguments

1. `gameInfo`: current state of the game board.

Possible responses

This message has no responses.

1.17 AskAction

`CVMessage` sent to the current player to ask to perform an action.

Arguments

1. `possibleActions`: list of possible cards that can be chosen by the player;
2. `isFirstRequest`: true if it is the first request.

Possible responses

- `SetAction`: the message has been received and an action is sent.

1.18 SetAction

This message is sent from the client to the server (`VCMessage`) when an action is performed by the player.

Arguments

1. `nickname`: the nickname of the message's sender;
2. `action`: move of the player.

Possible responses

- `AskAction`: the message was received but the action was impossible or there are still moves available.

1.19 ShowLastRound

This message is sent from the server to each client (`CVMessage`) when either a player ends his cards or the students' bag is empty. The game will finish at the end of the current round.

Arguments

This message has no arguments.

Possible responses

This message has no responses.

1.20 ShowEndGame

`CVMessage` sent to each client to report the end of the game and the nickname of the winner if there is one.

Arguments

1. `winnerNickname`: nickname of the winner. It can be null if no winner could be determined.

Possible responses

This message has no responses.

1.21 ShowDisconnection

This message is sent from the server (`CVMessage`) to every client, but the disconnected one, to report that the game must end due to the disconnection of a player.

Arguments

1. `disconnectedNickname`: nickname of the player who disconnected from the match.

Possible responses

This message has no responses.

1.22 Heartbeat

This message is a `SYSMessage` and it is sent to keep alive the connection between two sockets.

Arguments

This message has no arguments.

Possible responses

This message has no responses.

2 Scenarios

2.1 Game setup

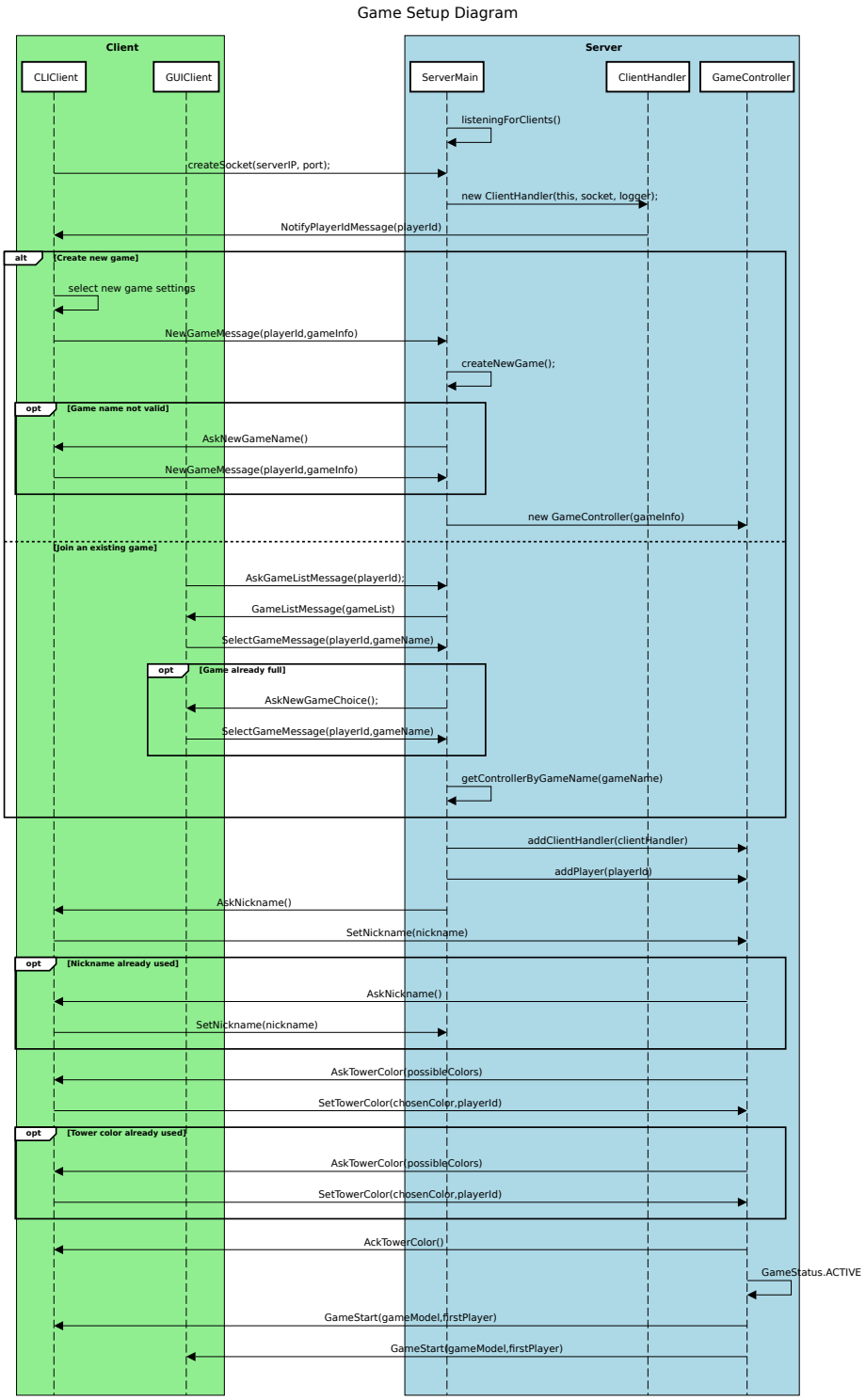

Figure 1: Game setup diagram

2.2 Planning phase

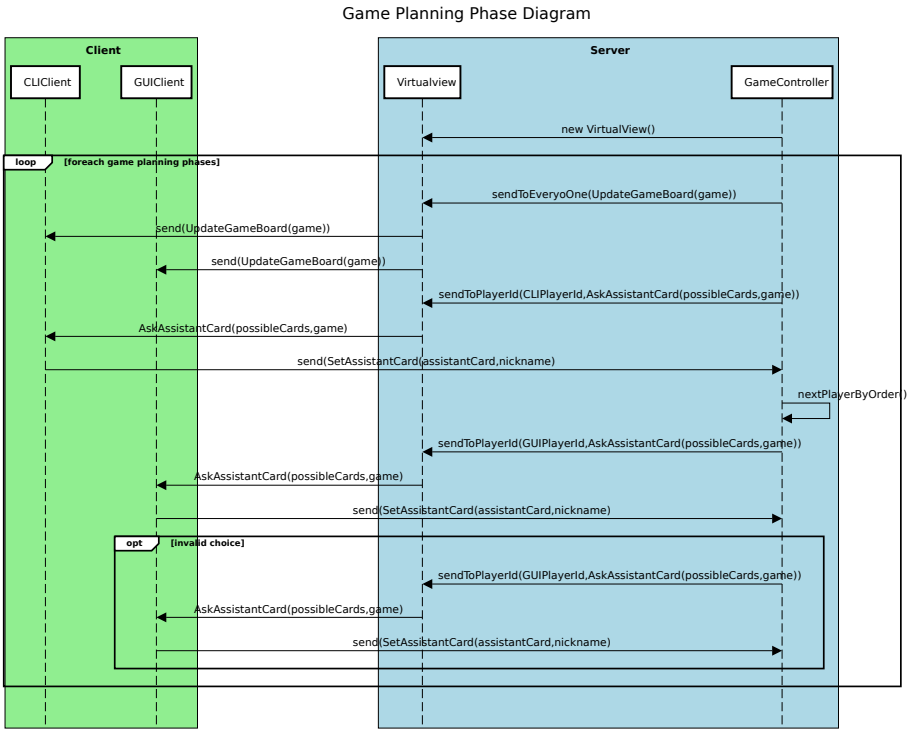

Figure 2: Planning phase diagram

2.3 Action phase

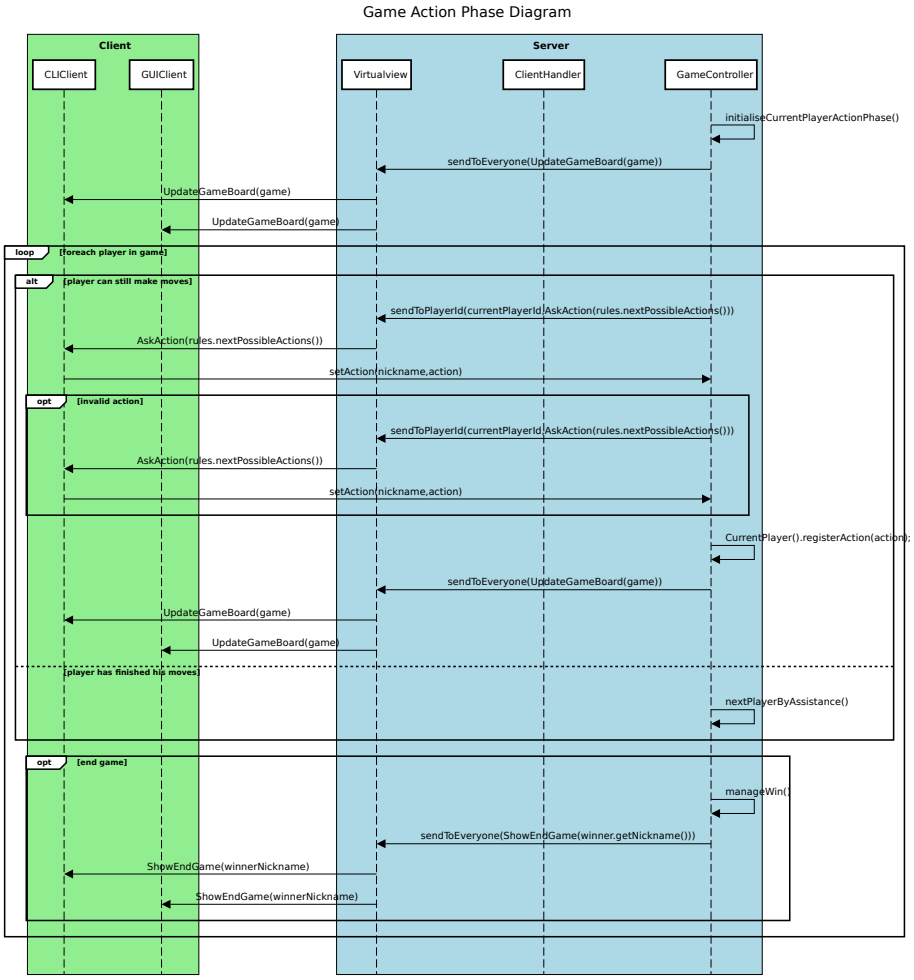

Figure 3: Action phase diagram