Communication protocol

group AM 35

14/05/2019

We chose a 'fat model' approach, with a very skinny view. Therefore, the model is the only one that knows the logic of the game and determines what each player can do in a specific moment. The set of the selectable items (squares, players, commands, actions, ...) is sent to the client as lists of strings. Once the login is performed (which is done on client initiative), the only interaction of the client is the selection and forwarding of one of these items (if any is available). The controller takes in charge the selection and makes the model update itself, then the new status and new selectables are sent to each player. Apart from selecting items, a client can choose to disconnect: in this case it sends a request to the server.

Once the match is started, players of the match are set permanently and remain until the end. When a player disconnects he becomes INACTIVE (can't do anything but he stays in the board and he can receive damages). Connection loss, player choosing quitting the game and timeouts are all treated in this way.

When the client connects to the server, the server starts waiting for client's Events. The client sends a loginEvent which contains the chosen username and then starts waiting for server's Messages. Server and client listen to each other for the rest of the connection.

1 Messages and events

Information travel in form of Messages and Events, both implementing Visitable interface. Depending on the type of connection chosen (socket or rmi), Messages and Events are serialized (and deserialized) as JSON strings or travel as objects. Messages travel from server to client, while Events travel from client to server.

1.1 Type of messages

The server can send different types of messages:

LoginMessage : states whether the login completed successfully or not

ConnectionUpdateMessage : contains a map with all the players and their connection status

GameOverMessage : states the end of the game and contains points and rank of each player

StartMatchUpdateMessage : states the beginning of a match. It also includes name and color of each player and the layout chosen by the server.

update messages: they contain some information about the state of the game. Their content is different depending on which player they are sent to.

CurrentPlayerUpdateMessage : contains the current player

DamageUpdateMessage : contains damages, marks, frenzy status and skulls of a certain player

KillShotTrackUpdateMessage : contains skulls, killings, global frenzy status of the match and points of the receiving player

LayoutUpdateMessage : contains weapons and ammo tiles on the board

PaymentUpdateMessage : contains info about payment (pending ammos and paid ammos) of the receiving player

PlayerUpdateMessage : contains position, ammos and state of a certain player

PowerUpUpdateMessage : contains the number of powerups of a certain player. If the player coincides with the receipient, it also includes a list with powerups' ID.

WeaponUpdateMessage : contains the unloaded weapons of a certain player. If the player coincides with the receipient, it also includes the loaded weapons.

SelectablesUpdateMessage : contains the list of items selectable by the receipient. They are sent as string (unique identifier)

1.2 Type of Events

The client can send 10 types of events:

LoginEvent : contains the chosen nickname

ActionSelectedEvent : contains the selected Action (as index of the selectable list)

ColorSelectedEvent : contains the selected color (as index of the selectable list)

CommandSelectedEvent : contains the selected command (as index of the selectable list)

ModeSelectedEvent : contains the selected mode (as index of the selectable list)

PlayerSelectedEvent : contains the selected player (as index of the selectable list)

PowerUpSelectedEvent : contains the selected power up (as index of the selectable list)

SquareSelectedEvent : contains the selected square (as index of the selectable list)

WeaponSelectedEvent : contains the selected weapon (as index of the selectable list)

2 Communication scenarios

The whole comminication can be described through 5 scenarios:

2.1 Login/reconnection

Login and reconnection are in fact the same event from the client. They are treated differently according to the state of the match. The client tries to login until it receives a positive loginMessage.

```
+-----------+     +------+                              +------+
|each client|     |client|                              |server|
+-----------+     +------+                              +------+
      |               |                                      |
      |               |    LoginEvent(name)                  | // checks if game
      |               |------------------------------------->| // is on and nickname
      |               |                                      |
      |               |                                      |
      |               |                                      |
      |               |          (on ok)                     |
      |               |<-------------------------------------|
      |               |       loginMessage(true, name)       |
      |               |                                      |
      |     connectionUpdateMessage(Map<name, state>)        |
      |<-----------------------------------------------------|
      |               |                                      |
      |               |                                      |
      |               |                                      |
      |               |    (on game full or alreadyLogged)   |
      |               |<-------------------------------------|
      |               |        loginMessage(false)           |
      |               |                                      |
      |               | (on name already taken or not found) |
      |               |<-------------------------------------|
      |               |       loginMessage(false)            |
      |               |                                      |
```

2.2 Game start (for each client)

```
// The server checks if there is a saved game with the same players.
// If yes it restores it, otherwise it starts a new match

+------+                              +------+
|client|                              |server|
+------+                              +------+
   |                                      |
   |     startMatchUpdateMessage          |
   |        (players, colors)             |
   |<-------------------------------------|
   |                                      |
   |         all update messages          |
   |<-------------------------------------|
   |<-------------------------------------|
   |<-------------------------------------|
   |                                      |
```

2.3 Player selects something

```
+------+                              +------+
|client|                              |server|
+------+                              +------+
   |      selectablesUpdateMessage        |
   |<-------------------------------------|
   |                                      |
   |     <item>SelectedEvent(index)       |
   |------------------------------------->|   // checks if selection is correct
   |                                      |
   |                                      |
   |               (on ok)                |  // updates GameModel.
   |<-------------------------------------|  // Checks if the action triggers
   |        some update messages          |  // the end of the game.
   |                                      |
   |                                      |
   |              (on error)              |
   |<-------------------------------------|
   |            genericMessage            |
   |<-------------------------------------|
   |         all update messages          |
   |                                      |
   |                                      |
```

2.4 Disconnection

A disconnection can always trigger the end of the game (if there are less than three players), see game over scenario. The disconnection can occur for three different reasons: the player disconnets on purpose, the connection fails or a timeout occurs. All this cases are threated equally: the corresponding observer is detached, the network-level connection is deleted and the client restarts from login phase (which includes choosing th etype of connection).

2.5 Game over

```
+------+                               +------+
|client|                               |server|
+------+                               +------+
   |                                       |
   |                                       |
   |   gameOverMessage(points, rank)       |
   |<--------------------------------------|
   |                                       |
```

3 An example: shooting

In this example the current player (beppe) starts with choosing an action. He will choose to shoot with "Machine gun" (basic effect + with focus shot) and he will damage Gianni and Enrico. The description of updateMessage is shortened: we write only the important fields.

```
+------+                               +------+
|client|                               |server|
+------+                               +------+
   |                                       |
   |<--------------------------------------|
   |    playerUpdateMessage {              |
   |      name: beppe,                     |
   |      state: CHOOSE_ACTION }           |
   |<--------------------------------------|
   |    selectablesUpdateMessage {         |
   |      selectableActions: [>, G, S] }   |
   |                                       |
   |-------------------------------------->|
   |          selectedActionEvent{2}       |  user chooses to shoot
   |                                       |
   |<-------------------------------------|
   |    playerUpdateMessage {             |
   |      state: SHOOT_WEAPON }           |
   |<-------------------------------------|
   |    selectablesUpdateMessage {        |  Select lock rifle
   |      selectableWeapons:              |  or machine gun
   |      "lock rifle", "machine gun" }   |
   |                                      |
   |------------------------------------->|
   |          selectedWeaponEvent{1}      |   user selects machine gun
   |                                      |
   |                                      |
   |                                      |
   |<-------------------------------------|
   |    playerUpdateMessage {             |
   |      state: CHOOSE_MODE }            |
   |<-------------------------------------|
   |    selectablesUpdateMessage {        |  select "basic effect"
   |      selectableMode [0]              |
   |                                      |
   |------------------------------------->|
   |            selectedModeEvent{0}      |  user selects it
   |                                      |
   |                                      |
   |                                      |
   |<-------------------------------------|
   |    playerUpdateMessage {             |
   |      state: CHOOSE_MODE }            |   select "with focus shot"
   |<-------------------------------------|     or press OK to skip
   |    selectablesUpdateMessage {        |
   |      selectableModes[1],             |
   |      selectableCommands[OK] }        |
   |                                      |
   |------------------------------------->|
   |            selectedModeEvent{0}      |  user selects "with focus shot"
   |                                      |
   |                                      |
   |                                     |
   |<------------------------------------|
   |    playerUpdateMessage {            |
   |      state: PAYING }                |
   |<------------------------------------|
   |    selectablesUpdateMessage {       |  do you wanna pay with ammos (OK)
   |      SelectablePowerups:[2,35]      |  or with a powerup?
   |      SelectableCommands:[OK] }      |
   |                                     |
   |------------------------------------>|
   |         selectedCommandEvent{0}     |     user chooses to pay
   |                                     |        with ammos
   |                                     |
   |                                     |
   |                                     |
   |<------------------------------------|
   |    playerUpdateMessage {            |
   |      state: CHOOSE_MODE }           |
   |<------------------------------------|
   |    selectablesUpdateMessage {       |  select "with turret mode"
   |      SelectableModes:[2]            |      or press OK to skip
   |      SelectableCommands:[OK] }      |
   |                                     |
   |------------------------------------>|
   |         selectedCommandEvent{0}     |  user confirms modes
   |                                     |
   |                                     |
   |                                     |
   |<------------------------------------|
   |    playerUpdateMessage {            |
   |      state: SHOOT_TARGET }          |
   |<------------------------------------|
   |    selectablesUpdateMessage {       |  select the target:
   |      SelectablePlayers:             |    gianni o enrico?
   |         [gianni, enrico] }          |
   |                                     |
   |------------------------------------>|
   |          selectedPlayerEvent{1}     |    user selects enrico
   |                                     |
   |                                     |
   |                                     |
   |<------------------------------------|
   |    playerUpdateMessage {            |
   |      state: SHOOT_TARGET }          |
   |<------------------------------------|
   |    selectablesUpdateMessage {       |  do you want to damage
   |      SelectablePlayers: [gianni] }  |    gianni, too?
   |                                     |
   |----------------------------------->|
   |          selectedPlayerEvent{0}     |    user selects gianni
   |                                     |
   |                                     |
   |                                     |
   |<------------------------------------|
   |    playerUpdateMessage {            |
   |      state: SHOOT_TARGET }          |
   |<------------------------------------|
   |    selectablesUpdateMessage {       |  who do you want to damage twice
   |      SelectablePlayers:             |    gianni or enrico?
   |         [gianni, enrico] }          |
   |                                     |
   |----------------------------------->|
   |          selectedPlayerEvent{0}     |    user selects gianni
   |                                     |
   |                                     |
   |                                     |
            //   end of action
```