Omi Online

(CO324 – Project II)

E/12/063 DILSHAN, M.E.L.

E/12/302SAMARANAYAKA, A.K.L.L.

7/5/2016

1. Application Messages

Responses from the server

The web front-end is continuously updated using Server Sent Event. Server keeps on sending JSON objects, encapsulated in HTTP response messages. These JSON strings contain the following fields.

cards:
an array of card objects (Each card object should have an image field with the file name of the card) representing the cards in the player's hand.

card1:
String showing filename of the card played by the 1st player.

card2:
String showing filename of the card played by the 2nd player.

card3:
String showing filename of the card played by the 3rd player.

mycard:
String showing filename of the card played by current player.

showHand:
Boolean value stating whether the GUI should show cards in the players hand.

showCards:
Boolean value stating whether the GUI should show the played cards. (card1, card2, card3, & mycard)

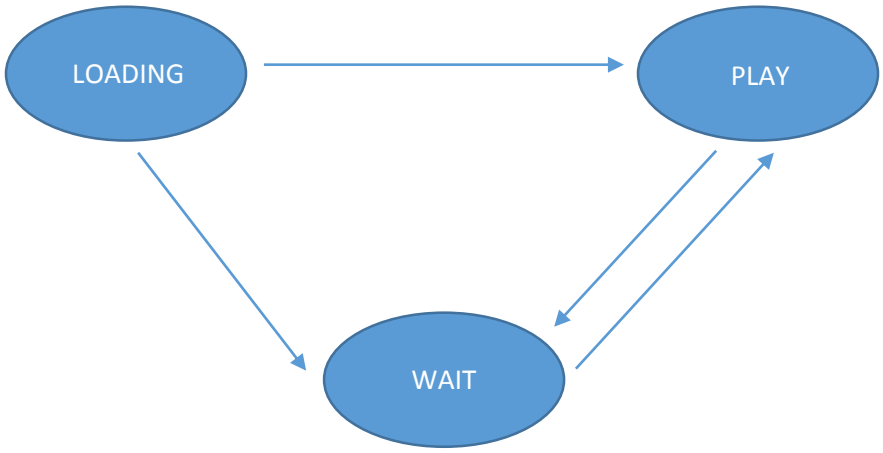
message:
The status message that should be shown to each player.

trump:
image of the trump card.

POST requests from the web front-end

card=<name of the card>

2. State Diagram of Game Play



LOADING State:

At This state the players are about to connect and after 4 players are connected.

PLAY State:

At this state players are playing their cards and states at the sever and each player is updated continuously.

WAITING State:

Waiting state until a player plays a card.

3. Design

Back-end a Java servlet that directs the game play of the JavaScript front-end that is provided. The server is tracking the state of each player's hand to ensure they play by the rules stated above. To do this State.java is used

The server waits until four clients have joined the game by visiting the Start page. Then deals each player 13 cards and declares the trump suite.

Each client is then given a turn to play a card, which is communicated to the other players. Once everyone has played a card, the server informs clients who won the trick. At the end of a round (13 tricks) the server updates player's scores. If no player has reached ten points, the server deals again and starts a new round.

At the end winner is chooses based on points.

GameLogic.java handles the requests and all the game logics. It uses both doGet and doPost methods to communicate.

At the first sever is initialize to no players has any cards. Then deals and starts the game.

4. Implementation

The client code provided uses the *Server Sent Events* ("EventSource") to update player state. GameLogic.java is implemented to servlet for handling every requests and response from the front-end to back-end also the other way.

- Client and server application message structures.

Used a standard serialization messages (JSON messages). So Jason objects are passed over network.

- Game state machine which models a round of Omi.

.
The game logic is abstracted into a separate enumeration.

5. References

I. Head first servlets and jsp, 2nd edition by Bryan Basham, Kathy Sierra, and Bert Bates

II. http://w3school.org

III. http://www.webterrace.com/cards/whist.html

IV. http://viralpatel.net/blogs/html5-server-sent-events-java-servlets.html

V. http://www.oracle.com/technetwork/articles/java/json-1973242.html

VI. http://learn.jquery.com

VII. http://knockoutjs.com/documentation