Eric Versaw

Multiplayer Programming – 320

Black Jack Game – Protocol Design Document

I was chosen to create a game of Black Jack that was multiplayer. It needed to include a chat room with username capabilities. I ended up making this an http game with a javascript made server, so you play the game on a web browser.

Tools used:

* Sockets.io
* Express
* Bootstrap

Methods used:

* HTML
* CSS
* Javascript
* JQuery

Multiple messages are sent between the server and the clients. Basically how I set it up is that whenever a client makes a move it instantly sends variable data to the server and then the server distributes all the data back to all of the clients and if a new client joins midway through a game, they receive all the variables from the server on start up.

Messages are sent to the server each time a client connects to the server it says “Connected %s sockets connected”, and the %s includes the connections.length meaning how many connections for a list of variables to the server.

When a client disconnects from the server the server will display a message saying “Disconneceted %s sockets connected”, and will include the new number of clients.

Each time a client makes a move it sends all the variables and console.logs them so the host can see how the 18 variables change throughout the game.

Each time the server sends out data back to the clients, each client receives a console.log if they hit F12 all of them list the 18 variables like so: "Data retrieved from server "+ variableHERE

|  |  |
| --- | --- |
| On socket connection | console.log('Connected %s sockets connected', connections.length); |
| On socket disconnection | console.log('Disconneceted %s sockets connected', connections.length); |
| On server sending data to X number of clients | console.log(variable); x 18 variables |
| On client sending data to server | console.log("Data retrieved from server "+ variable); x 18 variables |

I splice out connections from the list on disconnection. Also the data such as console.logs and actual variables get tossed around by socket.emit() which sends data from server to clients or clients to server. On the other end of the emit the server/clients have a socket.on event listener to receive the emitted data from.