Student name: Dhruv Jobanputra

Student ID: 22704304

1. This game is still very basic and there are a few ways to handle many more clients. For instance, right now the game can support one active game for up to 4 players where all other players are spectators. If you imagine there being 100’s of connected players, at any given time, only a maximum of 4 players will ever be playing. It makes much more sense to have multiple games running instead of one. When clients join, they will be sent to a random “room”. Each room will have some players and one game running. When there are a few clients, the rooms will combine with each other so that there is always a game running if there are enough clients overall. With this method, one issue of scaling is solved as there can now be many games running independently of each other.
2. To deal with multiple identical messages arriving simultaneously, you could look at the time stamp of when a message arrives. If the time stamp of multiple messages is the same and they contain the same message, then this would be an indication they are the same message that has been received twice (or more). After this you can continue to only select the first one and ignore the rests
3. The key difference between designing a network project and other programs I have developed is the fact that the players are all connected with each other. Most projects either just have one “client” or there can be multiple clients that can share information, but the clients aren’t actually connected. An example of the latter is a website. You can have multiple people add to a website, adding something will get stored into some database which is then shown to all other people using the website. But the clients in this case are still independent of each other. With networks this is not the case. For instance, one client’s actions have effects on all other clients. For example, in this game, if one person disconnects, the other player’s games are affected since all the games are connected with one another. Another thing you can utilise because of this is making a turn-based game like we did. As the players are all connected with each other. Clients may have to wait for someone else before being able to send a message.
4. Right now, the scale of the game is very small. This is because it can only support 4 people playing the game at any one time. As discussed before, a room system would solve this as additional players that join can be sent to other rooms with other games to play. As a result of this, performance is also reduced. By having multiple rooms, you can limit the maximum number of people in a room. With the correct method, there can be 100’s of players in the one game, and as a result, sending messages will be very slow as the messages are sent to each client individually via a for loop. A way to solve this (other than rooms) could be to broadcast messages instead of sending one message to each client that is connected.
5. I have been thorough while managing disconnections and player joining in this project. The project description mentioned that if a player exits during the middle of the game then the game should be handle this. I have implemented this but while I was implementing this, I found that if a player exits before getting this turn (given he is playing), then the server seemed to crash as there was no player to eliminate. I have successfully been able to handle this. As the player has not had his turn, they should not be eliminated and instead the game should just continue as if that player was never there.

As with joining while a game was running. Again, the project did not specify when a game “starts”. The way I have implemented my code, I have made sure that if a new player joins the server while there is a game created; if that game had not yet started by the first player making his turn, then the new player will also be part of the game. Essentially what this means that a game will be created if there are enough clients. However, the game is only considered to have started after the first player makes their move. If a new player joins before this happens, a new game will be made taking into account all of the players currently connected to the socket.

1. For the marker – I have not been able implement idle time for clients (part of tier 4). However, all other parts have been tested thoroughly and are working.

Some cases:

* Disconnecting a spectator
* Disconnecting a player before they had their turn
* Disconnecting a player when it is their turn
* Having more than 4 players connected at any given time
* A player joining before the first move of a game has been made
* A player joining after another player has disconnected/been eliminated

1. I have also ignored any messages sent by a client when it is not their turn. It did not make sense for me to store the messages and send them when it was their turn. What if the client wanted to make a different move when it was their turn?