Networking Documentation

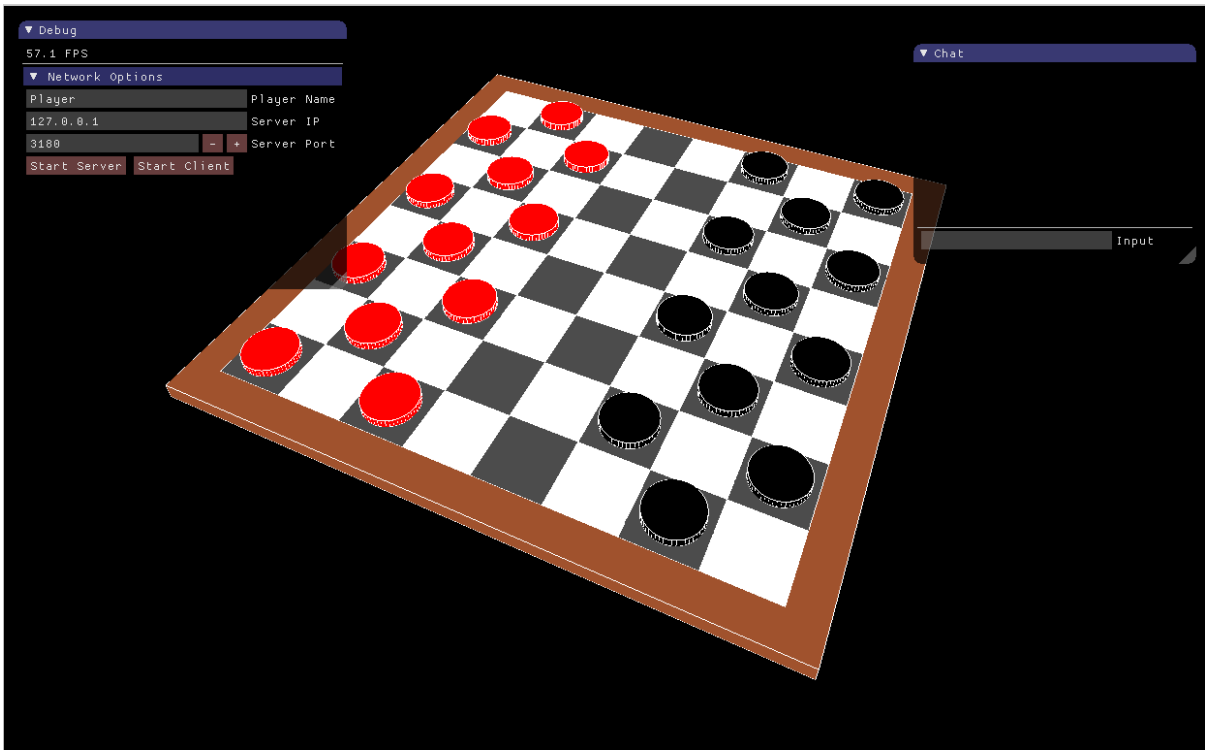
For my checkers game, I have chosen to demonstrate networking, rather than Artificial Intelligence. I chose networking, because it is such a vital part in so many games nowadays, and I'd never had any experience in it.

The checkers game itself is very simple. Everything is drawn with Gizmos, using simple AABB's (cubes) for the board, and cylinders for the pieces. I used ImGui for all user interactions outside of playing the game, as it is very clean and simple for the user.

When the player opens the game, they will be shown a checkers board, a console/chat window and another window that controls their networking. The network is of a Server-Client configuration, meaning that one user hosts the game as a server, and two join as clients/players. Here they can choose to host the game, or to join a server as a client. If they join as a client, they input their name and the IP address and port of the server. These have default values, defaulting to 127.0.0.1, the address of the computer you are on.

If the user chooses to join as a server, they will not be able to move the board. They can watch as turns are made, and talk to the players, but they can not do anything else. When a user joins as a client, the board will turn to the side that they are allocated to. Once two players have connected, the game can get underway.

As a client makes a move, they are heavily restricted as to what moves they can make, as it has to be a legal move. Once a move is made, it is sent to the server, it is checked, and sent back to each client as successful, then the board is updated. This should counter any form of cheating from the client's side.



There are 9 different messages that can be sent and received throughout the lifetime of this game of checkers. They are as follows:

ID_CLIENT_CONNECT:

Sent from client to server, requesting to connect to the game.

ID_CLIENT_TURN:

Sent from the server to the client, telling them when it is their turn.

ID_CLIENT_MESSAGE:

A generic message to be read in and displayed in the console. It is generally used for the chat window. It is sent from client to server, then relayed via the server's broadcast.

ID_SERVER_BROADCAST:

For generic messages from server to client. Anything sent with this message ID will go to all clients.

ID_SERVER_END:

Sent when the server is shutting down, from server to client.

ID_CLIENT_END:

Sent when a client is disconnecting from the server.

ID_CLIENT_LIST:

When a client joins, this message ID is broadcast to every client, sending a list of player names so that each client knows how many are connected.

ID_CLIENT_LIST_WIPE:

This is sent when a client first joins. It tells each client to wipe their current client list, so that it can be updated. As I write this, I realise that is redundant and could be done when receiving the "ID_CLIENT_LIST" message.

ID_ALLOCATE_SIDE:

Sent from server to client, as a client first joins. This tells the client which side/colour they will be playing as.