The ultimate chess battle

Iftimoaia Razvan

December 2021

Abstract

This document contains the main ideas which were used to build ”The ultimate chess battle”, as well as some pieces of code that has been written so far, possible future improvements and the technologies responsible for the connection to the network.

1 Introduction

”The ultimate chess battle” is an online multiplayer chess game which connects players all over the world. You only need to open the game, press ”Find a match” and you'll be queued up along with other players. In little to no time you will be paired with another user and start a good, quality chess game. So far the game is still in the implementation process leaving place for upgrades/functionalities that may not be mentioned in the documentation at this time.

2 Used technologies

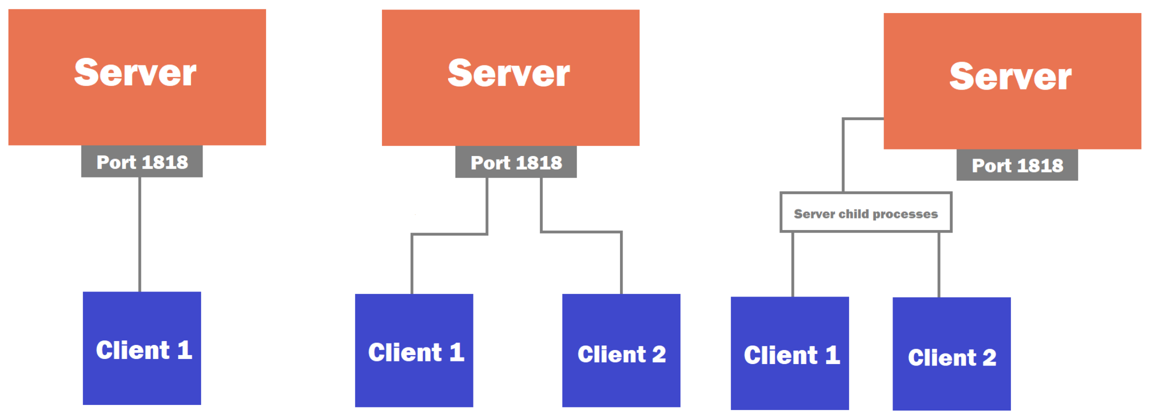
The game uses a TCP/IP protocol creating a safe connection between the clients and the server. The priority was sending the correct information over sending it fast. What do i mean by that is the Transmission Control Protocol is a connection-oriented protocol which establishes a connection between the client and the server before data can be sent. The server must always be listening for requests in order for the clients to connect. The three-way handshake, retransmission and error-detection are the characteristic safety measurements of this protocol which make it so reliable. Unfortunately the actions listed above come with the price of latency. It takes a considerable amount of time to ”wrap” the data, send it and for the receiver to ”unwrap” it, check it and send the confirmation back to the sender.

In short, the TCP/IP protocol checks whether the information was received correctly leaving no place for errors whereas the User Datagram Protocol (UDP) mainly focuses on speed. In a game like chess where every move matters, a simple transformation of a bit from 1 to 0 and vice versa will ruin the whole experience. The precision of the Transmission Control Protocol is needed to make sure that no errors occur during the game.

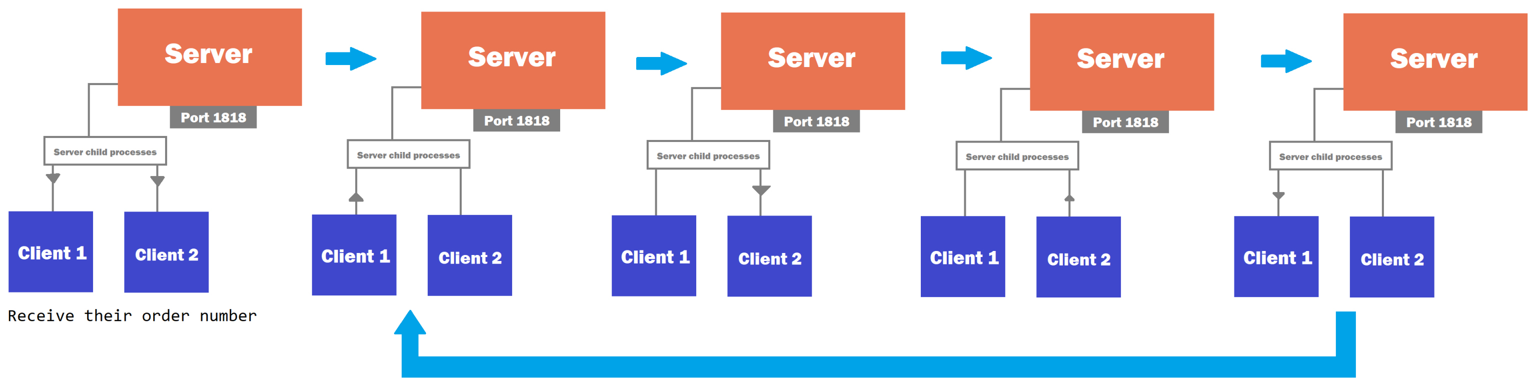
3 Application architecture

The way the application works is the following:

- The server is always open and listening to the clients that want to connect. If a connection request is made (a user presses ”Find game”) , the server accepts it and increments the number of players that are waiting in the lobby for a game with 1. As soon as there are at least two players, a child is borned and manages their game transforming the server into concurrent instead of iterative. This operation is represented graphically like this:



- After the child is put in charge of the game, it will send to each client their order number. This order number can be either 1 or 2. The player with the number 1 will have the white pieces and the player with number 2, the black ones. These numbers are chosen by the principle of "first come, first served".
- Now that the order has been established, the game may begin. The first player is free to make a move while the client of the second player is in a blocking state. The whole game is based on the transition between reading and writing states because the moves are predefined. The first player moves, writes the data to the server and then enters in a reading state. After the server's child receives the data, it sends it to the second client which is waiting. The second client updates the table with the new move and is free to write back to the server. The child's server receives it, sends it to the first client which is now eligible to make another move.



- The cycle mentioned above repeats until one of the two players wins or gives up.
- After the game is over, the child is killed and the connection between the server and the clients is closed.

4 Implementation details

The application does not require the users to authenticate and/or login before getting into the main menu. While in a match, each move will have 2 elements: the old cell respectively the new cell. With this information the table will update accordingly.

```
int oldCell;
int newCell;
```

The way the table is organised is using the following structure:

```
struct match {
  int x[65];
  int y[65];
  int pieceNumber[65];
  int state[65];
  int validMove[65];
}
```

X and Y store the coordinates for the visual part of the project. pieceNumber can either have -1 or a number between 1 and 12 where:

- -1: empty cell
- 1: white pawn
- 2: white rook
- 3: white bishop
- 4: white knight
- 5: white queen
- 6: white king
- 7: black pawn
- 8: black rook
- 9: black bishop
- 10: black knight
- 11: black queen
- 12: black king

State stores either 0 or 1 which means if a cell is selected or not. validMove is used to check on which cells can be placed the selected piece.

In the main menu is an ”exit” button and an ”options” button. The ”options” are not yet developed but can be upgraded with functions for volume, window size, etc.

5 Conclusions

For the moment the application is still in the making process. Besides what was already presented, other ideas may be implemented such as: background music as well as a volume button for it, different chess tables/pieces, player VS AI with different difficulties, the option of playing with a friend, a counter that shows how many players are online, another counter for the number of moves made by each player in a match and so on.

As for the graphical interface, Raylib was chosen for the most pleasing visual experience. The chess pieces and the table were drawn specially for this game.

6 Bibliography

Below i will list the links used for the documentation.

[1]Transmission Control Protocol https://en.wikipedia.org/wiki/Transmission$_{C}ontrol_{P}rotocol$
[2] Computer Networks course website https://profs.info.uaic.ro/ computernetworks/ProiecteNet2021.php
[3] Beej's Guide to Network Programming https://beej.us/guide/bgnet/html//index.htmlwhat-is-a-socket