Wentworth Institute of Technology

2D Chess

Project Design Document

Team: Indecisive

Tiffany Phan and Yuliya Smilyanova

[phant9@wit.edu](mailto:phant9@wit.edu) and [smilyanovay@wit.edu](mailto:smilyanovay@wit.edu)

GitHub Repo: <https://github.com/smilyanovayatwit/2D_Chess>

**Table of Contents**

1. Introduction……………………………………………………………………………………………………………………………..3

2. Application Features…………………………………………………………………………………………………………………3

3. Application Architecture…………………………………………………………………………………………………………..3

4. System Design………………………………………………………………………………………………………………………….3

5. Plan………………………………………………………………………………………………………………………………………….6

6. References……………………………………………………………………………………………………………………………….6

**1. Introduction**

2D Chess is a networking implementation of the popular 2-player board game, chess, using server-client architecture to allow for multiplayer gaming. Developing this application allows us to practice and play around, while mastering the concepts of socket programming and multithreading. It is also a great way to understand how the game of chess is played.

**2. Application Features**

* Allow client to join a game
* Allow client to make a move
* Allow client to leave the game
* Server broadcasts messages such as: client has joined, which client has won, client has left (these are part of other features explained in system design)

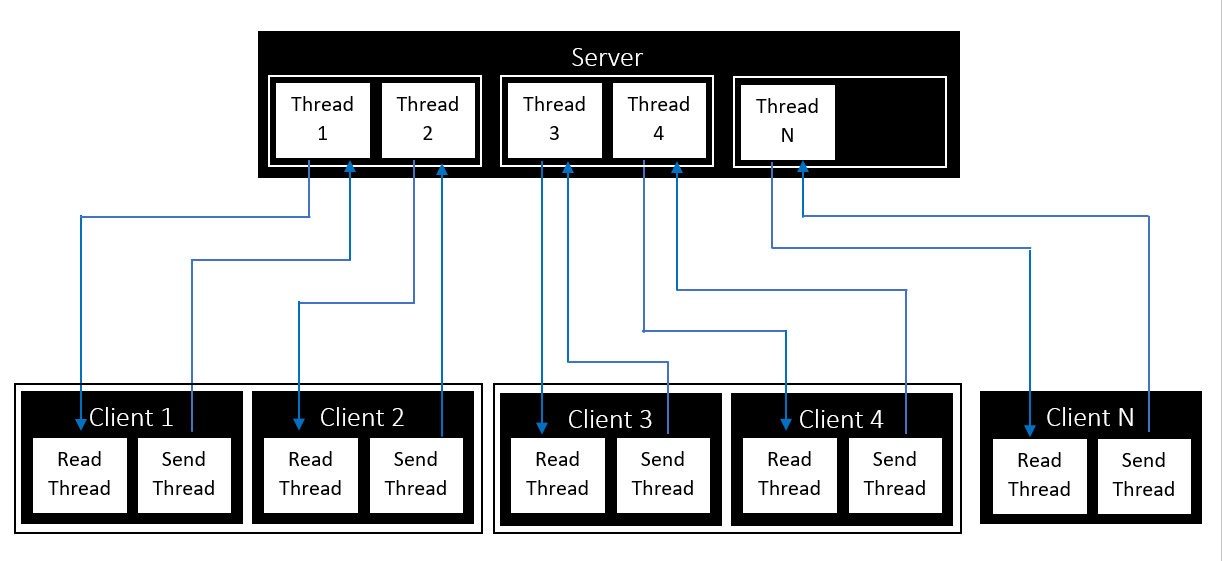
**3. Application Architecture**

This chess game application is server-client architecture applied to a regular chess game (multiplayer purposes).

Game: Board, game functions, all the pieces for the game

Server: Multithreaded, can handle multiple clients at the same time (but there can only be 2 clients per game), read from both the game and clients, write to the clients

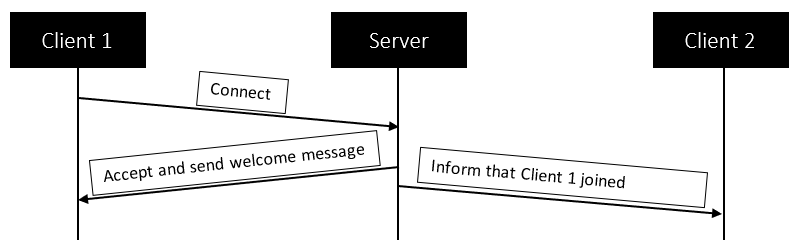
Clients: Multithreaded, can read from the server, write to the server, can play the game



**4. System Design**

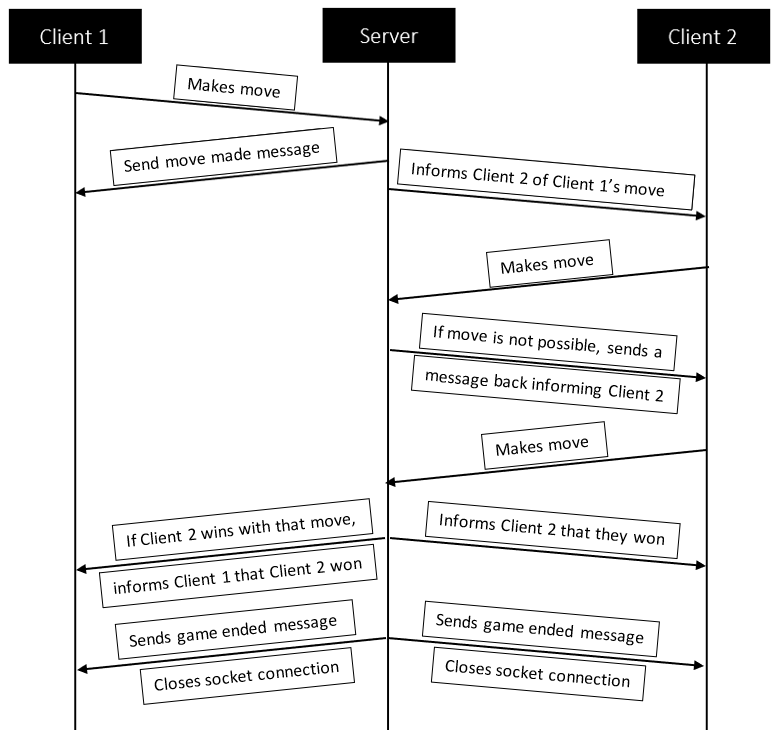
Allow client to join a game

* Client 1 connects to Server
* Server accepts connection and sends welcome message
* If there is a Client 2, Server notifies them of Client 1 joining the game



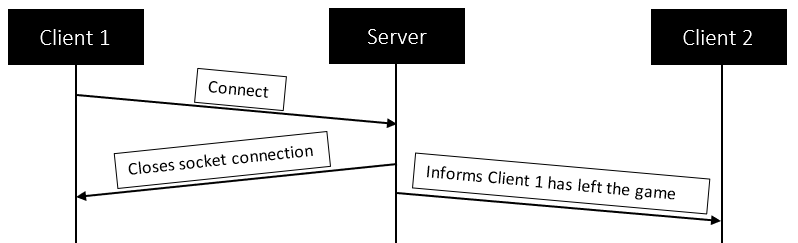
Allow client to make a move

* Client 1 makes a move; Server repeats move to Client 1 and notifies Client 2 of the move
* Client 2 makes a move; if the move is not possible, Server sends a message back to Client 2 saying that the move is not possible
* Client 2 makes another move; if Client 2 wins, Server notifies everyone that Client 2 won
* Server then sends a message saying game has ended, and all closes socket connections



Allow client to leave the game

* Client 1 sends {quit} message
* Server closes the connection with Client 1
* Server notifies Client 2 that Client 1 left
* Closing the window also follows the same steps



**5. Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Week | Feature | Task | Assign To | Done By |
| Week 9 | Project Design Document | * Complete the Project Design Document | Team | Week 10 |
| Week 10 | Chess Game | * Implement the chess game | Yuliya | Week 12 |
| Week 12 | Server-Client Interaction | * Implement the multithreaded server * Allow client to join a game * Allow client to leave the game | Tiffany | Week 13 |
| Week 12 | Server-Client Interaction | * Allow client to make a move * Research for interaction examples from other games | Yuliya | Week 13 |
| Week 13 | Final Presentation | * Powerpoint Presentation | Team | Week 14 |
| Week 14 | Final Project | * Final Project Design Document * Final GitHub Repo | Yuliya | Week 15 |
| Week 14 | Final Project | * Demo Video | Tiffany | Week 15 |

**6. References**

* <https://web.cs.wpi.edu/~imgd4000/d07/projects/proj3/proj3.pdf>
* <https://www.youtube.com/watch?v=yyhNBV_5ayM>
* <https://www.youtube.com/watch?v=Bj6N0pEVC-I>
* <https://massivetechinterview.blogspot.com/2015/07/design-chess-game-using-oo-principles.html>
* <https://medium.com/analytics-vidhya/school-java-project-chess-1-85f97a2d1877>
* <http://www.java2s.com/example/java/2d-graphics/draw-a-chess-board.html>
* <https://www.openprocessing.org/sketch/309165/>
* [https://www.geeksforgeeks.org/design-a-chess-game/](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.geeksforgeeks.org%2Fdesign-a-chess-game%2F&data=04%7C01%7Cphant9%40wit.edu%7C7876c08273e145d5fcfa08d88a5243b9%7C2af16cc576494528bc4d3d9b6f64c066%7C0%7C0%7C637411433235762309%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=syQlznFmJ3%2F6QyuWIPmpPHAlRyJsV1wIX%2F%2F832T6tDE%3D&reserved=0)
* [https://codemiles.com/finished-projects/java-chess-t618.html?mobile=on](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fcodemiles.com%2Ffinished-projects%2Fjava-chess-t618.html%3Fmobile%3Don&data=04%7C01%7Cphant9%40wit.edu%7C7876c08273e145d5fcfa08d88a5243b9%7C2af16cc576494528bc4d3d9b6f64c066%7C0%7C0%7C637411433235772264%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=yx%2FVyQnVpsM7ZyCR6Z3quJ1TavGQFOMXIhqMaCk5sTc%3D&reserved=0)
* [https://cs.lmu.edu/~ray/notes/javanetexamples/](https://nam04.safelinks.protection.outlook.com/?url=https:%2F%2Fcs.lmu.edu%2F~ray%2Fnotes%2Fjavanetexamples%2F&data=04%7C01%7Cphant9%40wit.edu%7C258f3c82a01a45d75ce408d897d77b22%7C2af16cc576494528bc4d3d9b6f64c066%7C0%7C0%7C637426299033403858%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C1000&sdata=ATW0T4LoGFHyw0bzhEndogPku%2FGOJ4hKkcAWz4aKhd4%3D&reserved=0)
* <http://freesourcecode.net/javaprojects/15421/Standard-chess-game-in-java-with-2-player#.X9f5-NhKhPY>