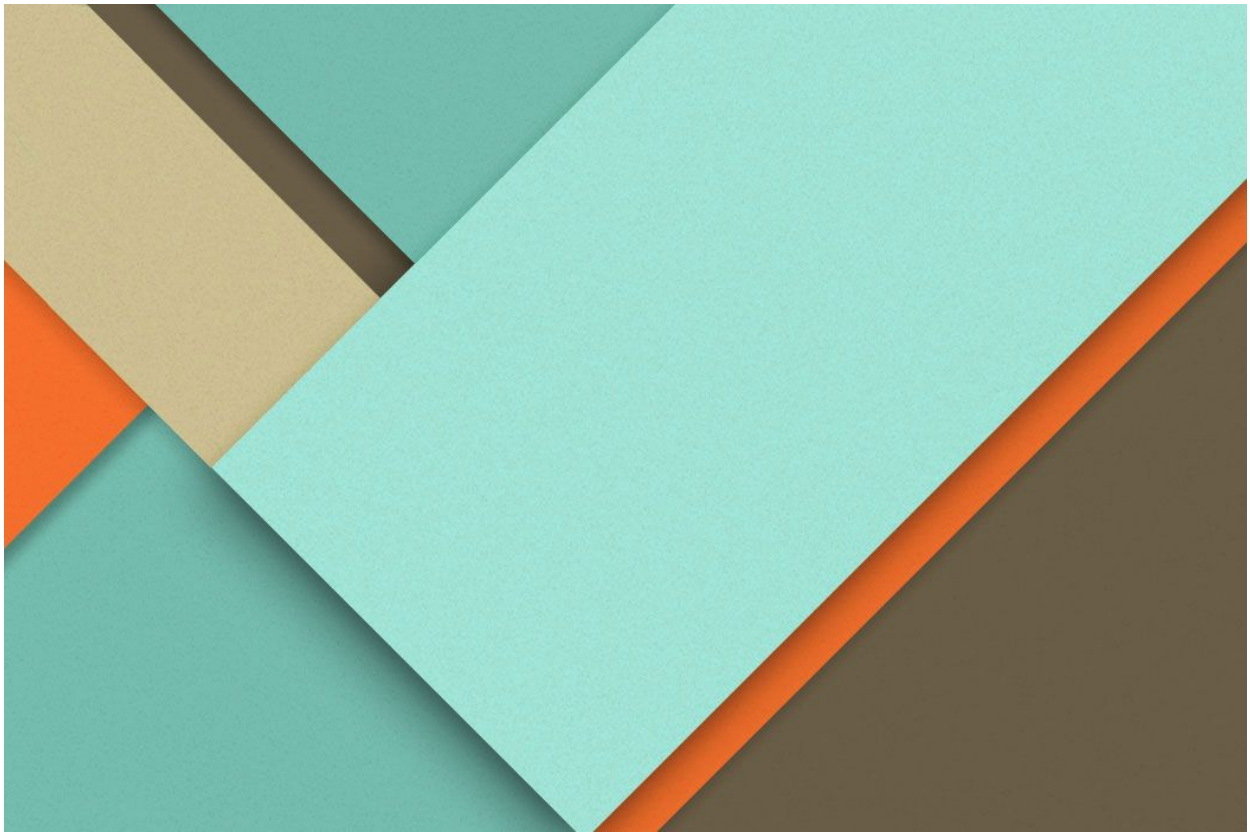


Project 5, Tic-Tac-Toe, Team 12

12.09.2019

Joshua Yuen
jyuen7@uic.edu

Eric Zavala
ezaval7@uic.edu

Miguel Alanis
malani20@uic.edu

General Description

Server:

In the server, we have two scenes. One that allows you to enter a port number to listen to. Once confirmed, we move to it's second scene which keeps track of all players that have been on the server, furthermore keeping track of the top 3 players and their scores as well.

When the server receives an object from the client, it will take the board and determine if the player has won, if not, it will use the MinMaxAlgorithm to decide the best/random/worst move depending on the difficulty that was sent from the client. After determining the move, it will send that back to the client. It will also check to see if the computer won, if so, it will update the client with an end game status, otherwise, the game will continue until either the player/computer wins or they draw.

Client:

In the client, we have four scenes. One that allows you to enter a port and ip-address. Once confirmed, the client will allow you to select a difficulty, after which it will direct you to the game. The player will always move first and it's piece will be "O." Once a move has been selected and confirmed it will be sent to the server in an object that contains the player's ID, the current gameboard, and the status of the game. After the server checks whether or not the game is finished, client will continue playing until there is a win or a draw. Upon receiving the status for a finished game, the client will move to a results scene that will show the finished board as well as the players with the top 3 scores. From here, you will be prompted with two buttons, one that will allow the player to play again to bring you to the difficulty selection screen, the other to quit the game.

MinMaxAlgorithm Implementation

No changes to original code.

Server creates a thread for the "FindNextMove" class that will that call AI_MinMax.

We return the moves-list and parse through the different option's weights and return the move best fit for the difficulty (easy, medium, expert.

Description of Each Teammate's Contribution

I. Joshua Yuen

UML Class Diagrams, MinMaxAlgorithm, Server/Client Communication, Initial Server/Client Setup, Server Logic, Server/Client Consumer Serializables, Win Condition Logic, Updated Diagrams, Test Cases

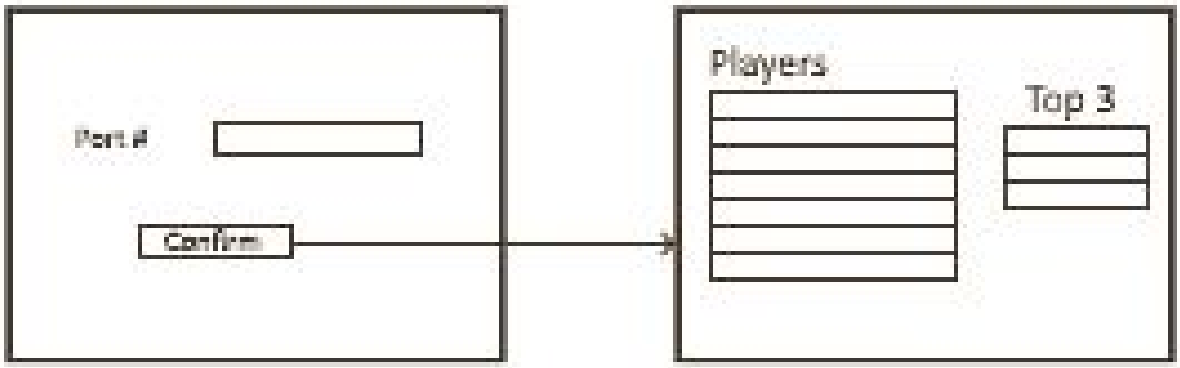
II. Eric Zavala

Activity Diagram, MinMaxAlgorithm, Server Logic, Leaderboard, Server/Client Consumer Serializables, Fixing Win Condition Logic, Revamp Win Condition Logic, FindNextMove Class & Threading, Cases

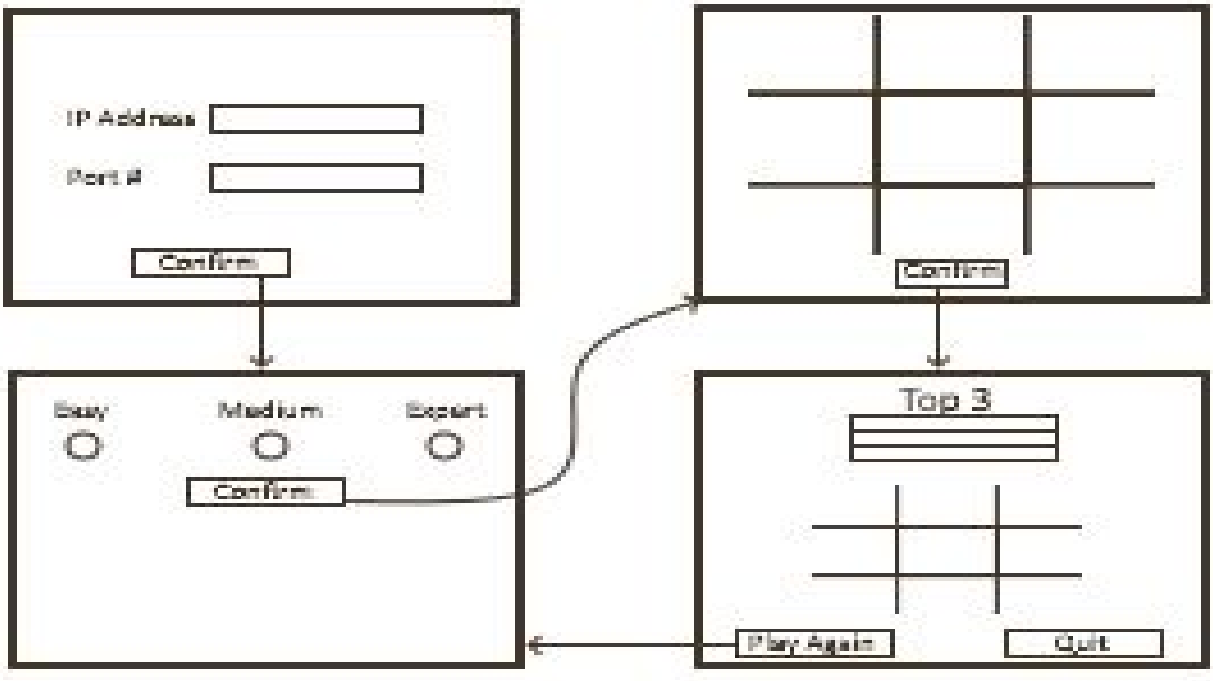
III. Miguel Alanis

Wireframe, Client Gui, Server Gui, Board Interaction, Board Logic, Client Logic, MinMaxAlgorithm, Server/Client Consumer Serializables, Server Leaderboard & Client ListView Manipulation

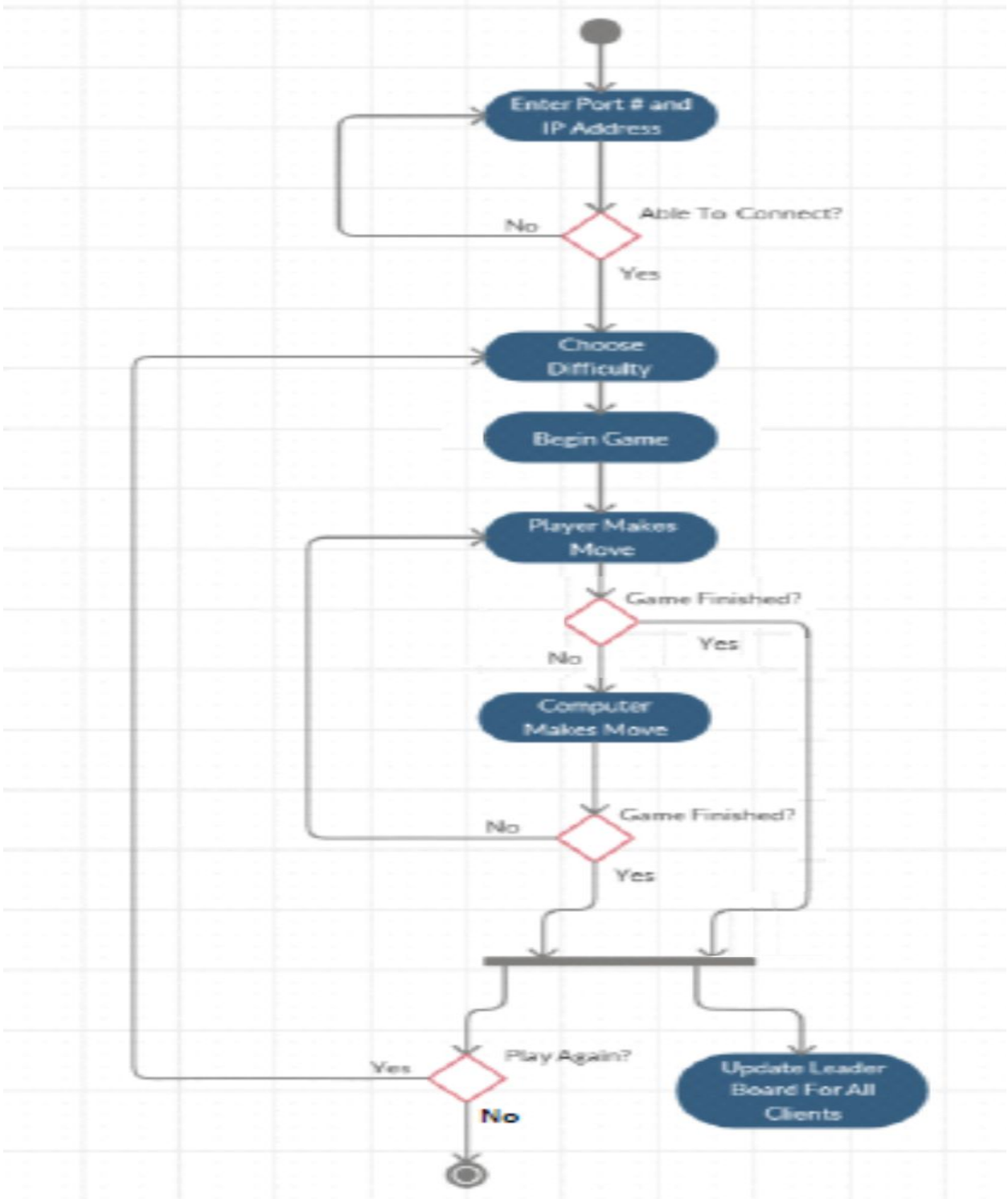
Server Wireframe



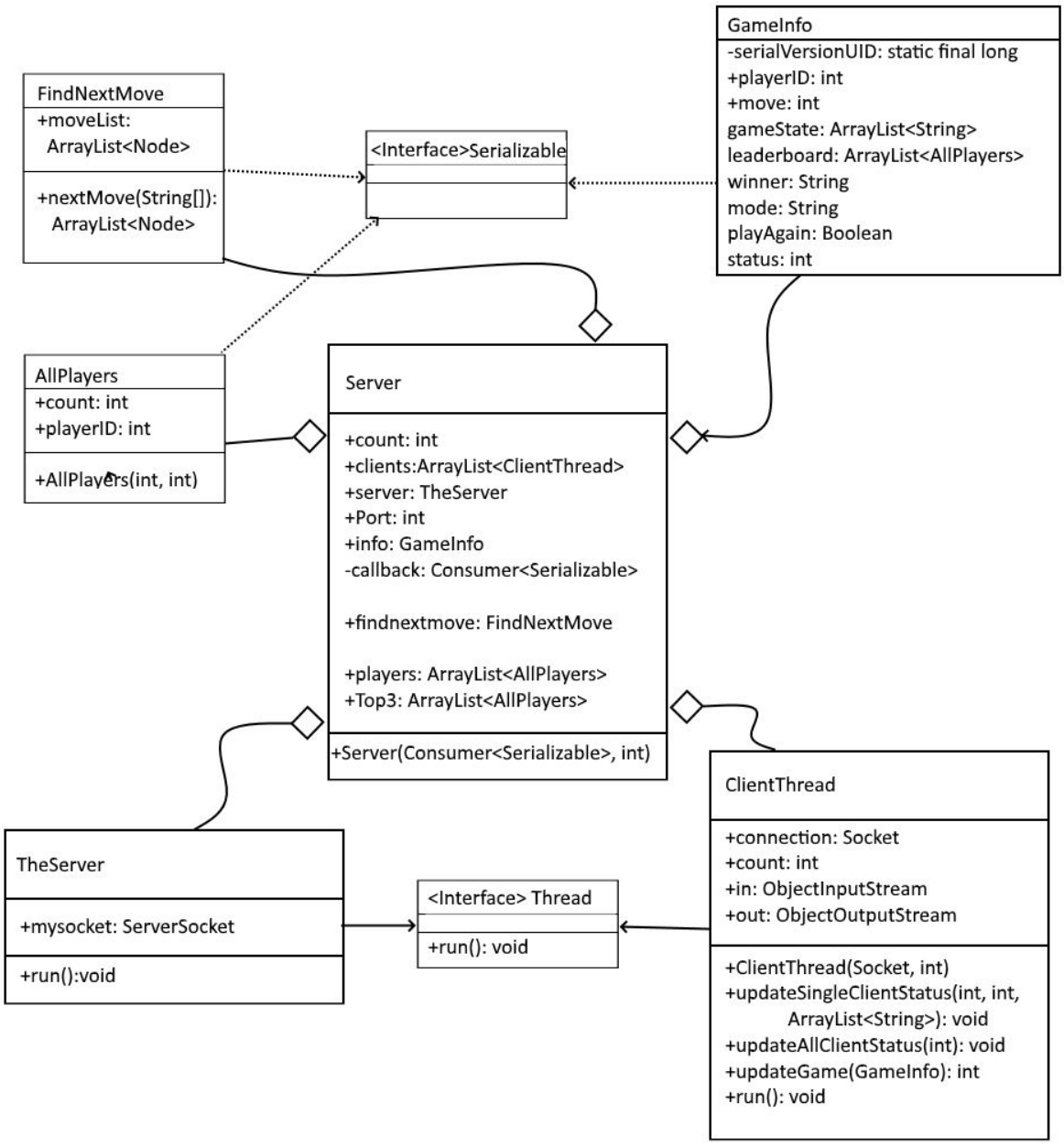
Client Wireframe



Activity Diagram



UML Class Diagrams



UML Class Diagrams(Continued)

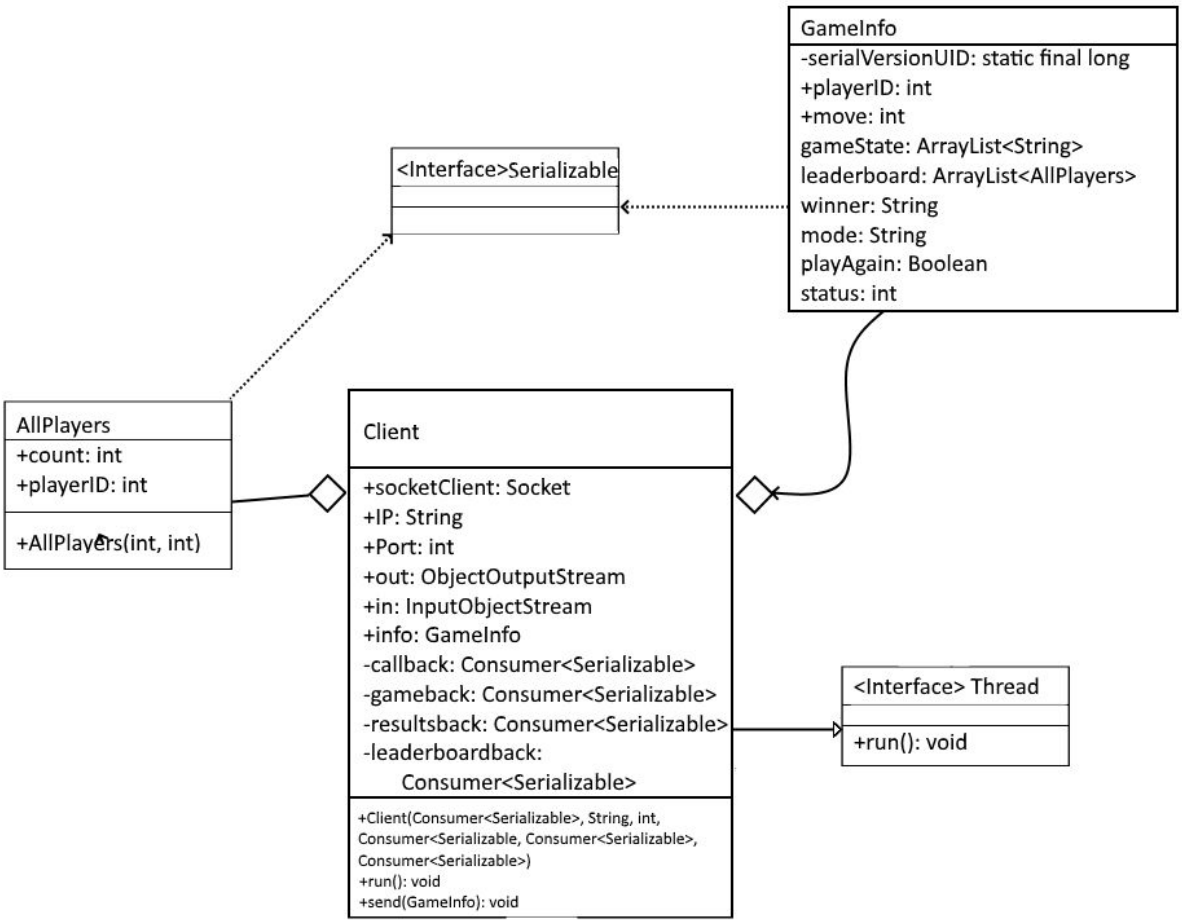


Diagram Changes

Wireframe

Cleaned up wireframe to match project visuals

Activity Diagram

Added another state between “Player Makes Move” and “Computer Makes Move” whether or not the “Game Is Over” because a player can win and the game will end.

Changed where “Play Again” goes to, instead of going to “Begin Game”, now goes to “Choose Difficulty”

UML Class Diagram

Added Classes Functions, Variables not included in initial class diagram