CS408 – Computer Networks – Spring 2024

Term Project : Networked Multiplayer Rock-Paper-Scissor Game

This project is made of two steps; each has different deadlines as specified below.

- Project Step 1
    - Submission: 23:50 on April 20, 2024
    - Demo week: April 22 - 26, 2024
- Project Step 2
    - Submission: 23:50 on May 18, 2024
    - Demo week: May 20 - 24, 2024

Introduction

In this assignment, you will work in a group to design and develop a networked multiplayer Rock-Paper-Scissor game. This assignment is divided into two stages: a first version and a final version. In the first version, you will focus on implementing the GUI of player and server and networking code. In the final version, you will add additional features such as adding more users, error handling and game statistics.

The application consists of two separate modules. To conduct the game, you need to build a server module. Furthermore, for your players to connect and play, you also implement a client module. In this document, the client module will also be referred simply as player, either term may be used to express the same concept.

The players will be able to enroll in an open Rock-Paper-Scissor game by connecting to the server and will be able to play with other players. On the other hand, the server will act as a facilitator/presenter so that it handles incoming connections from players so that they can participate in the game and play the game properly. Also, the server should keep the scores during the game. Moreover, it broadcasts the overall score table as well.

The server listens on a predefined port and accepts incoming client connections. Max 4 clients connect to the server at the same time. Each client knows the IP address and the listening port of the server (to be entered through an interface). Clients connect to the server on the corresponding port and identify themselves with their names. Server needs to keep the names of currently connected clients in order to avoid the same name being connected more than once at a given time to the server. Additionally, the server will maintain a record of users' victories, tracking them via their usernames in a .txt database. This will involve the creation of a "leaderboard.txt" file on the server, where usernames and corresponding win counts are stored. Upon joining the game, players will be able to see their win totals displayed alongside their usernames. Should a player secure a victory in the game, the server will then increment their win count accordingly. Also on the server side there should be richTextBox that will show the leaderboard of users.

The abovementioned introduction is for the entire project, which is to be completed in two steps. Second step is built on the first step and each has specific deadlines and demos.

The details of each step and some other general information are given below. Please read them carefully.

Project Details

You are going to develop a server module that can accept client connections. The connected clients should have unique names. Thus, if a client wants to connect with a name that is currently connected to the server, then the server should reject that name. The details are explained below.

Game Rules

- The game is played by 4 people.
- Each player must have a unique name. This name must be entered using the Client UI. This name is also sent to the server. The server identifies the clients using their names.
- On the client screen players see other players' names and their win count.
- The game will start when 4 players are connected to the server. If there is a request by the 5th player, the server should send a "The room is full" message to that player and add the player to the waiting queue.
- Server will inform players that the game will start in 5 seconds and a countdown is shown on the screen (5, 4, 3, 2, 1, Go).
- If one or more players disconnects while the countdown starts then the server automatically picks the first player from the waiting queue.
- In such cases other players have to see the "Player X left the game" message. If the player reconnects, it cannot join the ongoing game. After the game is started one or more players may leave the game at any moment (There should be a leave the game option on player screen). In such cases other players have to see the "Player X left the game" message. If the player reconnects, it cannot join the ongoing game. Also the server should not get players from the queue until the game is finished.
- If a game is started and three people leave the game then the server should select the winner automatically from the player who stayed in the room. If in the nth attempt two players are selecting their choice and one of the players leaves the game server should select the one as the winner who stayed.

- If the user leaves the game he or she can watch the other attempts. Leaving the game means not attending the game, it is not disconnecting.
- Players select their hand gesture by clicking on related letters (R, P or S).
- Players have 10 seconds to enter a hand gesture, otherwise he/she will lose.
- The server evaluates the results and shows it on the client screen.
- Rules of the game: rock crushes scissors, scissors cuts paper, and paper covers rock.
- If there is a tie situation, they will replay that turn.
- If there's a clear winner (a single gesture that beats all others), that person is declared the winner. If there's a tie, the players with the winning gestures continue to the next round.
- The player who successfully beats all other players wins the game.
- Server will send the results to the clients.
- The server will start a new game after the previous game is over.

Sample Game scenes :

Initially:	first attempt	second attempt	....	nth attempt	Finally
					Game over. X wins!

To help you understand the project better, we've made up a sample situation that explains how the game and it's waiting line system work. This example is just to make things clearer and show you how things might work. Remember, this is just one way to look at it, and you shouldn't only think about this example. You can see the example scenario from Example_Scenario_Project.pdf.

Networking

- The game will be played with network communication.
- The server will be responsible for creating game rooms and handling incoming player requests.
- Players will communicate to the server to request to join the game room, make a gesture, or leave the game (There should be a leave the game option for players).
- The server will broadcast the current state of the game to all players in the game room after each player's action.
- There is only one server running on the system.
- For the client, the server IP address and the port number must not be hardcoded and must be entered via the Client UI

- The server will start listening on the specified port. It must handle multiple clients simultaneously
- The players in the game can disconnect at their will and this should not cause your program to crash. If the player reconnects, it cannot join the ongoing game.
- Error handling: The game should handle errors gracefully and provide informative error messages to the players.
- Game statistics: The game should keep track of the number of games played, number of wins/losses/draws for each player, and display these statistics to the players.
- All activities of the server should be reported using a rich text box on the Server UI including names of connected clients as well as all the scoring, etc. Be as verbose as possible. <u>We cannot grade your homework if we cannot follow what is going on; so, the details contained in this text box is very important</u>.

Game scenarios for four people rock, paper, scissor game:

- All Unique Choices
    - Rock, Paper, Scissors, Rock: No clear winner or loser since each choice defeats and is defeated. All players move to the next round.
- Three Choices Alike
    - Three Rocks, One Paper: The three Rock players are eliminated; Paper wins the game.
    - Three Papers, One Scissors: The three Paper players are eliminated; Scissors wins the game.
    - Three Scissors, One Rock: The three Scissors players are eliminated; Rock wins the game.
- Two and Two
    - Two Rocks, Two Papers: Rocks are eliminated; Papers move on.
    - Two Rocks, Two Scissors: Scissors are eliminated; Rocks move on.
    - Two Papers, Two Scissors: Papers are eliminated; Scissors move on.
- Two of One Choice, One of Each of the Others
    - Two Rocks, One Paper, One Scissors: Paper and Scissors are eliminated; Rocks move on.
    - Two Papers, One Rock, One Scissors: Rock and Scissors are eliminated; Papers move on.
    - Two Scissors, One Rock, One Paper: Rock and Paper are eliminated; Scissors move on.
- All the Same Choice
    - All Rocks: All move to the next round.
    - All Papers: All move to the next round.
    - All Scissors: All move to the next round.

Project Step 1 – Code Review - Demo

Below are requirements for Project Step 1. If you don't do Step 1 you will get %25 deduction from the final grade of your project.

- Server's and Player's GUI should be ready.
- Players can connect to the server. When 4 players are connected to the server it will notify the players that the game will start.
- On Server's GUI there should be a leaderboard from higher wins of usernames to lowest. And also when players join the room other players can see his or her total win count near to their usernames. For Project Step 1 all will have zero but you can adjust manually from the .txt file as fake wins.
- Submission: 23:50 on April 20, 2024
- Demo week: April 22 - 26, 2024
    - We will share a google sheet where you can select a time slot for making a demo April 22 - 26, 2024.

Project Step 2 – Submission & Demo

- Submission: 23:50 on May 18, 2024
- Demo week: May 20 - 24, 2024
    - We will share a google sheet where you can select a time slot for making a demo between May 20 - 24, 2024.

Group Work

You can work in groups of three – four people for both steps (not less than three except really exceptional cases). No group changes are allowed after the first step. Any group changes must be performed before the submission of the first step. However, we do not recommend changing groups once you start the project since moving codes might lead to plagiarism.

There will be 3 points peer evaluation grading for term project groups. Each group member will evaluate the performance of other members.

Equal distribution of the work among the group members is essential. All members of the group should submit all the codes for both client and server. All members should be present during the demos. In case of any dispute within the group, please do not allow the problematic group members to submit the code, so that he/she will not be graded. However, if a group member submits the same code, then the other members automatically accepts his/her contribution.

Your TA (yasinugur@sabanciuniv.edu) is responsible for keeping track of the groupings. He will send an announcement to the class about how the group information will be collected, how underpopulated groups will be merged and how people without groups will be grouped.

You have a chance to form your own groups. However, if you cannot find enough people to form a group, then you have to accept to work with people that we assign. If you do not like to work with people that you do not know, then form your own groups!

Programming Rules

- Preferred languages are C#, Python and Java, but C# is recommended.
- Client and server programs should be working when they are run on at least two separate computers. So please test your programs accordingly.
- Your code should be clearly commented. This affects up to 5% of your grade.
- Your program should be portable. It should not require any dependencies specific to your computer. We will download, compile and run it. If it does not run, it means that your program is not running. So, do test your program before submission.

Submission

- Submit your work to SUCourse. Each step will be submitted and graded separately.
- All group members must submit the same code! Not submitting means no contribution, so non-submitters will not be graded.
- Delete the content of debug folders in your project directory before submission.
- Create a folder named **Server** and put your server related codes here.
- Create a folder named **Client** and put your client related codes here.
- Create a folder named **XXXX_Lastname_OtherNames_StepY**, where XXXX is your SUNet mail ID and Y is the project step (1 or 2) (e.g. yasinughur_Ughur_Yasin_Step1). Put your Server and Client folders into this folder.
    - Compress your XXXX_Lastname_OtherNames_StepY folder using ZIP or RAR.
- You will be invited for a demonstration of your work. Date and other details about the demo will be announced later.
- 24 hours late submission is possible with a 10 points penalty (out of 100).

For personal questions and support, you can contact course TAs.

Good luck!