Real-time network multiplayer game

RITTWICK BHABAK, Indian Institute of Technology, Delhi, India
SAGAR AGRAWAL, Indian Institute of Technology, Delhi, India
MANIK JAIN, Indian Institute of Technology, Delhi, India

This is a system architecture project where a **Real-time network multiplayer Game** is built. The application is divided into multiple components and the components are loosely coupled. In this application we have used a single server and there can be multiple clients and one database system.

Additional Key Words and Phrases: System Architecture, GUI, Version Control, Project Documentation, Re-distributable packages.

1 INTRODUCTION

The application is built in python programming language as there are a lot of materials available how to build GUI applications in python. Although the underling architecture is independent of the programming language. The architecture which is followed can be implemented by using any programming language and it is also independent of the platform and operating system. The layers and the components inside the layers are loosely coupled. There is a server to which clients are communicating and the server maintains a database to keep track the records and helps the clients to roll back to handle the clients who are in slow network.

2 ABOUT THE GAME

We have made a multiplayer game.So the rule of the game is that initially the screen will have many squares and the players have to click on those squares to earn points and the player with maximum points finally wins the game.So, initially the screen will have many squares, all squares will have their coordinates. The game will have multiple players and all of them will join the server to play the game.All squares will be at same position in all the players screens. All players click on join button. After that, the game will start. So the game is to click on the squares. As one player clicks on the square, that particular square should disappear from screens of all the players and that player will get the point for that square. Now if there is a delay, meaning if a player clicks a square and another player clicks on the same square before the square disappears, so this way both players will get the point for the same square. This should not happen. So, to resolve this every square will have its id and when a player clicks a square, the player's name along with the time at which square is clicked will be sent to the server. In this way, if a player clicks the square after a player, then its points will be rolled back as his timestamp value will be higher than the other player. This way, as all the balls disappear from the screen, the game will finish and finally the name of the winner will show up on the screen.

3 ARCHITECTURE AND DATA FLOW

3.1 Architecture

The application is divided into the following components:

- Server
- Client
- Database

Authors' addresses: Rittwick Bhabak, Indian Institute of Technology, Delhi, Delhi, India, mcs222054@iid.ac.in; Sagar Agrawal, Indian Institute of Technology, Delhi, Delhi, India, mcs222065@iid.ac.in; Manik Jain, Indian Institute of Technology, Delhi, Delhi, India, mcs222832@iid.ac.in.

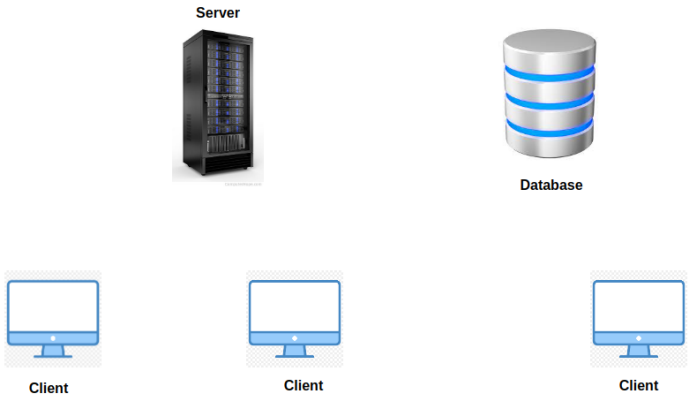

Fig. 1. System Architecture

There can be any number of clients in our system and all of the clients will run an instance of 'Client' separately from different machines.

The clients cannot communicate with each other directly. The clients send messages to the server and server sends responses to that client and sometimes server has to send broadcast messages to every client.

The server maintains a database so that it can handle all of the concurrency issues and sync all of the clients. Here we've used NoSql database so that Server can insert fields as per requirements.

3.2 Data Flow

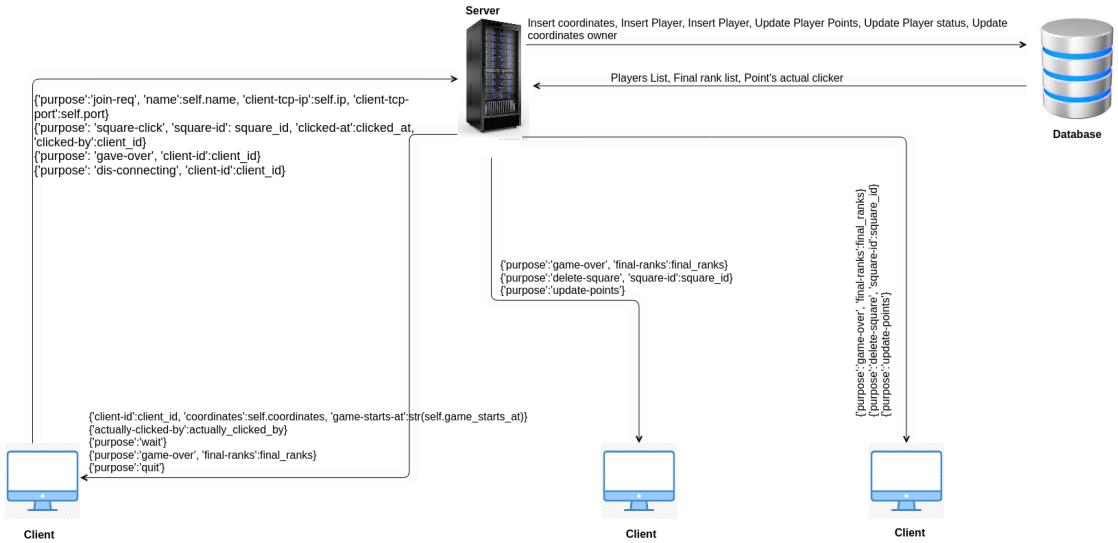

Fig. 2. Data flow

3.2.1 *Messages from Client to Server.*

- 'purpose':'join-req', 'name':self.name, 'client-tcp-ip':self.ip, 'client-tcp-port':self.port
- 'purpose': 'square-click', 'square-id': square_id, 'clicked-at':clicked_at, 'clicked-by':client_id
- 'purpose': 'gave-over', 'client-id':client_id
- 'purpose': 'dis-connecting', 'client-id':client_id

3.2.2 *Messages from Server to a particular Client.*

- 'client-id':client_id, 'coordinates':self.coordinates, 'game-starts-at':str(self.game_starts_at)
- 'actually-clicked-by':actually_clicked_by
- 'purpose':'wait'
- 'purpose':'game-over', 'final-ranks':final_ranks
- 'purpose':'quit'

4 INTERNAL MODULES

There are mainly 5 internal modules in the application:

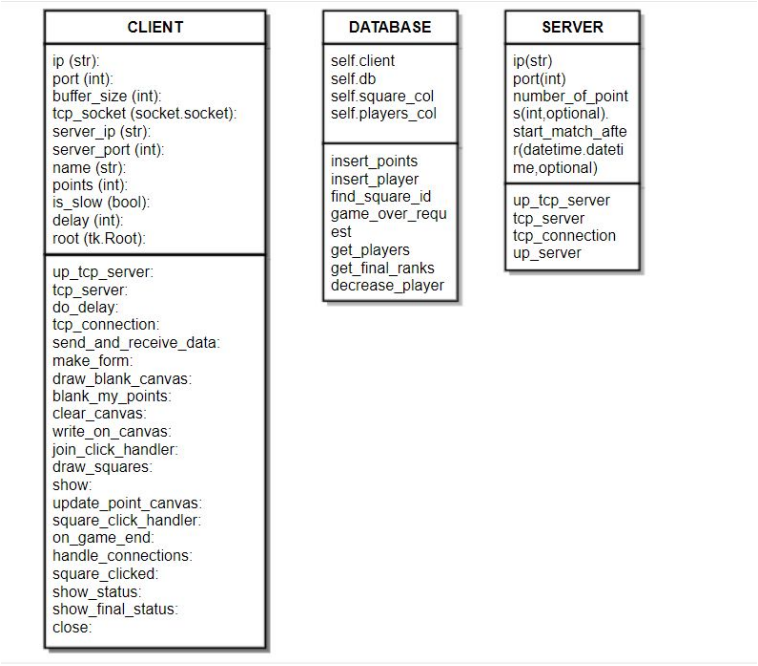

Fig. 3. Classes

Client is the class. Each player has to run one instance of the **Client**.

Server is another class. Server machine will run one instance of the server.

Database is another class. Server maintains a database to handle concurrency issues and roll backing instructions for clients.

Shared Module: The shared module is used by both server and clients.

4.1 server.py

In the server there are a tcp server running always. The clients to that tcp server and sends the joining request. The server creates an record on the database for that client and sends an unique id for that client and the position of the co-ordinates of the squares as response. Also when the client send the input data, server stores the data and checks if some other player has clicked on that square previously, if yes then after matching the timestamps the server sends appropriate messages to both of the clients whether the points increases by the clients for themselves has to be decreased or not.

4.2 client.py

The client takes the ip address of server's tcp server and sends an joining request on the beginning. After server sends the response, client stores gets its id and waits for the starting time. When the starting time comes the client

draws the squares on the screen. When player clicks on a square, client first increase its points, then client sends that square id and timestamp when the square is clicked to the server. After the analyzing the timestamp if server sends to decrease the points, client decreases its points. Also, client listens to a different tcp server, and there can server sends a 'decrease-points' message, then also client decreases its points.

4.3 database.py

4.4 measurements.py

This is a shared module. Using the measurements module the server generates the points. Server uses the screen size, squares sizes to generate non-overlapping random points. Using this measurements and the coordinates the client makes its screen and draws the squares on the screen.

5 SIMULATING LATENCY AND ROLLING BACK

5.1 Simulating Latency

At the starting of client, it takes input whether the client is on a slow network. If it is entered yes, then after sending the request the client waits 'delay' amount of time also after getting the response the client waits also 'delay' amount of time before processing the delay. This delay actually simulates the delay in the network. Here we are physically simulating some amount of delay but in case of real network this delay will induced by the networks.

5.2 Rollback

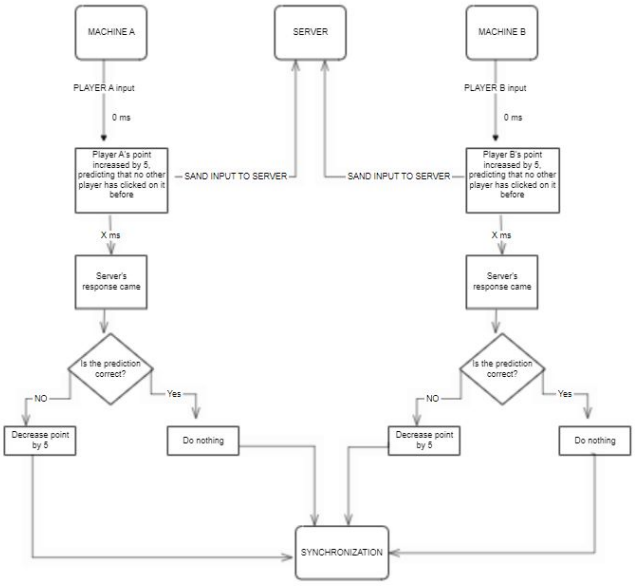

Fig. 4. Rollback

Also please note that the client also has a tcp server to which server can send 'decrease-points' request. The reason is, lets assume a player A has clicked on square 5 on time t=10 but the request has reached to the server very late, say at t=50. Now assume that player B has clicked on the square 5 on time t=15 and this request has reached to the server at t=20 and server stored player B as winner for square 5. But when playerA's request has reached to server, the server has to tell player B to roll back, i.e. to decrease its points by 5. Now this request from server to client will go to the cilent's tcp server. And thus consistency will be held. And also server will make the clients wait until 'game-over' request (which indicates that game is over from all clients) came from all clients.

6 EXTERNAL MODULES

Several external modules are used in the application:

- `tkinter`: Tkinter is used to make the GUI
- `sphinx`: Sphinx is used to make project documentation
- `pymongo`: Pymongo is used to connect database and server.
- `datetime`: To get timestamps for clicking times.
- `socket`: To do the networking stuffs the socket python library is used.

7 SCREENSHOTS OF GUI

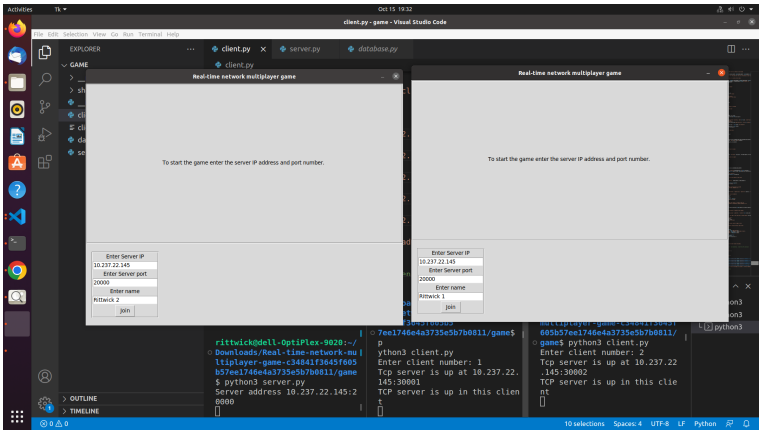

Fig. 5. Joining Screen

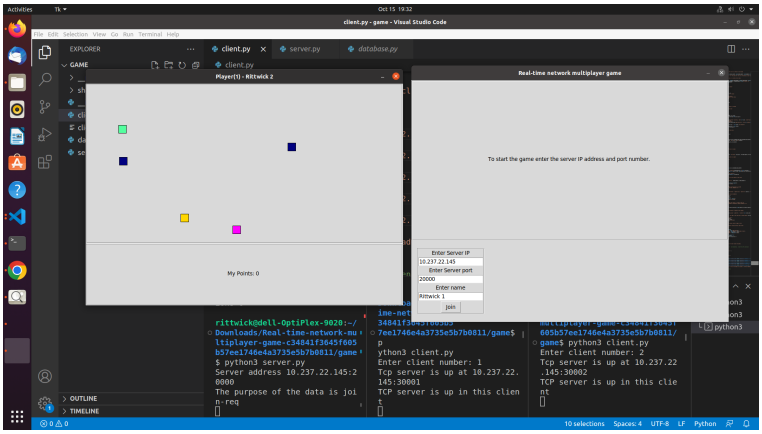

Fig. 6

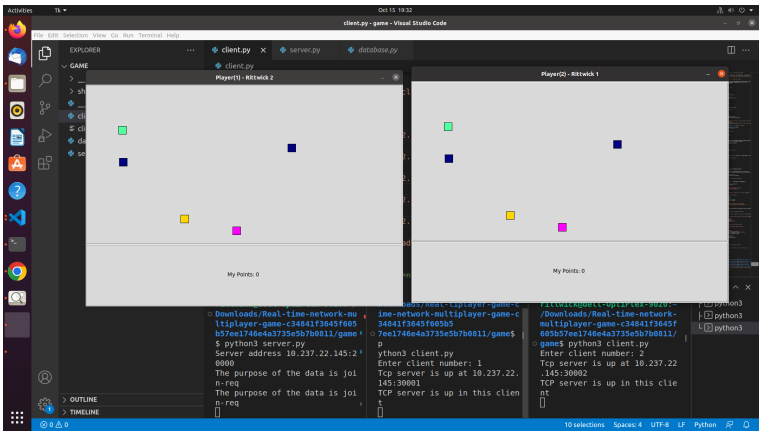

Fig. 7. After the points are rendered on the screen.

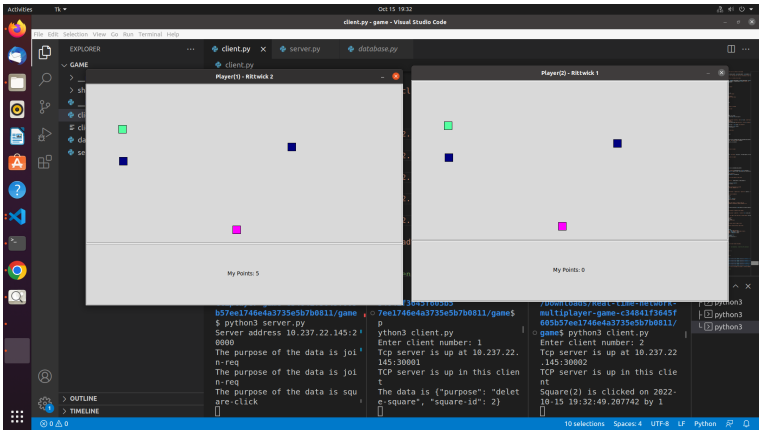

Fig. 8

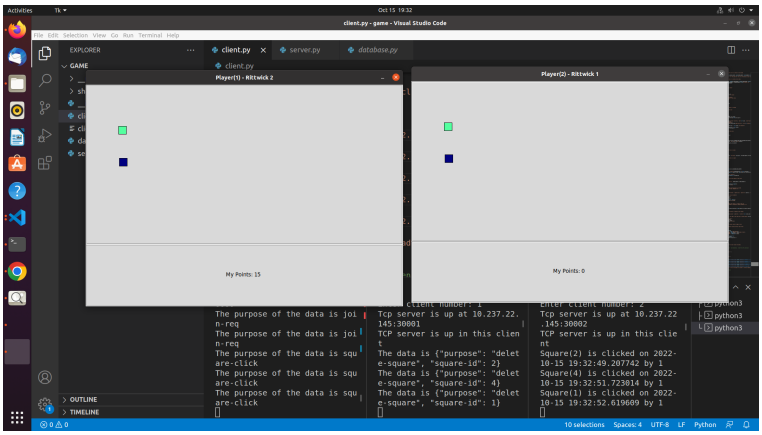

Fig. 9

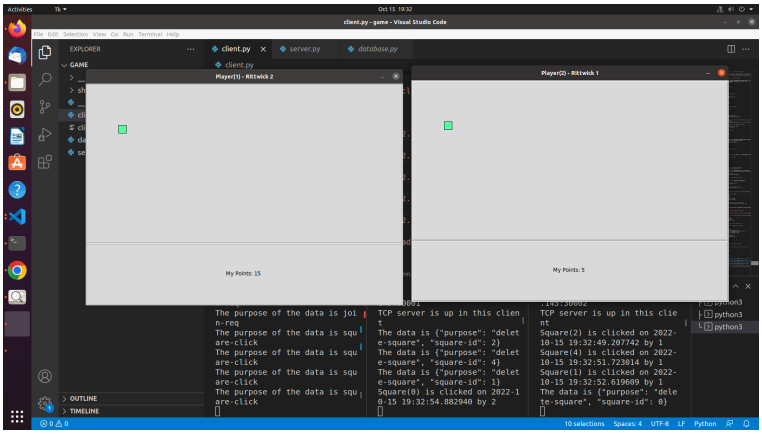
client.py - game - Visual Studio Code
Player(1) - Rittwick 2
Player(2) - Rittwick 1
My Points: 15
My Points: 5

Fig. 10

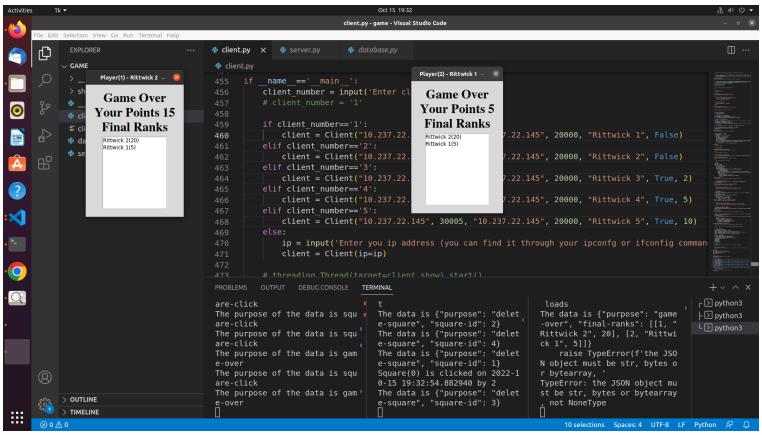
client.py - game - Visual Studio Code
Player(1) - Rittwick 2
Game Over
Your Points 15
Final Ranks
Rittwick 2(20)
Rittwick 1(5)
Player(2) - Rittwick 1
Game Over
Your Points 5
Final Ranks
Rittwick 2(20)
Rittwick 1(5)

Fig. 11

8

- Version Control Using github.com. The link to git repository: https://github.com/rittwickBhabak/Real-time-network-multiplayer-game/tree/main
- Project documentation using Sphinx Library
- Redistribute package, whl file

9 FURTHER SCOPE

- Presently the project has been developed taking one time zone into consideration i.e. Indian local time. The same may be expanded to client working in different geographic location with different time zone. For ex. Different client in India and US.

10 CONTRIBUTION BY DIFFERENT GROUP MEMBERS

- `Rittwick Bhabak`: Contributed in deciding the architecture and has build the server module and client module. (50% of total contribution)
- `Sagar Agrawal`: Contributed in deciding the architecture and build some parts of database module and made the project documentation. (25% of total contribution)
- `Manik Jain`: Contributed in deciding the architecture and build some parts of database modules and shared module and made the automated documentation of the project. (25% of total contribution)

ACKNOWLEDGMENTS

We would like to express our special thanks of gratitude to my teacher Rahul Narain sir as well as our TA and CodeWithHarry[**codewithharry**] YouTube Channel, Codemy.com[**codemy**] YouTube channel, AI Swiget[**atbswp**] and Bhagath Mamindlapelly YouTube channel who helped us to do this wonderful project on the topic **Real-time network multiplayer game**, as a result we learnt so many new things. We are really thankful to them.