Distributed Shared White Board

Dechao Sun (980546)

1. Introduction

This is a project to implement a real time distributed shared white board which allows a manger and multiple users to draw shapes, input text, chat with other users and so on.

2. Project Structure

2.1 Class components

In this Project, I have implemented 12 classes which could be separated to 4 parts (server, client, GUI and drawing helper functions).

Server part: Server class, ServerThread class.

Client part: CreateWhiteBoard class, JoinWhiteBoard class, Client class.

GUI part: WhiteBoardGUI class.

Drawing helper functions part: User class, UserList class, PaintSource class, PaintSourcList class, Draw class, PaintBoard class.

First, Server will be started with input the IP address, port number. After that server is running and listening the connection request. When the new connection constructed, server will build a thread for that connection for communication. The ServerThread class will receive requests and process it, and then send back to the client.

Second, for client part, the shared white board system required a manager of the system. The CreateWhiteBoard class will be started. It will generate a manger to the shared white board system. After that, the JoinWhiteBorad class will be started to generate users to the system. Both two classes require IP address, port name and username as command line inputs. Then the Client class will be constructed in create and join classes. It will receive, process and send information to the server. At this step, the real client-server connection will be built.

Third, for GUI part, Client class will run the WhiteBoardGUI class. It will implement all the user interface which includes user paint board, chat box, drawing selection buttons and user list.

Fourth, User stores information about the user such as user socket, user name and manger Boolean. User will be stored in UserList class which stores all the users' information. In the same way, PaintSource class stores the paint information like drawing type, drawing color, drawing text and so on. PiantSourceList stores all the PaintSource. Furthermore, PaintBoard class is used to listen mouse movement and then send instructions to Draw class. Draw class will paint the final picture on the paint board.

2.2 Socket

In the distributed shared white board application, I have implemented TCP connection which is a connection-oriented, reliable and ordered connection. It will send data in order which means no request will be missed. It guarantees the picture completeness. It is important for the system. UDP connection cannot guarantee the completeness of the picture. Although it faster transmission than TCP. However, speed is not really a problem for the application. Therefore, I choose to use TCP connection.

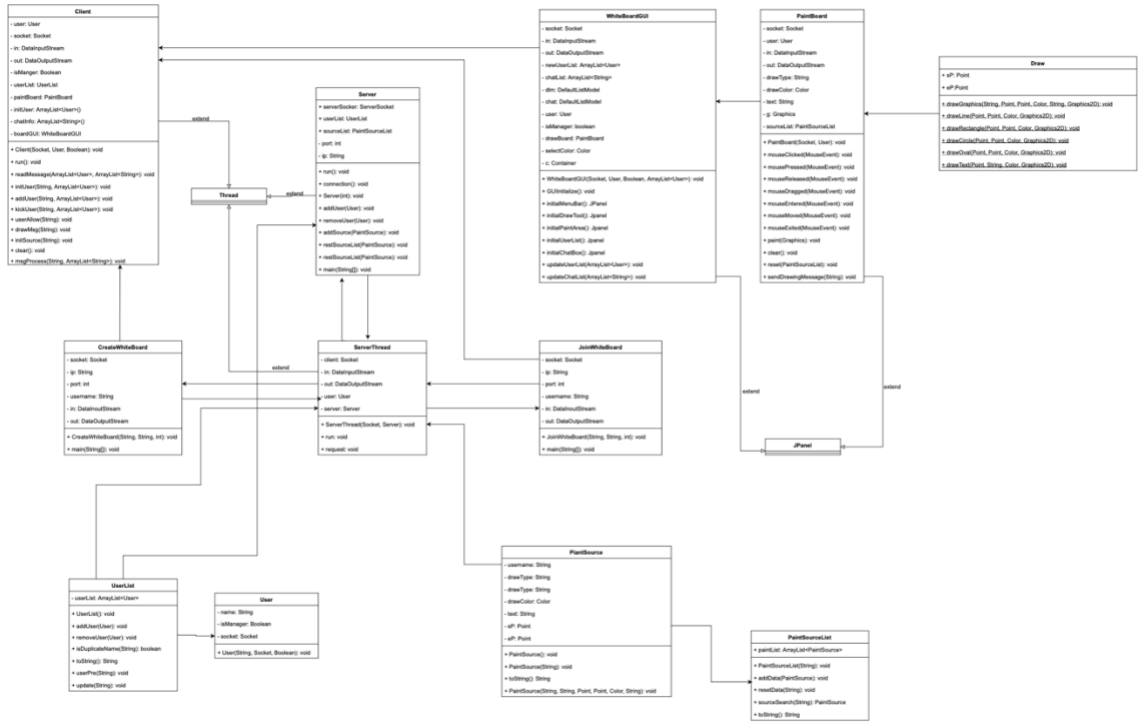
2.3 Thread

In this project, I have implemented ServerThread class to support multi thread execution with thread-per-connection architecture. When the new user connection to the server, the connection will be holed until the user close the connection or manger close the connection. The thread-per-connection model is clear for information transmission. Moreover, clients and the server is connected all the time. It is fast for client-server communication. However, it will leads to context-switching is high, because of too many threads executing in parallel. It will have problems when hundreds of thousands thread is running. For my small application, clear code and communication structure is more important for the distributed shared white board system. Therefore, I choose to use thread-per-connection architecture.

3. Project Diagrams.

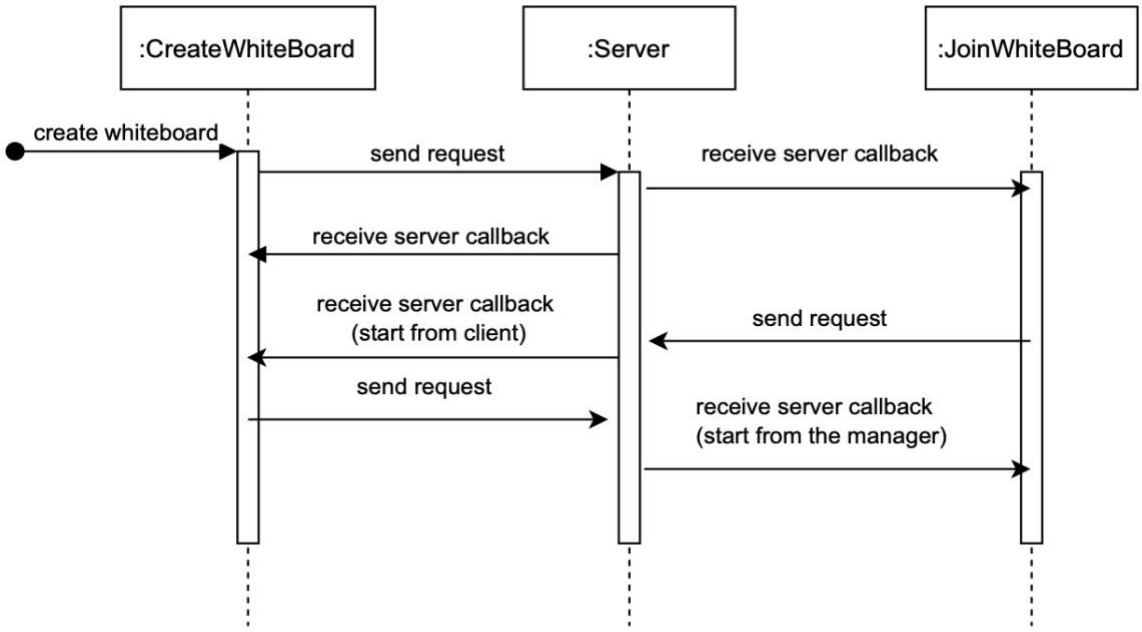
3.1 Design Class Diagram.

I have implemented 12 class as I pervious explained. The deign class diagram below shows the relations between classes.



3.2 Interaction Diagram

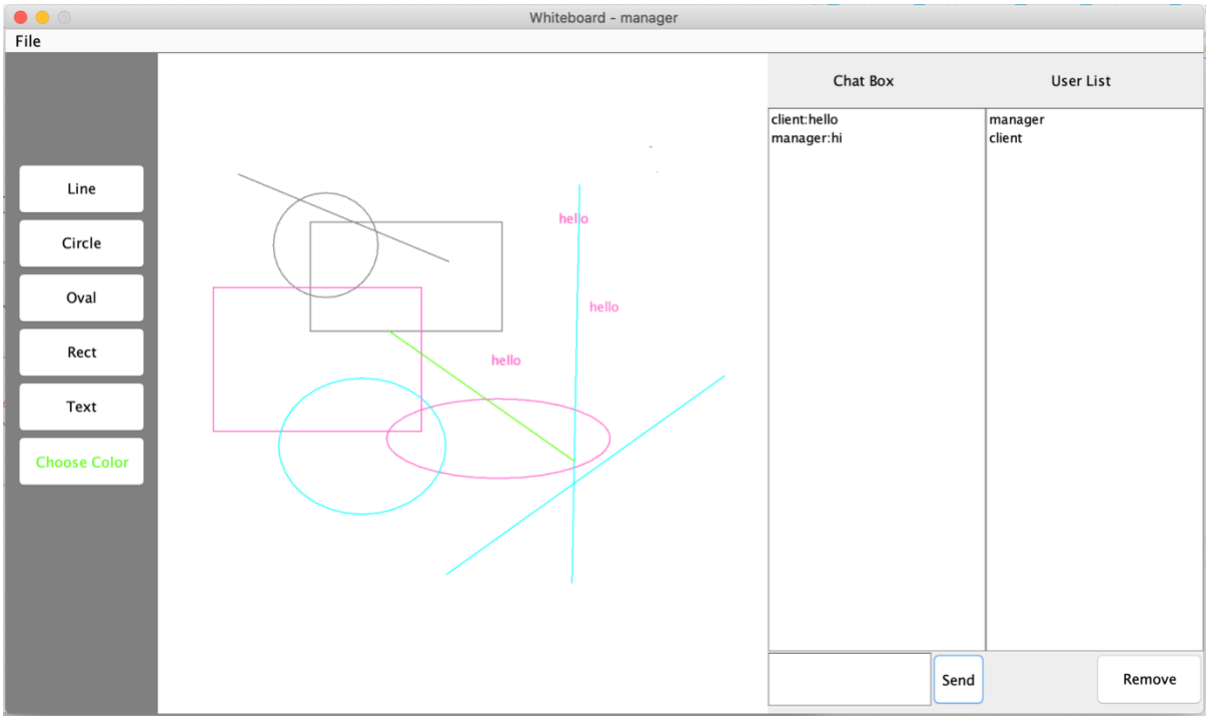
The distributed shared white board system start from server. After that, a manger (CreateWhiteBoard) will connect to server. And then, a client (JoinWhiteBoard) will send join request to server, server will send the request join information to manager. The Manager will send the join result to server, then server will send to the client. If the join result is yes. Client will join in the shared white board. It is a complete request join communication loop. Furthermore, users cannot communicate with each other directly. All the information will be process by server first, then send to the clients. Therefore, once a user draw a picture, the picture will be sent to server. Server will broadcast to all the other users in the shared white board system.



3.3 GUI Diagram

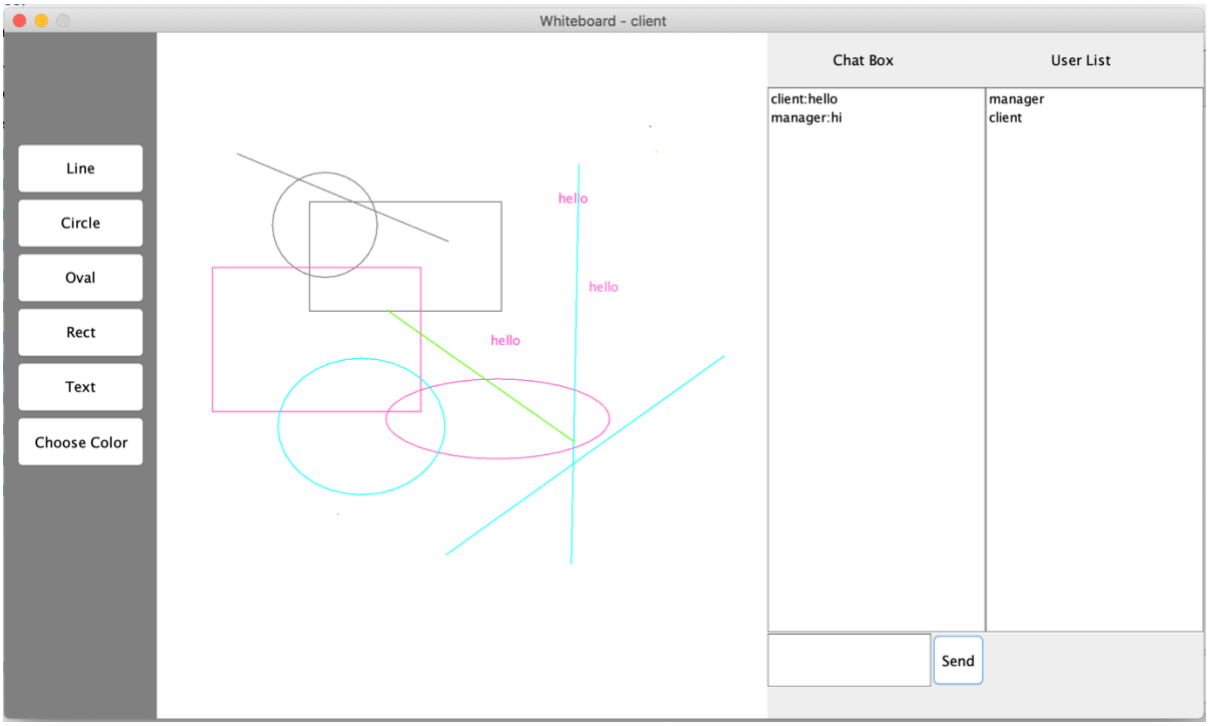
3.3.1 Manager UI

The manager interface has drawing type and drawing color selection buttons on the left. In the middle, it is a paint board which allows users to draw any pictures. On the right side, it is a chat box, users input the text in the box below, then presses send button the chat information will show on the chat box. On the most right, it is a user list, It shows all the users who current in the shared white board system. At the bottom right corner, there is a remove button that allows managers to remove users in the system. Furthermore, at the top left corner, there is a file button that allows user to clear the paint board and exit the system. The remove and file buttons are the manager's functionalities. Thus, these two buttons only show on the manager's interface.



3.3.2 Client Interface

Clients UI have almost the same interface expect the remove and file functionalities.



4. Advance Features.

1. I have implemented chat window (text based): users could send messages to the client box to achieve chat functionality. It shows on the GUI diagram above.
2. I have implemented a file menu with new, open, save, saveAs and close buttons. However, I only achieve new and close button functionalities.
3. I have implemented manger kick out user functionalities. Manager could kick out clients except the manager self.

5. Conclusion

This project achieves a distributed shared white board system. It uses the TCP connection for reliable and ordered connection. Second, the project uses thread-per-connection architecture that is a fast server-client communication model. Third, all data operations are synchronized. To avoid two operations access on same data which will lead to inconsistent state. Lastly, the GUI implementation is more clear and easier for the user to use the shared whiteboard system.