**Problem analysis**

The main problem of this project is to rewrite the server-side code so that multiple users can participate in the game. In order to do this, the following issues need to be considered:

1. How does the server monitor each client at the same time

2. How does the server decide whether to start the game

3. How does the server ensure that the game progress of all players is synchronized?

4. How does the server increase the audience while the game is in progress, and synchronize the progress of the game

5. If you solve the impact of the player's midway launch on the game

6. Join the client to disconnect, how to clean it up, and ensure the normal progress of the next round of the game

In view of the above problems, the main framework of the server code is drawn below:

**Detailed solution**

1. Use multithreading

Start a child thread to monitor client connections through the listening port created in the main thread. The important part of this part in the server code is client\_listen(sock). After receiving the client connection, use tiles.MessageWelcome() to assign an id number to the client. If the game is in progress, then the customer will become the audience, tell the customer the original game player through game\_client\_idnum\_init, and then synchronize the game progress of the customer through the game history information msg\_history.

2. Synchronize game progress for all customers

First, the information of each user is processed in client\_handler(). After receiving the user's information, judge whether the information can be straight-lined, and if it can be executed, inform the user that the operation is executed successfully, and at the same time send the successfully executed operation to each user. After the operation is executed, the game information is obtained through board.do\_player\_movement(game\_client\_idnum), and the eliminated players are synchronized to all clients.

3. About the connection problem of game customers

All client connections will be stored in the client\_connection\_list list. In addition, when sending information to the client, use the try method. If it fails, it means the client connection is disconnected. If it is a player, the player will be added to the elimination list, and then the audience id number will be recorded, and the game will be cleaned up after the game is over.

4. Engineering characteristics

For the server project, in order to be able to connect with multiple clients at the same time, it is necessary to use sub-threads other than the main thread to monitor, which is the biggest feature of this project. In addition, considering that, after the server determines the order of the player's rounds, each round is dedicated to a player, so there is no need for multiple threads to process player requests, only one thread is needed, which saves resources.

5. Limitations of the project

The project cannot start multiple games at the same time, nor can the user determine the number of games. Suppose that this game is very popular and there are many people who want to start the game, but every time the server only starts a handful of the game, the rest can only wait as spectators for the next round of drawing. I think a better server should be that users create rooms by themselves, wait for other users to join, and can allow multiple game rooms to start multiple games.

**Instructions**

If only one user is connected, the game will not start. If two or three users connect, they will wait for 3 seconds. If there is no fourth person to join, the game will start. If four people join, the game starts immediately.