Report on Client-Server Pong Game Implementation

Background:

The client-server Pong game is designed to allow two players to play Pong over a network. The server manages the game state and communicates updates to the clients. The clients handle user input, update their local game state, and send updates to the server.

Design:

Client Design:

The client script utilizes the Pygame library to create a Pong game interface. The `playGame` function manages the game loop, handling user input, updating the game state, and sending updates to the server. The `joinServer` function establishes a connection to the server, receives initial game information, and then launches the game loop.

Server Design:

The server script sets up a socket and listens for two clients to connect. It sends initial game specifications to each client and creates a separate thread for handling updates from each client. Two semaphores are used to ensure exclusive access to the game state and avoid data corruption.

Implementation:

Client Implementation:

The client script uses Pygame to create a game window, handle user input, and display the game elements. The `playGame` function includes the main game loop, where user input is processed, the game state is updated, and updates are sent to the server. The `joinServer` function establishes a connection to the server, receives initial game information, and then launches the game loop.

Server Implementation:

The server script initializes a socket, accepts connections from two clients, and sends initial game specifications to each. Two threads handle client updates concurrently. Semaphores ensure thread safety when updating the game state. The server sends and receives updates to and from the clients to keep the game synchronized.

Challenges:

- Synchronization: Ensuring proper synchronization between clients posed challenges. The server handles synchronization by comparing the sync counters of clients and updating the lagging client's game state.

- Multi-threading: Managing multiple threads for client updates and responses required careful synchronization using semaphores.

Lessons Learned:

- Thread Safety: Implementing thread-safe mechanisms, such as semaphores, is crucial when dealing with shared data among multiple threads.

- Networking Challenges: Handling network communication introduces complexities, especially in synchronizing game states between clients and the server.

Conclusions:

The client-server Pong game implementation provides a functional multiplayer experience over a network. Challenges were encountered in ensuring synchronization and managing multiple threads, but the implementation successfully facilitates gameplay between two clients.