**Ping-Pong Game Report**

**Abstract:**

A classic ping-pong game has been implemented with a client-server model. This project leverages the Python programming language and Pygame library for graphical rendering. The server handles the game's logic, while clients control their respective paddles. The communication between clients and the server is facilitated through sockets and JSON-encoded messages.

**Server Implementation:**

The server side of the ping-pong game is responsible for coordinating the game state and managing communication with multiple clients. It utilizes the TCP protocol for socket communication. The server keeps track of paddle positions, ball position, and scores for both players. The game loop continuously updates the game state, handles paddle movements, ball collisions, and determines when a player wins. The server supports two clients and ensures synchronized gameplay by exchanging acknowledgment messages.

Key Features:

* Game initialization and configuration for two players.
* Continuous game loop for updating and synchronizing game state.
* Handling paddle movements and ball interactions.
* Managing scores and determining the winner.
* Communication through JSON-encoded messages over sockets.
* Acknowledgment mechanism to ensure synchronization.

**Client Implementation:**

The client side is responsible for rendering the game using Pygame and interacting with the server to receive game state updates. Each client controls one paddle and communicates with the server to send paddle movements and receive updates on the game's progress. The client-side game loop handles user inputs, updates the game display, and communicates with the server.

Key Features:

* Pygame is used for graphical rendering and user input.
* Connection to the server using sockets.
* Sending paddle movements to the server.
* Receiving and processing game state updates.
* Displaying scores, paddles, and the ball on the screen.
* Acknowledgment mechanism for synchronization.

**Synchronization Mechanism:**

To ensure synchronization between clients and the server, acknowledgment messages are used. Both the server and clients send acknowledgment messages after critical events, such as the start of the game, paddle movements, and score updates. This helps maintain a consistent game state across all participants.

**Conclusion:**

The ping-pong game demonstrates the implementation of a client-server model for interactive multiplayer gaming. The server manages the game state, and clients control their respective paddles, contributing to a synchronized and enjoyable gaming experience. The use of sockets and JSON-encoded messages facilitates seamless communication between the server and clients, ensuring a responsive and engaging gameplay environment. This project has provided valuable insights into networking, game development, and collaborative programming.

Screenshots: