Technical Documentation

Zoltan Batoczki | Robert Florence

ServerUDP.py

Purpose

The server file acts as a listener for a client wishing to play a game of Rock, Paper, Scissors. It is set up using a UDP sockets. Since a UDP connection does not accept incoming connections like TCP does, there is the risk of data/packet loss. Multiple nets are implemented such as a timeout exception to take into account, the possibility of data loss. The server is purposefully programmed to keep its duties to a minimum in order to make this program run mostly on the client-side computer to minimize the need for computation.

Functions

- Main(): runs the main function of the server program. It creates a UDP socket and listens for incoming data. If the first command sent is ‘play’ it calls the function generateMove() to send back a random move to the client. If the command is ‘quit’, it breaks and closes the socket connection.
- generateMove(): Once data has been received, and properly parsed to play a game, the server calls the generateMove function which generates a random move of either rock, paper, or scissors, based on their numerical value (1=rock, 2=paper, 3=scissors) by using a built-in python random number generator. It then returns it to the main function, where it can encode the result and send it back to the client. Once the generateMove() function is finished, it resumes listening for further connections.

ClientUDP.py

Purpose

The client file allows a user to connect to a certain user-designated IP address and port. Once it successfully connects, the user can choose to play a game. The user will receive a move randomly generated by the server, in which it will be compared locally to the user’s pre-chosen move. The client program then determines the outcome. The user can keep playing the game

by continually sending ‘play’ after each result, at the menu, or end the game by sending ‘quit’. Upon quitting, the client-side program prints the final score of the play session.

Functions

- Main(): this function prepares a socket for the user to send information to. Asks user for IP and port. If successful, the user can either play a game or quit. If ‘play’ is chosen the client side program calls the play() function which compares the servers move response, tallies a score and output a final result. If the user decides he is finished and chooses quit, the client responds with the final score of the play session.
- play(): compares the user’s move to the server’s move. Determines a winner and updates the client-side score to reflect the result.