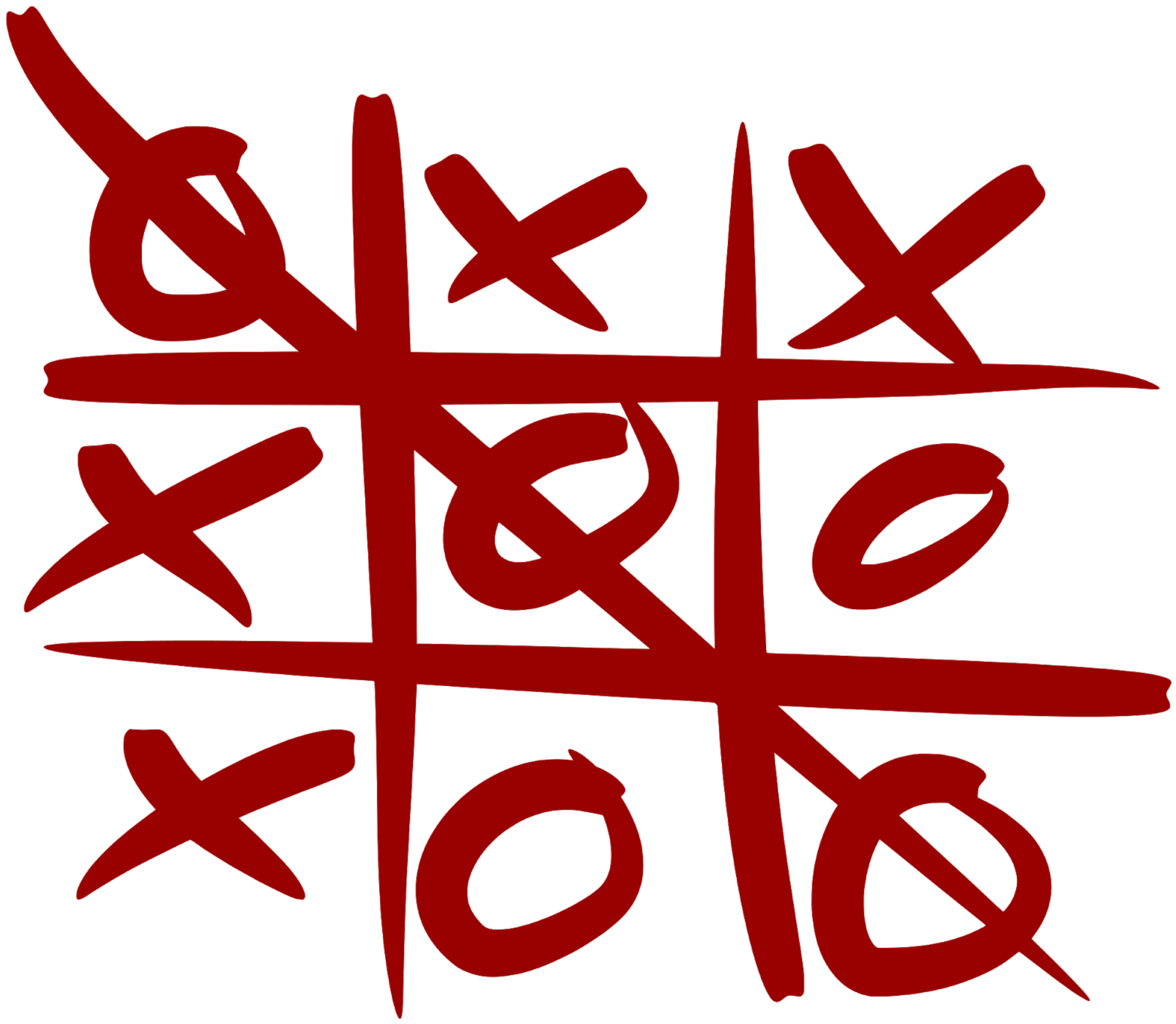
Tic Tac Toe!



Jamil Naber, jnaber3@uic.edu

Wisam Abunimeh, wabuni2@uic.edu

github.com/wisabi/TicTacToe.git

The general server and client logic is very similar to Project 4. The server class begins by creating a server socket on the user-inputted local port number. Once the server is generated, a new thread is created for the purpose of waiting for clients to connect. There is only one thread total that waits for clients to connect. When a new client connects, a new server-client thread is created for that specific client. There is a server-client thread for each client that is connected. Out streams and in streams are defined here to transfer game information between the server and each unique client.

The client class begins by trying to connect to the game server with the IP address and port number provided by the client. If there is a successful connection to the game server, then the game begins immediately prompting the player for their first move. Out streams and in streams are defined here to transfer game information between the server thread and the unique client.

The client and server transfer game information through a serialized class GameInfo. This class has data members such as the current board, the servers move, and the declared winner of the game. Every game has an instance of a GameInfo object.

The Min-Max code is implemented into the server project folder as it would be run from the server side. The Min-Max code method of gathering game board data is modified; instead of receiving the game board information from the terminal by scanning, the game board is a string object that is passed from the server as a parameter of the AI_MinMax constructor. The AI_MinMax class also extends class Thread and defines the run() method. This run method essentially implements the same commands and code of the original constructor. This new thread is started by calling the start() method from the server when a move is needed.