**2: Architectural Decisions**

We have a game lobby service. When a player wants to host a game, they must first choose and finalize the rules. The rules consist of the map selection, and possible modifiers to the game like increased wood production or decreased tree growth rate. Once the rules have been set, the user connects to the central server that lists all games to the public and tracks statistics.

The statistics we will be tracking are: the number of units of each type that were created, the number of units that died due to lack of funds, the average number of turns games take, the number of units a Player killed by invading and the number of resources a Player has taken from others by invading enemy territory. Statistics are transmitted to the server upon game completion.

We will be using a hybrid between peer to peer and client-server. The person who hosts the game is also a server. All other players connect directly to the host. The players get the game state from the host and the host simply updates his game state when playing before transmitting it to the other players. Therefore, the host contains the definitive game state. There will be one executable where the game host follows a slightly different logic than the peers while playing a turn.

Move verification will be done after the player tries to make a move. If the move is deemed valid then it is sent to the host, who then notifies the other players of the move. If the player making the move is the host, then all players are notified after the move is verified.

In order to load a game, the User must host a game then change the settings to specify that a game is loaded. The game cannot be started until the same players who were in the saved game have joined the lobby and declared themselves as ready.

When a user wants to save the game, the game host must approve the request. The client making the request then synchronizes the game state with the host and all users are notified which player saved the game.