How will information be passed between computers?

We'd need a client-server model:

- One machine acts as the host/server and maintains the game state.
- Others connect as clients and send inputs (like dice rolls, stone placement, turn actions).

The server sends:

- Current player turn
- Board state
- Dice values
- Player stats (names, scores, stones left)

The clients send:

- Dice roll commands
- Piece placement
- Turn completion

Control passing between computers

Right now, turn control is local: `currentPlayer` is tracked in a variable and used to enable/disable UI.

In multiplayer:

- The server determines whose turn it is and broadcasts that info.
- Only that client will have its controls enabled.
- Turn actions must be validated and confirmed by the server.

What Needs to Change

Game Setup

- Add hosting and joining options:
    - Host sets up player count, name, etc.
    - Others join using IP or server code.

User Interface Adjustments

- Add "Host Game" and "Join Game" buttons.
- Host sees a "lobby" with connected players.
- Clients just choose name and color — they don't control player count or start the game.

Player Roles in UI

- Host gets full UI (player count, start button, etc.).
- Clients get a simplified version — just name, color, maybe a "ready" toggle.

Game State Sharing

Currently

- Game state (like board, dice, scores) is stored in memory (`KiviGameplay` fields).

For Networking

We'd:

- Represent game state as a serializable object (e.g.`Serializable`).
- Send it to all clients after each significant change (e.g. dice roll, end turn).

What Data to Send

You probably *don't* need to send everything to everyone.

Option A – Full Sync (simpler):

- Server sends entire game state after each change.
- Clients just render and act accordingly.

Option B – Minimal Sync (harder):

- Only send relevant info (e.g., just dice values to current player).
- More efficient, but complex and prone to bugs.

For your project's size, Option A is safer.

Turn Handling Changes

Currently:

- `currentPlayer` is a local int that cycles after each turn.

In Multiplayer:

- Server sends "It's Player X's turn" to everyone.
- Only that player can roll dice, place pieces, and click "End Turn".
- Server enforces this — others are locked out.

Starting and Ending the Game

- Start: Only host can start. Clients wait in lobby.
- End: Server evaluates win conditions, broadcasts final scores and winner.

Summary : Comparison of current features and the updates needed for multiplayer support

Feature	Current	Multiplayer Changes
Game State	Stored locally in KiviGameplay	Needs to be shared via network
Turn Control	Local variable (currentPlayer)	Server-controlled and synced
UI	Local-only UI	Host vs. client UIs
Dice Rolls	Triggered by local button	Only enabled for current player, sent to server
Piece Placement	Direct on UI	Sent to server for validation
Game Start/End	Single user triggers	Host controls, clients follow
Communication	Not needed	Use sockets or any networking API