Intro:

In this tutorial we are going to learn how to turn our single player game into a multiplayer game.

Clip 1:

First so that we could differentiate between one game clients to another we have to get the player’s unique Id.  
 we do this by going to the tickTacToeMain class (pause)

Than storing T custom of CUSTOM\_INFO\_KEY\_myUserId in our myUserId parameter.

Clip 2:

After sending us the gotCustomInfo callback, the server will wait for the rival player to connect, and then he will send us a gotMatchStarted callback, if you remember that in the previous tutorial we initialized our game from the constructor, we now initialize it from the gotMatchStarted callback, this is made so that the players could play a couple of games consecutively.

Besides letting you know the game has started ,the gotMatchStarted callback transfers some key elements, all Player Ids which is an array containing all the users playing the game, finished Player Ids an array of all the players that left the game and server Entries which will be discussed later on.

Clip 3:

Remember in the first tutorial that we took each game move from the game logic, and instead of directly committing it, we sent it to the TickTacToeMain class and only then committed it, well now we are going to send it to the server instead of committing it at all, we do it by calling the doStoreState function, the doStoreState function has to get an Array of User Entry elements, so we take our game Move and place it in a User Entry by calling the static create function of the User Entry, with a key, which is the name by which the data we send will be known, and value which is the actual data we want to send.

Clip 4:

After one of the players sends any data using the doStoreState call, all users get a gotStateChanged callback containing the data the player sent, in our case we don’t need to look at the key because it is irrelevant and therefore we just check the value is from the type we expected, but in cases a user sends more than just one entry, and other cases which will not be discussed here, the key might be useful as well.

From this point on, any change to the game’s logic will only occur through changes sent to the server, and not in a direct way

Clip 5:

Now let’s go to the emulator and see what our changes have made (pause)

Notice that now any move happening in one player window reflected in both.