Making our single player game go multi

In this chapter we will:

1) Learn how to start the game

2) Learn how to send user input

3) Learn how to receive user input

Getting the players user Id

To get the player’s user Id we call T.custom(CUSTOM\_INFO\_KEY\_myUserId, null) again after or during the gotCustomInfo callback, just as we learned in the previous tutorial.

The user Id will help us to distinguish one game client from the other, and it will prove invaluable, during multiplayer game creation.

\* For more information about gotCustomInfo visit:

<http://code.google.com/p/multiplayer-api/wiki/gotCustomInfo>

<http://code.google.com/p/multiplayer-api/wiki/custom>

Starting the game

From this tuturial our game stops being a single player and is turned into a multiplayer game, the first change we make is instead of calling the startNewGame function from the constructor, we now call it from the gotMatchStarted callback, this is because we want our players to be able to play more than one game consecutively.

The server will call gotMatchStarted each time a game starts, allowing our users to play as many games as they want without reloading the game, and each time getting a fresh slate.

To do this we override the gotMatchStarted callback by calling:

**override** **public** **function** gotMatchStarted(allPlayerIds:Array,finishedPlayerIds:Array, serverEntries:Array):**void**

{

}

allPlayerIds – is an array of integers representing all the players user ids, all the players get the same array.

We will use the length of the allPlayerIds array as a parameter telling us the amount of players playing in our Ticktacktoe game.

And we will use its order to determine the turn order in turn based games (this is a suggestion, not a must)

\*for more information about gotMatchStarted visit: http://code.google.com/p/multiplayer-api/wiki/gotMatchStarted

Sending user input

1. Remember that all our input passes through the TickTacToeTuturialMain class before being applied to the game’s logic, well now instead of sending it right back to the logic through the makeTurn function, we will send the information to the server instead using the doStoreState function (which we get from extending the ClientGameAPI).
2. The doStoreState function gets an array containing UserEntry elements.  
   In our case we send one UserEntry representing the move our player made.
3. To make a new UserEntry we will call UserEntry.create(key:Object,value:Object);
4. We put the move positions in the key, and a class representing the player’s move in the value.

\* Every class that is sent to the server should extend SerializableClass, have a constructor with no arguments, and have all its values public.

\*All Serializable classes must be registered in the game’s constructor, for example, in our game we have TickTacToeMove which extends SerializableClass, and we call (new TickTacToeMove).register() in the game’s constructor.

Each time we store a value on the server all the players, including the player that stored the data, will receive a gotStateChanged callback that represents the changes made.

\*for more information about doStoreState visit: <http://code.google.com/p/multiplayer-api/wiki/doStoreState>

http://code.google.com/p/multiplayer-api/wiki/UserEntry

Receiving user input

Every time another player calls doStoreState, the server takes their call and sends a gotStateChanged callback to all the users participating in the game, the gotStateChanged callback will contain the data sent in the doStoreState call.

To retrieve this data we override the gotMatchStarted callback:

**override** **public** **function** gotStateChanged(serverEntries:Array):**void**

{

}

serverEntries – an array of ServerEntry elements each containing a key corresponding to the key the user calling the doStoreState used, and a value corresponding to the value stored.

When we get the gotStateChanged callback we check we got the class we expected in the ServerEntry value, and then send it to the makeTurn function.

\*for more information about gotStateChanged visit: <http://code.google.com/p/multiplayer-api/wiki/gotStateChanged>

http://code.google.com/p/multiplayer-api/wiki/ServerEntry