Multiplayer Design Plan

Think about what messages you need to send from the client to the server, and from the server to the client, to achieve your tasks. You should write your design plan **before** you attempt to start any coding, and then adjust it as necessary. Messages from the skeleton code have already been completed for you, and you **do not** need to change these.

# Socket.IO Messages Required:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Message is Sent From (client/server)*** | ***Message Name*** | ***When the Message is Sent*** | ***Data Sent & Description*** | ***What Happens when the Message is Received*** |
| server | maze data | When a client connects for the first time  **AND**  When a new maze is generated | **mazeSize**  An object containing the following members:   * rows – an integer describing how many rows there are in the maze * cols – an integer describing how many columns there are in the maze   **mazeCells**  A 2D array of objects, with the following members:   * x – the row number of the cell * y – the column number of the cell * top – Boolean, describes whether there is a wall at the top of a cell * bottom – Boolean, describes whether there is a wall at the bottom of a cell * left – Boolean, describes whether there is a wall at the left edge of a cell * right – Boolean, describes whether there is a wall at the right edge of a cell   **mazeStart**  An object containing the following members:   * x – an integer describing the row at which players start the maze * y – an integer describing the column at which players start the maze   **mazeEnd**   * x – an integer describing the row at which players end the maze * y – an integer describing the column at which players end the maze | The client should replace its existing maze information:  mazeCells replaces the *maze* variable, mazeStart replaces the *mazeStart* variable and mazeEnd replaces the *mazeEnd* variable, mazeSize.rows replaces *cellsWide* and mazeSize.cols replaces *cellsHigh*. |
| ***Message is Sent From (client/server)*** | ***Message Name*** | ***When the Message is Sent*** | ***Data Sent & Description*** | ***What Happens when the Message is Received*** |
| Server | New Player | When a server receives an information about new socket being connected to the client  **AND**  When a player changes its name  **OR**  colour  **AND**  When player makes a move | **players[socket.id]** – 2D array that stores information about connected player-sockets which each index is associated with socket ID and contains variables such as:   * **x** –Integer, stores player’s row number on the canvas cell * **y** – Integer, stores player’s column number on the canvas cell * **name** – String, stores player’s sprite name visible below the sprite image on canvas * **colour** – String, stores player’s sprite name colour visible below sprite image on canvas | The client should iterate through received “**players**” array in a for loop and by searching by “**ID**” extract each separate player object and assign it’s variables (x,y,name,color) to a new player object that is stored on client side array named “**player**”  Therefore, these newly created object/s on client side should be used to draw player images on canvas via **animate**() function. |
| Server | New Log Name | When a player changes its name  **OR**  colour | **Names** – it’s an object that stores following members:   * **nameOld** – String, stores name previously assigned to the player * **nameNew** - String, stores newly set player’s name | The client append the **#GameLog** div tag with a (**H1**) text message with “Player “**nameOld**” is now “**nameNew**”, and **removes first**() (previous) message to keep the log tidy. |
| Server | New Log Connected | When a server receives an information about new socket being connected to the client | **players[socket.id].name –** Variable in 2D array that stores information of current socket name. | The client append the **#GameLog** div tag with a (**H1**) text message with “Player “**players[socket.id].name**” has connected”, and **removes first**() (previous) message to keep the log tidy. |
| Server | Disconnected | When a socket disconnects from the server | **socket.id –** build in node, unique socket identifier assigned as random once a socket is connected to the server. | Client is updated with the **socket.id** that needs to be removed from client side array named **player[id]**, because one of the sockets (players) has been disconnected. Therefore, it needs to be removed from the game.  Player is no longer being drawn on the canvas. |
| Server | New Log Disconnected | When a socket disconnects from the server | **players[socket.id].name –** Variable in 2D array that stores information of current socket name. | The client append the **#GameLog** div tag with a (**H1**) text message with “Player “**players[socket.id].name**” has disconnected”, and **removes first**() (previous) message to keep the log tidy. |
| Client | New Player | When new socket connects to the client | None. The purpose of this message is to inform server that new socket has been connected to the client and is ready to receive new player data. | Server receives a very polite information from the client:  “Hey a new socket just connected to me, could you do me a favour and create a new object on your side for me, so I know where to draw him on the canvas, please?”  Therefore, client crates a new player object in 2d array named “**players[socket.id]**” on server side and assign default starting values for a newly created player:  **X** = 0,  **Y** = 0,  **name** = nameID (socket.id)  and  **color** = defaultColor(#ff0000)(red) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Client | key | When player makes a move using a keyboard  **OR**  mouse button  **OR**  finger direction swipe on canvas | On keypress:   * **event.key** – “**event**” stores information about any key that is being pressed by the user. “**key**” selector looks for the specific key name. In this case “W”,”S”,”A”,”D”.   On mouse click:   * **this.id** – “**this**” is referring to button that is currently being pressed by the player and “**id**” selector looks for “**this**” specific button “**id**” name.   On finger swipe:   * **swipe** – String, stores specific letter that is assigned to directional movement on the server side. Letter is assigned upon recognising direction of swipe made on the canvas.   In this case:  “w” – moving up  “s” – moving down  “a” – moving left  “d” – moving right |  |
| Client | newName | When player change its name  **OR**  colour | * **name –** String, stores name that user has inputted into text field on html page. * **playerColor -**  String, stores colour that user has selected from the colour picker on html page. |  |
| Server | maze data | When one of the players reaches finish line (cell 9,9) | **getMazeData()** – As listed at the beginning row of this table it passes variables such as: **mazeSize** , **mazeCells**, **mazeStart**, **mazeEnd.** |  |
|  |  |  |  |  |
|  |  |  |  |  |