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 | Maze data  Cells | When the client connects for the first time, it will draw the maze cells for them . | **cellsWide;**  A 2D array of objects, with the following members:   * wide – the width of the cell   **cellsHigh;**  A 2D array of objects, with the following members:   * y – the height of the cell | The client should replace its existing maze information:  cellsWide replaces the *maze* variable, cellsHigh replaces the *cell variable*. It will then draw the maze cells. |
| Server | Mouse controls. | When a client connects for the first time it gives a message on the server, which the controls would send their input to the server, through standard or ajax posts. | **MouseX;**   * Move the player side to side with the mouse curser. If it reaches the end it will stop.   **MouseY;**   * Moving the player up and down on the maze. If it reaches the end it will stop.   **PlayerX:**   * Setting the width position of the player   **PlayerY:**   * Setting the height position of the player. | The client should replace its existing maze information with the mouse controls, in order to move the player around.  PlayerX and PlayerY are used to create the position of the player when moved.  I |
| Server | Maze data  Socket.io | When the client connects for the first time. It will show a socket.io message on the server-side saying player has connected and a maze has been generated. | **socket:**  This message allows the client to connect to the server using; http://localhost:8081. | The client should replace its existing maze information.  The client will then be able to run the game on the server using port number 8081, by simply calling the server.js file in node.js. |
| Server | Keyboard controls | When a client connects for the first time it gives a message on the server, for the keyboard event. | **case 65:**  **case 37:**   * This would move the player to the left on the maze.   **case 87:**  **case 38:**   * This would move the player up on the maze.   **case 68:**  **case 39:**   * This would move the player to the right on the maze.   **case 83:**  **case 40:**   * This would move the player down on the maze.   **PlayerX:**   * Setting the width position of the player   **PlayerY:**   * Setting the height position of the player.   **PlayerSize:**   * Creating the size of the player when moved. | The client should replace its existing maze information with the keyboard events, in order to move the player around.  PlayerX and PlayerY are used to create the position of the player when moved. PlayerSize would set the size of the player when moved. |
| Client | Touch button events | When a client connects for the first time it will show the buttons on the client side. | **Up:**   * Moving the player up clicking/touching the up button.   **Down:**   * Moving the player down clicking/touching the down button.   **Right:**  Moving the player to the right clicking/touching the right button.  **left:**  Moving the player to the left clicking/touching the left button. | The client should replace its existing maze information with the touch and click events, in order to move the player around. |
| Sever | Players | When a client connects for the first time, on connection add to players. | **Player[]:**  The players should have an ID, X coord and Y coord.  **Socket.emit:**  When sending to server.  Send to the sender.  Socket.broadcast.emit – send to all but sender | The client should replace its existing maze information with the multiplayer and on connect players[] players[]. For each in players[] create sprite for other players in the canvas. |
| Server | Dealing with movement for the players | When a client connects for the first time, on connection move the player. | Players[].ID = player.ID;  Players[].x = player.x;  Players[].y = player.y;  Socket.emit.(players[ID]);  Sending the players data on the server using socket.emit(player data). | The client should replace its existing maze information with the player and then send the players data ID,X and Y. |

*Add as many extra rows as you need…*