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.

For each message you think you need to achieve working multiplayer, please complete the table with the following information:

* **Message is Sent from (client/server)**
  + Write ‘server’ if this is a message sent from the server to the client
  + Write ‘client’ if this is a message sent from the client to the server
* **Message Name**
  + What is your message called? Try to give your messages sensible, but short, names – like variables.
* **When the Message is Sent**
  + What triggers your message to be sent? Is it when a specific event happens? Is it when something happens in your game? Is your message sent when several different things happen? Describe them all (briefly) in this column.
* **Data Sent & Description**
  + What data/information might you need to send with this message, and what format does it take. Are you sending an object? What variables are included in that object? Do you only need to send a variable? What will these be called? Where does the information come from? What assumptions have you made, or what do they represent? What types of data can they carry?
* **What Happens when the Message is Received**
  + What do you need to do in the event handler for this message? If your message is sent from the server to the client, remember that your event handler will be on the client (and vice-versa). Does your message make something happen? Does it mean that you need to do something with some of your variables? Briefly describe that here.

An example message, already included in the skeleton code, has been completed for you on the next page. Study this carefully (and compare it to the skeleton code) and use it to help you complete the rest of the table.

Think carefully about the messages you will need, and refer to ‘*Real-time Servers V - Server Design: From Start to Finish*’ for more guidance on planning the design of client-server architecture.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Message is Sent From (client/server)*** | ***Message Name*** | ***When the Message is Sent*** | ***Data Sent & Description*** | ***What Happens when the Message is Received*** |
| server | dungeon data | When a client connects for the first time  **AND**  When a new dungeon is generated | **dungeon**  An object containing the following members:   * maze – a 2D array of integers representing the dungeon layout. 0s represent impassable spaces (walls), 1s represent corridors, and numbers 2 or greater represent rooms. * h – the height of the dungeon (size in the y dimension) * w – the width of the dungeon (size in the x dimension) * rooms – an array of objects, describing the rooms in the dungeon. Each object in this array contains the following members:   + id – an integer representing this room in the dungeon, numbered by order of creation   + h – the height of the room (size in the y dimension)   + w – the width of the room (size in the x dimension)   + x – the x-coordinate of the top-left corner of the room   + y – the y-coordinate of the top-left corner of the room   + cx – the x-coordinate of the centre of the room   + cy – the y-coordinate of the centre of the room * roomsize – the average size of the rooms, used when making the dungeon * \_lastRoomId – the id of the next room to be generated   **startingPoint**  An object containing the following members:   * x – the x-coordinate at which players should start in this dungeon * y – the y-coordinate at which players should start in this dungeon   **endingPoint**  An object containing the following members:   * x – the x-coordinate at which players can escape this dungeon * y – the y-coordinate at which players can escape this dungeon | The client should replace its existing maze information:  dungeon replaces the *dungeon* variable, *startingPoint* replaces the *dungeonStart* variable and endingPoint replaces the *dungeonEnd* variable. |
| Server | player\_on\_client | When the player connects | A message is emitted containing the players server array   * players - a object containing the players details   + I - the players horizontal position   + J - the players vertical position   + Id - the players socket.id | The array will be used to tell the players there initial i and j position along with the id which is stored in the player\_client\_array. |
| Client | Update Server | When the directional arrow keys are pressed  or  When the directional buttons are clicked  or  When the player pans in a certain direction  or  When the player reaches the end of the dungeon | The client sends the player client array to the server which contains:   * Player objects   + I - the players horizontal position   + J - the players vertical position   + Id - the players socket id | Sends the updated I and J positions from the client which has moved to the server. It writes these new positions onto the server array. |
| Server | Update position | When the Update server function is called | The server sends a array to the client which contains   * Player objects   + I - the players horizontal position   + J - the players vertical position   + Id - the players socket id# | Allows other clients to see the player has moved. |
| Server | clientside\_id | When a new player connects | The server sends a variable   * player.id - contains the socket id of the latest player to connect to the server | This is used to determine which player is which while making decisions for player movements. |
| Client | Dungeon Reset | When the players I and J are equal to the dungeons ending x and y positions | Boolean   * Emits a message to the server   + Sends a message which sets a boolean input to true   The player client array   * Player objects   + I - the players horizontal position   + J - the players vertical position   + id - the players socket id | Essentially when the player reaches the end of the dungeon. It makes the server generate a new dungeon. It also makes the servers x and y position set to the start point making the client go to the start of the new dungeon. |