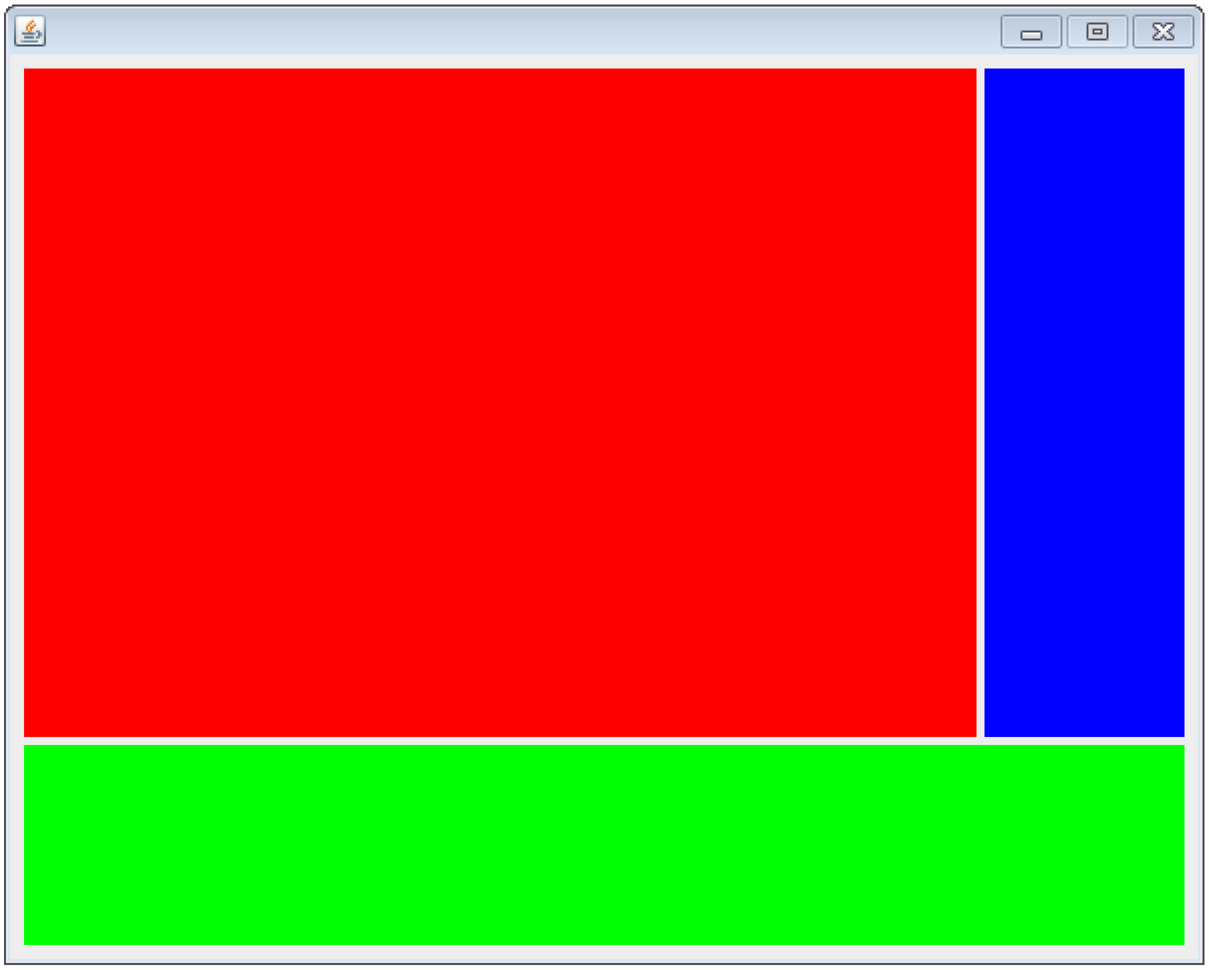
Running Technical Specification (Quoridor - Clientside)


Proposed Client Layout

Client The program entry point. Instantiates a new GUI allowing the user to start a game of Quoridor.

GameManager The state machine responsible for game setup, turn execution and game resolution. Utilizes the **State** interface, implemented by all states within the machine.

GameState Stores the overall state of the board: all game pieces, possible moves, shortest paths, etc. Utilizes the **GamePiece**, an interface for game pieces such as the pawns and walls; the **Wall**, representing a wall in a game of Quoridor; the **Pawn**, representing a playing in a game of Quoridor.

Network For now this represents *something* that will be polled for strings and sent move encodings, initially will be a connection to a dummy server with user input or Netcat. However, an outgoing connection should still be used for testing rather than direct user input. All input and output will be displayed within the **Console**, represented by the above image in green.

Board The GUI for displaying the game state, represented by the above image in red. Implemented as a JPanel that takes in a set of GamePieces.

Supplemental Information Represented by the above image in blue, this area will be used to display quick information that would be a hassle to find in the console.

The State Machine The state machine is currently made up of three major portions:

- **Init -** Generates *n* pawns in the GamePiece set, where *n* is the number of players, centered at each cardinal direction, establishes connections to servers then transfers control to the turn loop state.
- **Turn Loop -** For each pawn query for move, check legality, if legal transmit move to all other servers, else give server goodbye message, notify other servers and remove reference from the game. Once move is committed check for win, if win transfer control to the win state.
- **Win -** Notify all servers of player win, after notification sever connection with server. Display player win dialog and clean up the game leaving the user at the main menu.

Transition Logic The transitions between states will be executed with boolean flags. When init executes it returns true if execution was successful, false otherwise. If the value returned was true it transitions; otherwise it severs all made connections, cleans up resources and drops the player at the menu. On the turn loop, execution returns true if a player has won, false otherwise; if true transition to win state, if false transition back to turn loop state. On the win state no transition is neccessary.