Project Design and Documentation

Programmers: Bosco, Liam, Nihal
Project: Super Mario Brawl Stars Game
Mr. Cadawas (ICS 4U1 Final Culminating Performance Task)
Friday, June 14, 2024.

Client's Requirements

Here are the client, Mr. Cadawas' requirements as communicated in our May 27, 2024 conference. Please be advised that the following requirements also embody the "Common Requirements" criteria stipulated under the "ICS 4U1 CPT Rubric".

Data Files:

There will be 2 levels maximum, each with their own .csv files. Data will be formatted such that values are separated by commas.

Data (of terrain tiles) will be held as character values (ex: 'a' for "air", 'b' for "brick", 'g' for "grass", etc…).

Networking:

The game must have a multiplayer option where 2 players can join simultaneously. The networking encryption "language" will distinguish between chat messages, game messages, movement, etc…

GUI:

- Overall window (JFrame) must be exactly 1280 x 720 pixels.
- The above will be the one and only window; though there can be more than one panel.
- Dimensions: Each map will be 20 units high x 100 units across
- Animation & Chat: At any given moment of gameplay, users should have access to a Chat panel and an animation panel as follows.
- The animation panel will assume the role of a dynamic viewport (dimensions 20 units x 20 units) that follows the user's movement.
- A variety of JComponents will be used to enhance user experience (ex. JButtons, JTextFields, JTextAreas, etc…)
- Images will be original unless otherwise consulted with Mr. Cadawas.

Characters:

A maximum of 3 characters is permitted. Characters can vary in their appearance and/or their physics/abilities ~ up to programmers' discretion.

Coding Paradigms:

Code files are to be separated by Model-Control (Math, Logic, SSM) and View (GUI, JComponents).

Optional Features:

- Users jump on a pole at the end of the race to determine the winner.

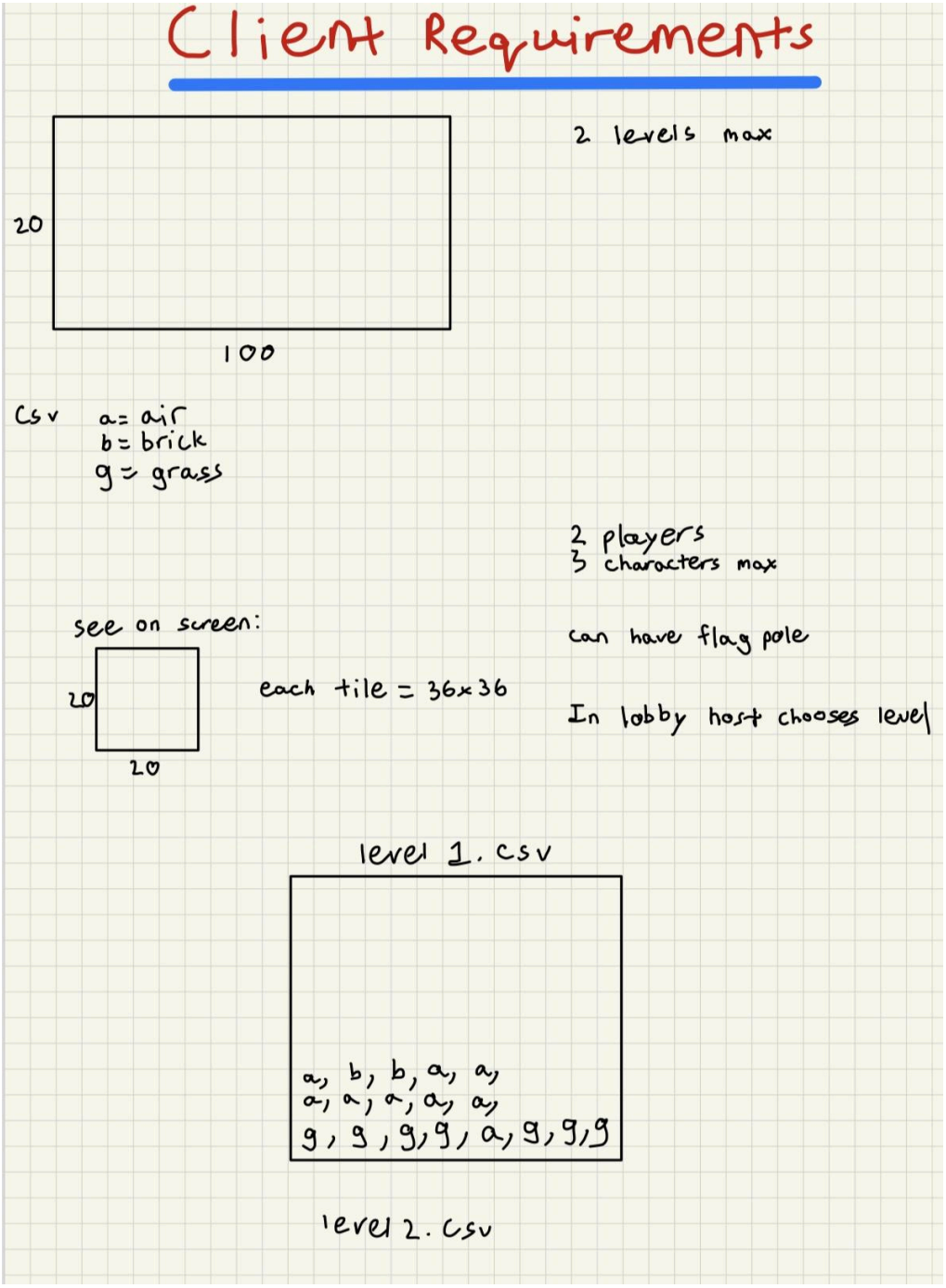


Figure 1
Quick Notes from Client Conference Specifying Our Program-Specific Requirements (w/ Diagrams)

UX Designs

Main Menu Panel

Every user starts at this panel whether they first join the game or if they return back from a map. There are three key interactive elements; all of which are JButtons for the user to navigate through the different modes or settings of the game; each with its own panel.

- Play Button: Used by the host to select map and move to character selection screen
- Connect Button: Used to either connect as a host or connect to a host as a client
- Tutorial(Demo) Button: Used to practice the controls and get a feel for the gameplay

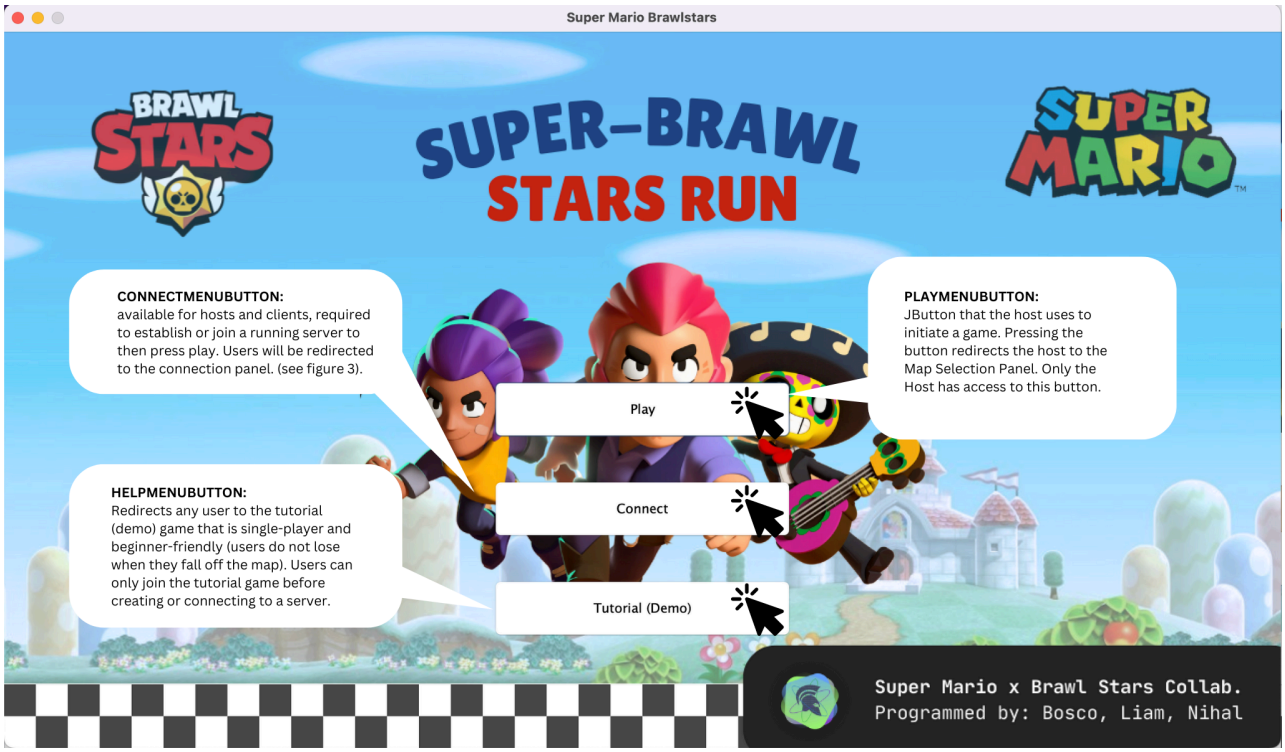


Figure 2

Main Menu Panel

Connection Menu Panel

Before starting any “formal” (multi-player) game, a player must first establish a connection with a new (as a host) or existing server (as a client). After pressing the ConnectMenuButton from the main menu, users are redirected to this page (see Figure 3a & Figure 3b).

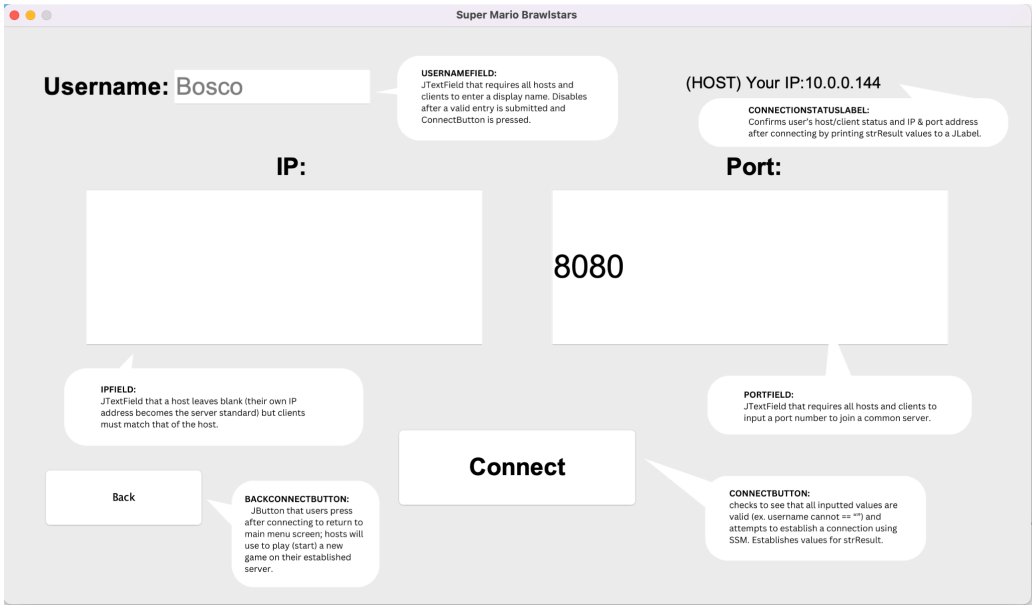


Figure 3a

Connection Menu Panel (Host POV)

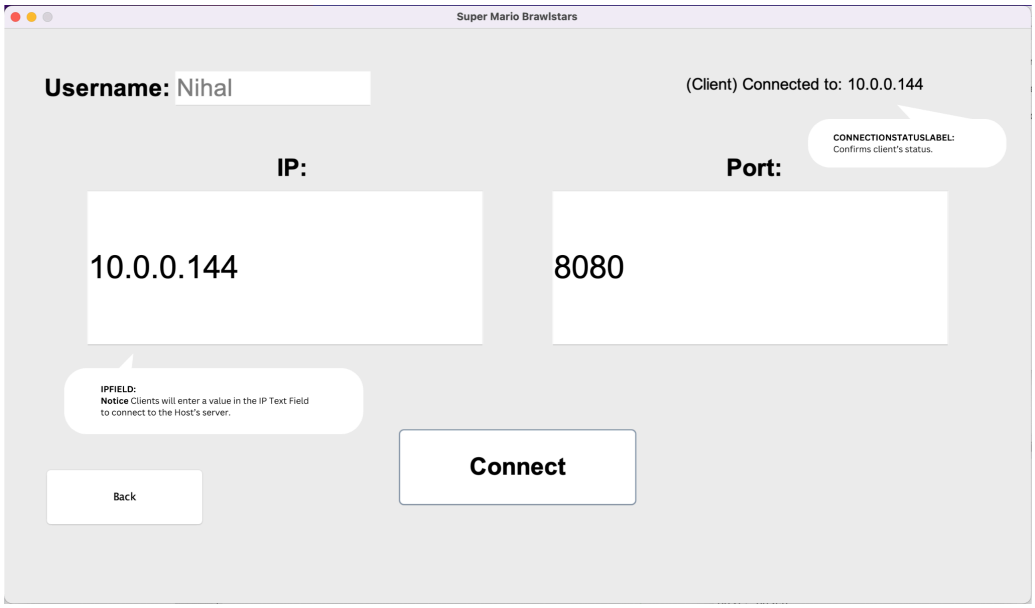


Figure 3b

Connection Menu Panel (Client POV)

Tutorial (Demo Level)

Unconnected users will be able to press the “Tutorial (Demo)” button (see Figure 2) and be automatically directed to the Game/Animation Split panel with Colt selected as the character and Map3.csv as the data for the level terrain. Most interactive elements such as chat input text fields will be disabled for users in this stage.

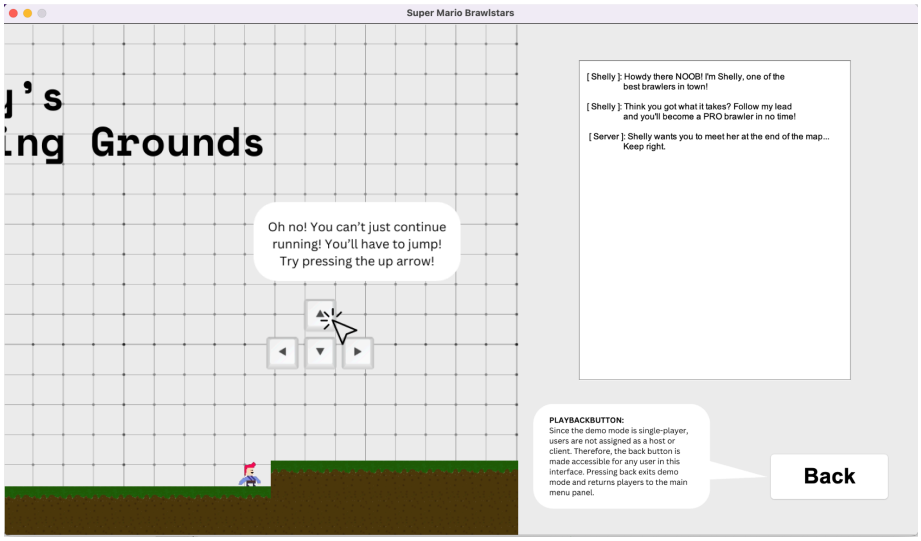


Figure 4

Tutorial/Demo Play Screen

Map Selection Panel (Host Only)

Once the host presses the PlayMenuButton, the host will be redirected to the map selection panel. Here, the host must select one of two levels.

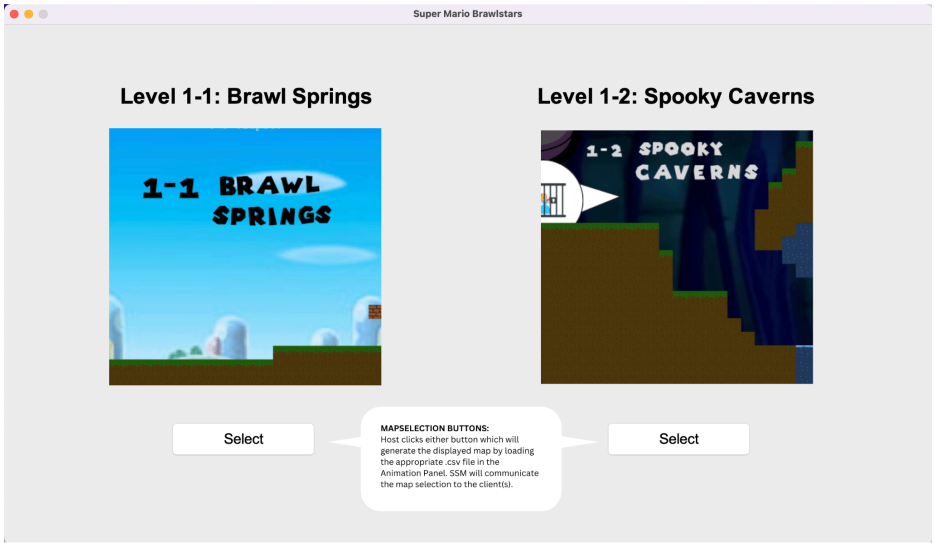
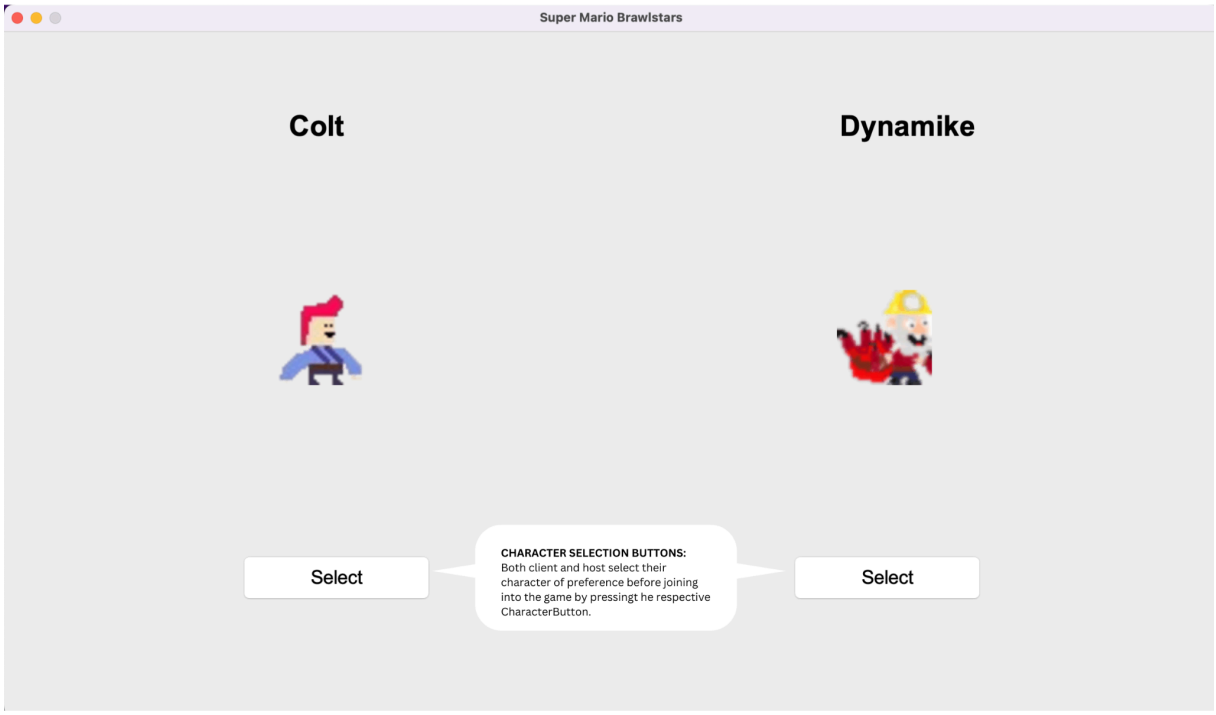


Figure 5

Map Selection Panel

Character Selection Panel

Once the host has selected the map, BOTH the host and the client will then be redirected to the character selection panel to select their character of preference.



Play Split Pane

A special pane occupies the main JFrame during game-time. This pane divides the panel in two parts ~ the Animation Panel (view.AniPanel) and the Chat Panel (view.ChatPanel). The Animation Panel is a child class extension of the regular JPanel which repaints according to a specified timer (at 60 fps to be precise…) to refresh the pixelated graphics shown on the screen which includes the map terrain, SSM-communicated movements and status of foreign players, viewport positioning, character positioning, etc… The chat panel consists of a text field to input user messages and displays a live stream of message outputs from other players on the same server.

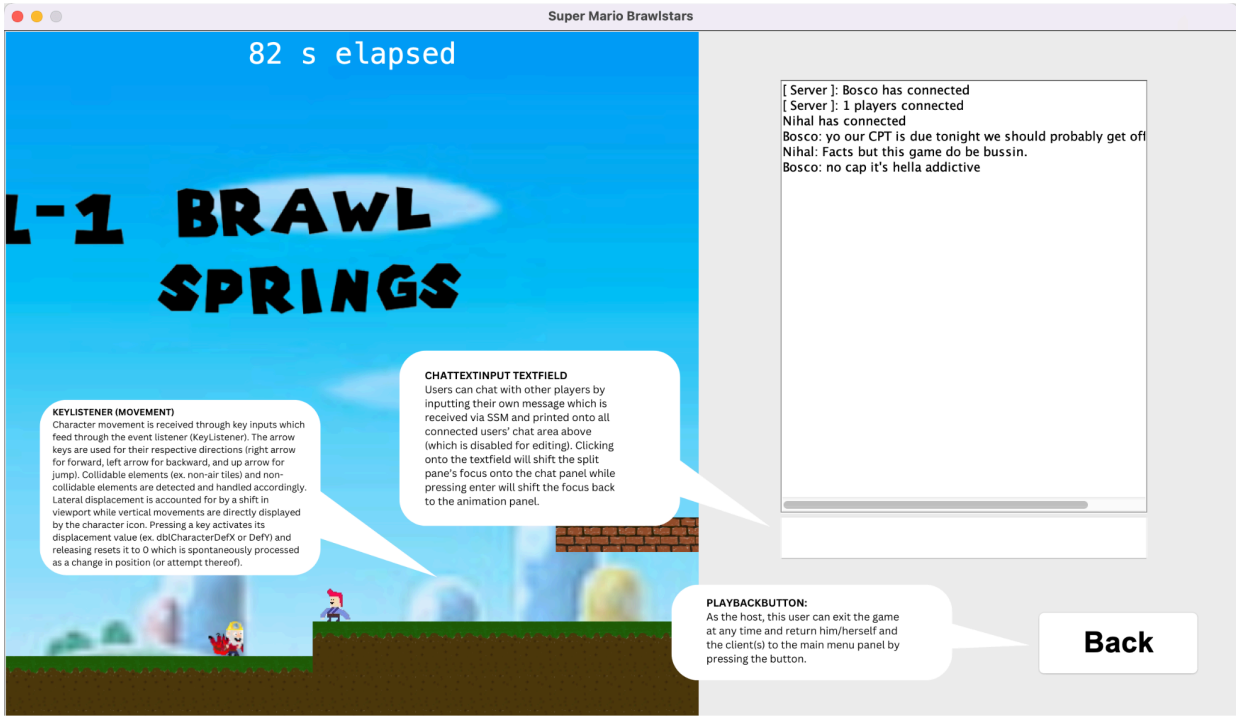


Figure 7a

Play Split Pane (Host POV)

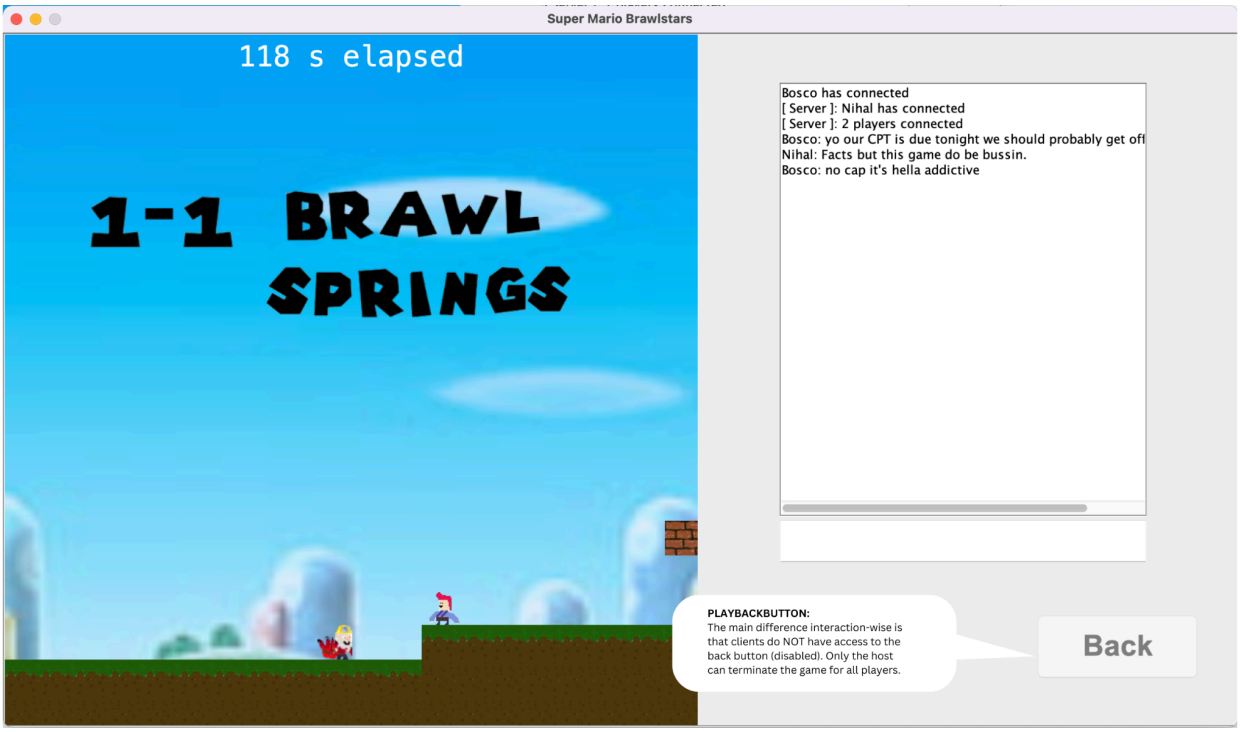


Figure 7b

Play Split Pane (Client POV)

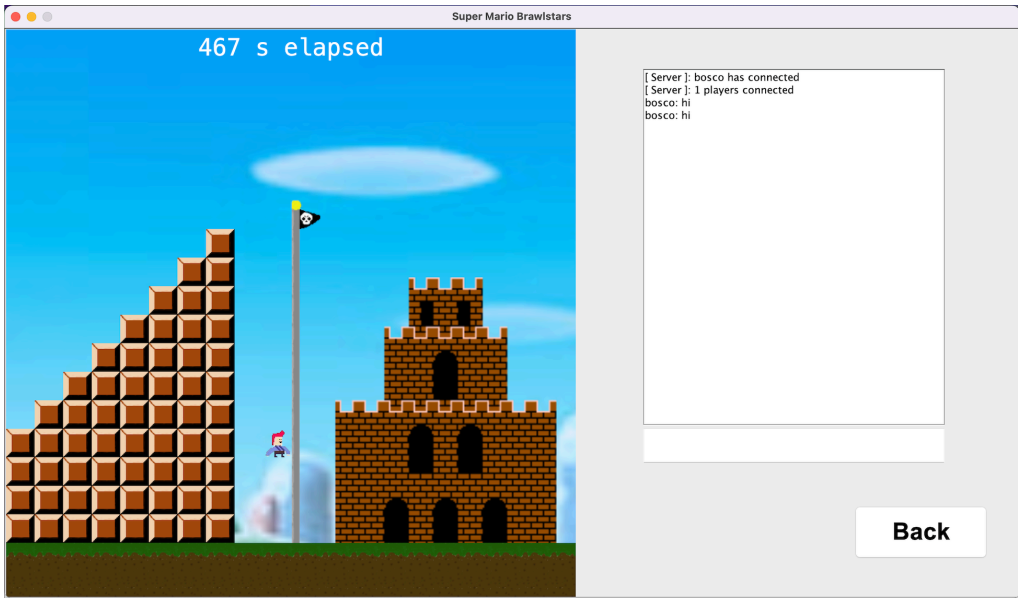


Figure 7c
Play Split Pane (Objective POV)

Ultimately, this is a race. The objective for each player is to be the first to reach the bottom of the pole at the right end of the map. Reaching that X coordinate disables all movement and prepares the win screen.

Win Panel

The only way to get to this screen… is to reach the bottom of the flag pole… first! The chat will also process an SSM message that announces the winner and the time it took them.

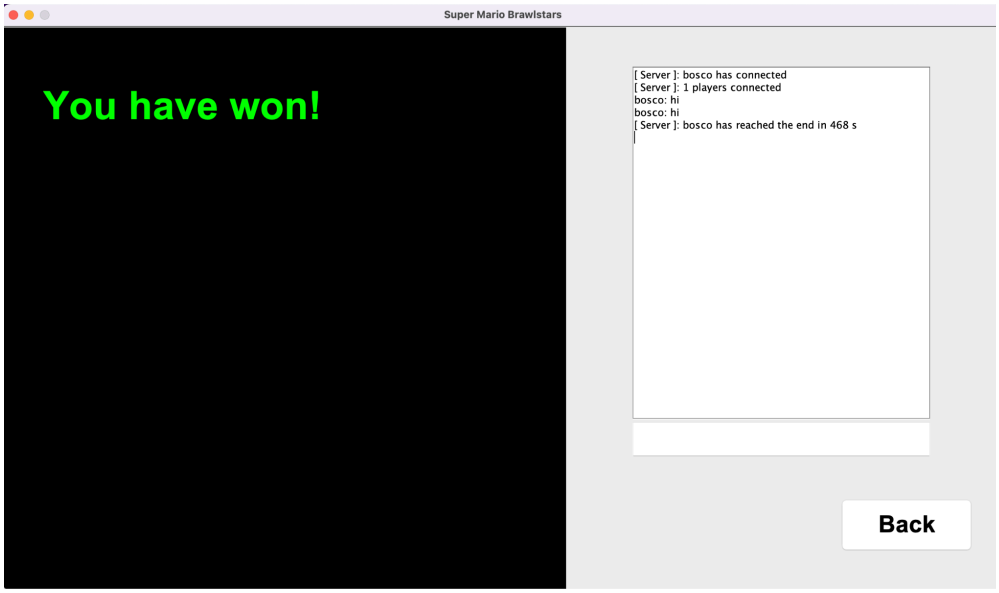


Figure 8a
Win Panel (Winner's POV)

Lose Panel

There are 2 ways to lose (deaths also announced):

- Fall into the void (fall off the map)
- Get beaten by another player to the bottom of the flag pole first

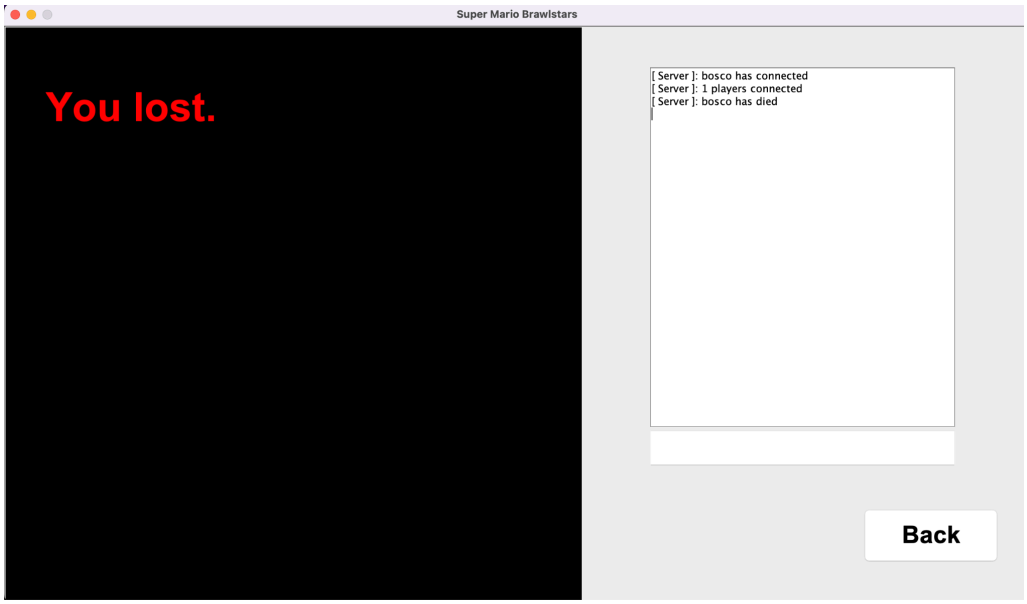


Figure 8b

Lose Panel (Loser's POV)

Network Message Design

Below is a brief breakdown of the encryption formats used to mutate information between users through a network using socket programming concepts.

-ssmMessage[0].equals(“character”)
When detected: After the host selects the map, it would send the character message allowing the client to pick their desired character.
Ex: “character”

-ssmMessage[0].equals(“chat”)
“chat*,str(Host/Client)Username,view.ChatTextInput.getText()*”
When detected: After any player sends a text it would display in the chat area the message with the username.
Ex: “chat,bob,hello”
Ex: “chat,joe,bye”

-ssmMessage[0].equals(“reset”)
When detected: When the host presses the reset button it would send the reset message forcing the client into the main menu and restarting the game.
Ex: “reset”

-ssmMessage[0].equals(“server”)
“server,death/win,*strplayerUsername,intRaceTime*”
When detected:
- Once the message is sent it would display in chat area if a player has died or won
- If a win is detected, it would show the opponent the “lose” screen while displaying in chat the time it took the winner to reach the pole.

Ex: “server,win,bob,17”
Ex: “server,death,joe”

-ssmMessage[0].equals(“connection”)
“connection,*strUsername*”
When detected: As players connect to the game it would update the number of players playing and display in the chat area that the user has connected.
Ex: “connection,bob”
Ex: “connection,joe”

-ssmMessage[0].equals(“position”)
”position,view.*dblCharacterX*,view.*dblCharacterY*,(*directionofcharacter ~ right/left)*”
When detected: After every movement, it would send this message to update the opponent's character to match the new position of the movement and the direction the character is facing. Works to send and receive. Character updates position to opponent(s) and opponent(s) update position to character.

Ex: “position,10.0,10.0,right”
Ex: “position,3.0,5.0,left”

-ssmMessage[0].equals(“Map”)
“Map,*mapchoiceinteger(1 or 2)*”
When detected: As only the host can click on the chosen map, the message would be sent so the client can load the same map chosen by the host.
Ex: “Map,1”
Ex: “Map,2”

-ssmMessage[0].equals(“characterChosen”)
“characterChosen,*characterchoiceinteger(1 or 2)*”
When detected: After selecting a character it would send this message to update the opponent’s game to display the character chosen by them. This allows both the client and host to see the same characters each selected.
Ex: “characterChosen,1”
Ex: “characterChosen,2”

-ssmMessage[0].equals(“EndY”)
“EndY, *y position of bottom of pole*”
When detected: As only the host can select the map, it would send a message updating the client the position of the bottom of the pole since each map has a different height.
Ex: “EndY,612”
Ex: “EndY,504”