CITS3002 Project Report

Jono Jones (22479345)

2021

Tier 1 - Enabling Multiple Players

Make it open to two players

For this project I used the select approach rather than going into threads. Threads can lead to confusing memory errors so if there is a module that will minimize that like select.py then I will take that.

I have organised my project into two main parts, Hermes and the GameMaster. Hermes will cover the server and sending/receiving type roles while the game master will be in charge of the game logic.

For clients to play in the same game together the server must:

1. Maintain a game board
2. Welcome the player to let them know their player id.
3. Introduce the players to each other by letting everyone see which players are in the game
4. Manage the player tiles by sending them an initial hand and replacing any tile they place.
5. Determine and broadcast whose turn it is. The first round of turns the players need two turns in a row (to place their token) but after that the players only need one turn. -> implemented in GameMaster's **next_turn()**
6. Process any incoming messages from the client (if its their turn) and take appropriate action with the game board.
7. Update all other clients about the state of the board and along with any eliminations or disconnections.

To achieve this Hermes has two functions **send_to(msg,connection)** which sends a message to a specific client (needed for point 2. and 4.) and **send_all(msg)** (for 3., 5. and 7.) which sends the message to every connection in select's input list (and in Tier 2 everyone in the silent list)

Use of select

Inspiration was taken from this article and the lectures.

Tier 2 - Scaling Up

When game finishes, restart

To restart the game correctly multiple things need to happen.

1. The game needs to identify that it has ended (less than one player left) -> implemented in GameMaster's **is_finished()** when determining whose turn it is next.
2. The game needs to clean up, ie make all of the remaining players into users so that we can pick from the pool of users randomly to make the new game and reset the gameboard and clear player_order
3. Let each client know that each player has left. -> implemented in GameMaster's **clean_up()**
4. Send out the countdown and start the countdown. (I was challenged by this.. Hermes now looks inside every message sent to everyone and if its a countdown will use time.sleep for the countdown time.)
5. The game can restart by calling start game again.

Allowing the server to handle up to 4 players

Random selection of clients

Keeping clients updated

Tier 3 - Connection Issues

Scenario 1

Scenario 2

Tier 4 - Player Issues

Bring a late client back up to date

What to do for a slow client