Reflection - Karan Rakesh

I believe most of the decisions I took regarding my game engine design paid off handsomely at the end, where it was pretty straightforward to build a different game using the same constructs used to build the 2D platformer that we made over the course. In my opinion, I was very successful at designing for reuse, as shown by my code. The code for my Snake game, had 25 different sections of code from the 2D platformer of which there were :

- Changes pertaining to declaring/calling/drawing a text on losing the game.
    - I decided since there was a definite end in this game, aka hitting yourself or hitting a boundary, to use a text that signified the game was over rather than just shutting the game down abruptly which exposed me to the sf::Text class and sf::Font class which were similar to the other SFML objects to implement.
- A vector of platforms and the related collision boxes and logic to build the border for the snake game.
    - In the initial game, I used 2 platforms and 2 moving platforms for achieving the specified requirements. In the snake game, I didn't need the moving platforms, so rather than declaring 2 more static platforms, I put all the borders into a vector Platform and carried out drawing and collision detection using these objects.
- The Character became the food that the snake had to eat.
    - Since the character was a single object in my initial game design and all the other connected clients were stored in a separate vector. I decided to convert the single object into the food which the snake had to eat to grow. I called a function to get a new position of the food once the snake head had collided (eaten) the food which the server computed and sent back.
- The vector of Clients became the snake.
    - Since the initial implementation had a vector of clients to store all the connected clients, I used this to implement the snake since the snake needed to grow as the user ate more food. Hence it made sense to use a dynamic sized data structure like a vector.
- The processing was done on the client side with minimal server requirements.
    - I shifted all the snake position computation to the client side as I didn't want to send the entire snake coordinates over to the server to conduct collision detection and compute movement. So I only performed my food repositioning using the server, where the server calculated a random value within the bounds of the game and sent it to the client. I also tried only requesting for the food repositioning when required and otherwise not sending msgs at all, but it caused some lag in the response so I started sending null messages to just keep the server ticking and ensure timely updates of the food position.
- The game doesn't support multiplayer.
    - Since I used the vector of clients for the snake, there is no mechanism to store the positions of other clients that connect to the server. Additionally, the clients do not send their coordinates to the server so it is unaware of the position of the

clients at any point in time, hence it is not possible for it to broadcast it to the other clients.

I kept the variable names similar to try to maximise the similarity between the two codes and kept in any redundant calculations occurring in the game loop that didn't affect the snake game in as well to maximise similarity. So I was able to achieve around 50-100 out of 800 lines of different code in the client.cpp (>80% similar) despite completely altering the game mechanics and doing all processing on client-side versus server side. That was my opinion on the changes that I had to make from the final 2D platformer to the snake game.

Regarding the changes to my overall game engine design through the homework assigned throughout the semester I feel I overall did a good job in designing an engine that was able to adapt to most of the requirements required of us with not too much code refactoring. One particular point that I wish I had considered earlier was building a game engine using Threads and doing the computations on the server-side since this would have reduced the load in the intermediate assignments where a lot of my time was spent just refactoring the code to make it compatible with the outlined requirements. There were obstacles along the way that needed lots of debugging and understanding of concepts to solve, but that made the solving of these problems feel even better. Overall, I felt a lot of the design decisions I took turned out to make my life easier for e.g. from the outset I designed my game to be compatible with an arbitrary number of clients and allowed them to join at any time during the game. This decision helped me solve one of the sections in a future assignment without making any changes. Hence overall I was satisfied with my design decisions over the span of this course, I was able to learn a lot over the course of the last few months.