CS4344 Networked and Mobile Gaming
AY14/15 Semester 2
Group 6
Elemental Frenzy

Team members

Name	Matriculation Number	Email
Chang Yan Qian	A0098998R	A0098998@u.nus.edu
Chue Sai Hou	A0097812X	A0097812@u.nus.edu
Lim Yu De	A0105829B	A0105829@u.nus.edu
Yip Jiajie	A0101924R	A0101924@u.nus.edu

Contents

1. Introduction

In this project, our team developed a real-time multiplayer game coded solely with HTML5 and Javascript. Our game, Elemental Frenzy, runs on Chrome for both the client and server. Moreover, our game is also portable on mobile Chrome.

In this report, we will first talk about the game mechanics and features of our game. We will then discuss how we implement multi-playability for the game and the type of communication model that we used. Finally, we will cover the techniques used for various situations in multiplayer gameplay.

1.1 Splitting of Workload

Generally, Yan Qian and Yu De worked mainly on the logical and networking aspect of the game on issues such as players joining a game session, multiple game sessions, and short-circuiting, local perception filter and server-side prediction to reduce the lag perceived by the players. Sai Hou and Jiajie worked on the beautification of the game, which includes aspects such as the user interface (UI), sprite sheets, map design, and the heads-up display (HUD) where the health points (HP), mana points (MP) and power-ups remaining duration and other player attributes are displayed.

Although it may be the case that Yan Qian and Yu De mainly handling the implementation for networking issues, whenever there would be a major decision to be made for the networking side of things, every member of the team would chip in ideas and come to a conclusion as to what the best networking technique to solve a problem was, and as such everybody learned together and contributed equally.

2. Game Design

Elemental Frenzy is a 2D side-scrolling platform player-vs-player (PvP) battle arena game. 4 players battle it out in a free-for-all, death-match style mode by shooting elemental balls (eleballs for short) at each other. Players can navigate around the map to either run away from the danger or to scout for powerups and potions. The player with the highest kills (tie-breakers are resolved by least number of deaths) wins the round when time expires.

2.1 Library used

We made use of the open source 2D game engine library named Quintus. Quintus provides many functionalities for our game such as game physics, the handling and creation of game objects, animating the sprites, map creation, collision detection and it even handles the playing of sound. However, given the vast capabilities of this library, it does not support multi-playability and that essentially is our main job scope to implement it. Further details about Quintus can be found in their website: http://www.html5quintus.com/

2.2 Type of communication model

For the most part of the game, the permissible-server-client communication model was adopted with short-circuiting on the client-side. Clients collect and then send events to the server which in turn simulates those events and then updates other players.

However, our clients are not directly connected to the server due to the limitations of Quintus. Instead, the clients and server are connected via a middle-man server (app.js) which holds the information of all the game sessions being hosted by the server. The information includes only the player count, maximum players, connected players and map created (by the server) for each server hosted. These are presented in the game lobby which we will talk about it in the later section.

2.3 Synchronizing states among players

By using such a permissible-server-client communication model where the server holds the one true state for all players connected to a game session, states among players are easily synchronized.

When a client receives an update from the server and finds that its state is different from the server, it will perform linear convergence to synchronize its state. We also decided that a client should be authoritative about its movement so as to ensure the smoothest experience for the players (with the server checking to ensure no cheating). This is done so because if the server were to tightly synchronize with the players' movements, it may result in jittery and teleportation of the players.

As such, the server will also perform linear convergence when it finds that its state for the player's position or velocity is out of sync with the player's state. This is the only time that the state on the server side follows the state on the client side because we aim to reduce visual disruption due to convergence for the players.

2.4 Strategies to reduce bandwidth

Due to the nature of our game being similar to that of a FPS, it is necessary for updates to be sent frequently from clients to server and vice-versa to ensure that the states between them are closely synchronized and to ensure game play is not disrupted. However, we managed to reduce the number of update packets to only five per second. Update packets from the clients only contains the action taken as well as the current state of the player while packets from the server contains the decision and current states of other clients.

3. Features

In this section, we will talk about the features of Elemental Frenzy.

3.1 Server

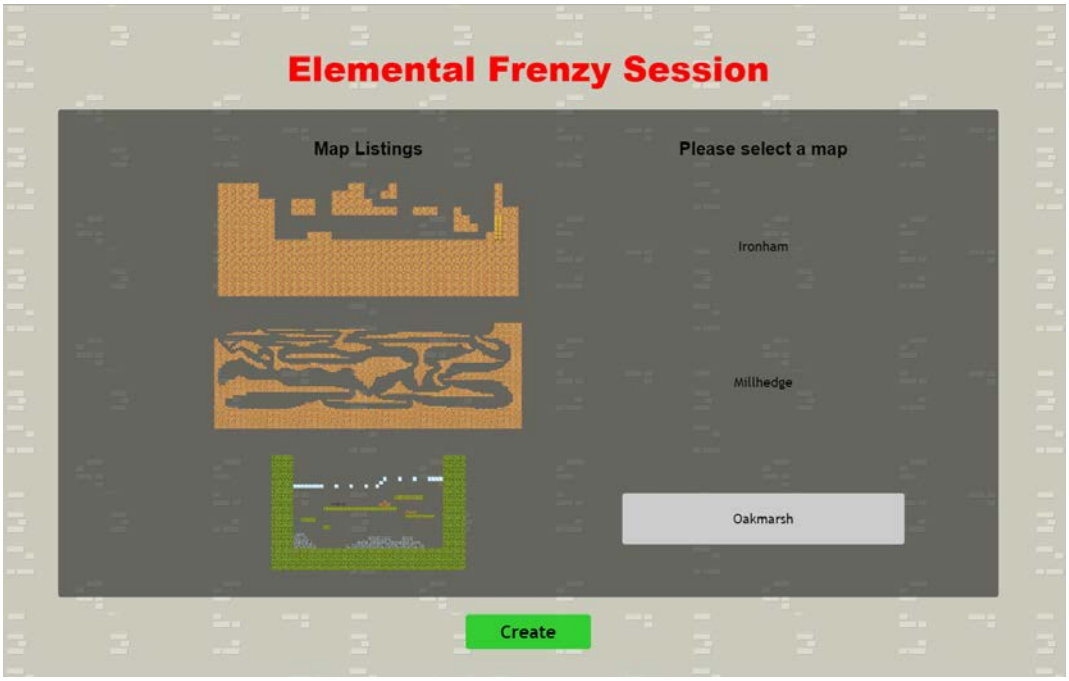

Figure 1: Map selection

Once connected as the server, you will be brought to the map selection page as shown in *Figure 1*. Over here, there is a variety of maps of different sizes and layouts for the host to choose from. After creating the map, you will be brought to the server's view as shown in *Figure 2*.

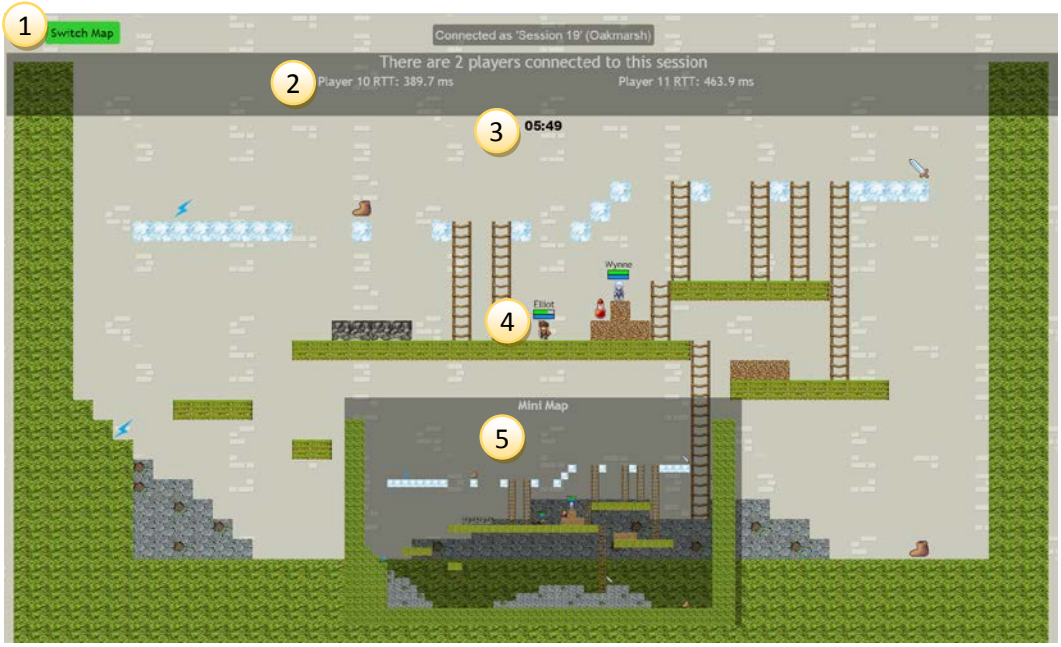

Figure 2: Server's view

In the server's view (*Figure 2*), you can notice several things (numbered):

1. Switch Map
    - As a host, you can choose to switch maps
2. Connected players' RTT
    - Players' RTT is displayed here
3. Round timer
    - The timer shows the remaining time for that round
4. See all players / focus on one player
    - By using the arrow keys (up, down, left , right), you can move the view around the map to see the situations
    - By pressing 'F' key (follow), you can choose to toggle the focus view among connected players. Also, you can press 'G' key (un-follow) to stop focusing on the player.
5. Mini Map
    - A mini map is useful to track players in bigger maps

3.2 Client

When the server is done setting up, players can now connect to the game via the game lobby (app.js) as shown in *Figure 3.*

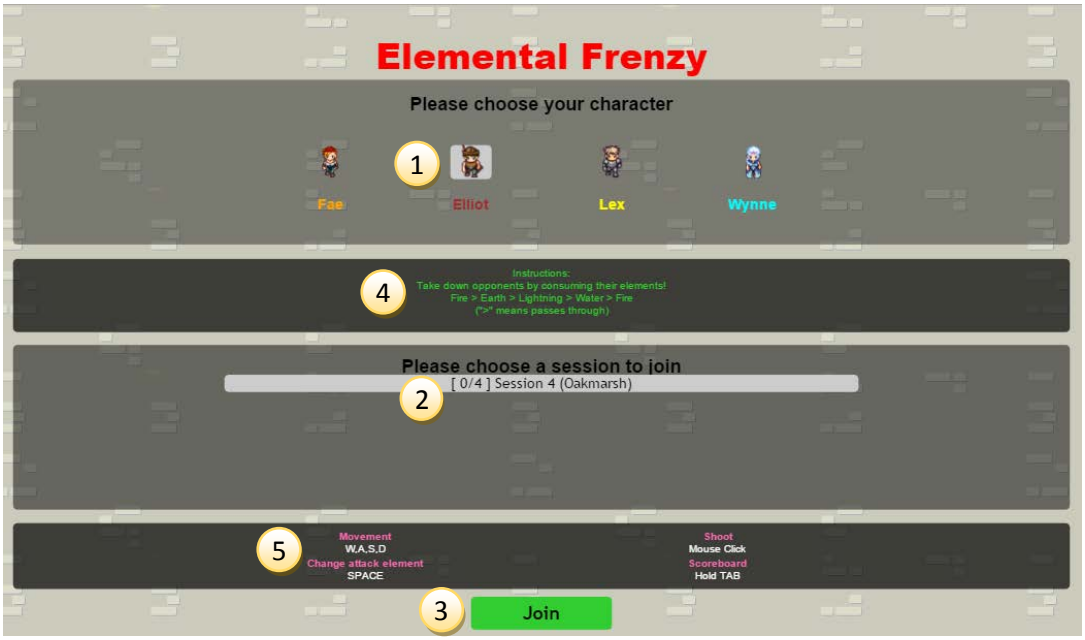

Figure 3: Game lobby

In the game lobby, you can notice the following (numbered):

1. Choose character
    - Choose character out of the four choices
    - However, when a character is already in used in the selected session, it will be unavailable for the client to choose
2. Choose game session
    - Supposed there are many servers (limited to five), players can choose which game session to join
    - Servers are capped at five to ensure maximum quality of game play

- Map created by each server is shown in the selection section as well

3. Join game
    - Once player has chosen his/her character and game session, he/she can now join the game
    - Server processes player's join request by packet arrival time
    - If the session is ending in 15 seconds or is full, the server rejects player's join request
4. Elemental advantage/weakness explanation
    - There are four elements that each player can choose from – Fire, Earth, Lightning and Water, and they can change their attack element at any time in the game
    - If element A has an advantage over element B, then the eleball (element ball object fired by player in short) consumes and passes through the eleball of element B
    - The list of elemental advantage is given as Fire > Earth > Lightning> Water > Fire, where ">" means "consumes" or "passes through"
5. Game controls instructions
    - 'W,A,S,D' movements
        - o 'W' causes the character to jump or climb up the ladders.
        - o 'A' causes the character to move left
        - o 'S' causes the character to climb down the ladders
        - o 'D' causes the character to move right
    - Space
        - o Allows player to toggle among the four elements
    - Mouse click
        - o Shoots eleball in the direction of the mouse click from the player's character
    - Tab
        - o Shows the scoreboard

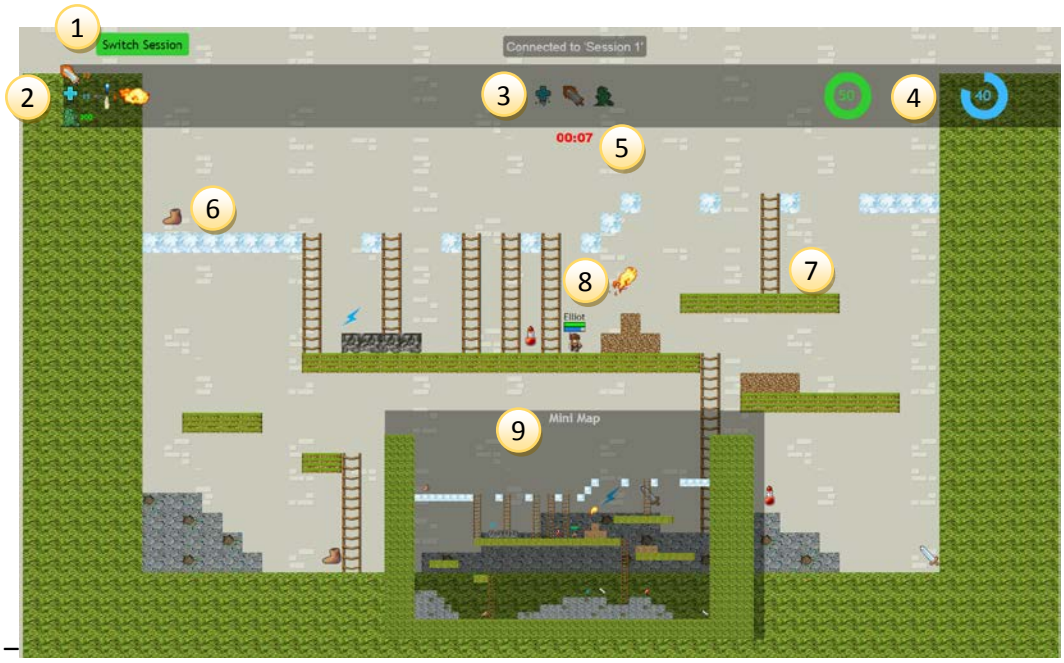

Figure 4: Player' view

After joining the game, the player can see the following in *Figure 4* (numbered):

1. Switch Session
    - After joining a game session, players can choose to switch to other game session
2. Status HUD
    - Displays the current attack power, mana cost per shot, movement speed and current element that the player is firing
3. Power-ups timer
    - Power-up icons will be lighted up when a player collected the power up and a duration bar will be shown to the player
4. Health Points (HP) and Mana Points (MP)
    - Indicates current HP and MP of the player
5. Round timer
    - The timer shows the remaining time for the round
6. Power-ups
    - Spawns randomly in the map
    - Sword: 50% damage boost
    - Boots: 50% move speed boost
    - Lightning: -70% MP cost for shooting
    - Bottles: Heals 30% of max HP
7. Ladder
    - Spawns randomly to add a “kick” of freshness to the maps
8. Eleballs
9. Mini map

And at the end of each game session, there will be an overall display to show how many times you have died and how many players you have killed. Moreover, it also allows has the option for you to play the game again. These are apparent in *Figure 5*.

Figure 5: End of game

3.3 Mobile platform

Elemental Frenzy is restricted to Chrome browser for mobile devices. Moreover, our game is capable of handling touch events to control the character. We also made use of the mobile 'shake' sensor to replace the 'Space' key for the toggling of the type of elements. Images of mobile platform can be seen from *Figure 6* and *Figure 7*.

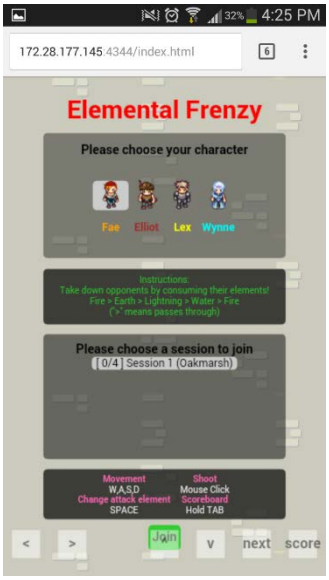

Figure 6: Running on Android Chrome

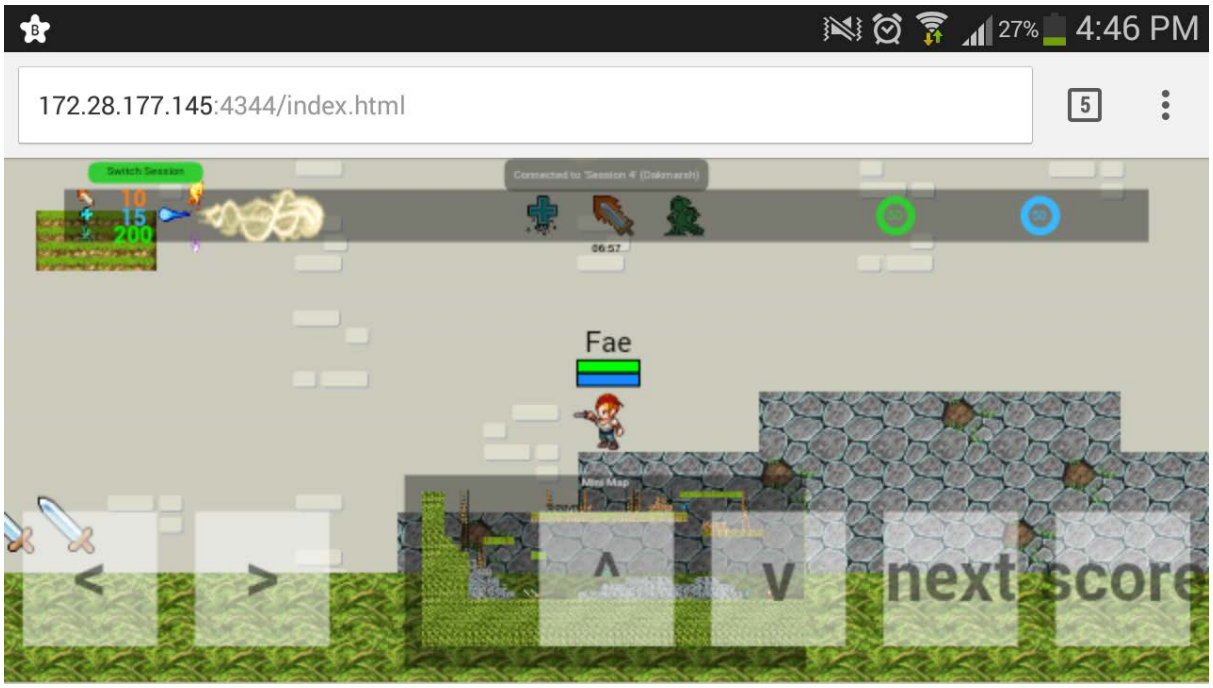

Figure 7: Running on Android Chrome

3.4 Assets and artwork references

We got most of our artwork and assets such as sprites, animation, sound and map tiles from Google Image. We made sure that all of these assets are royalty free before we use them in Elemental Frenzy.

4. Implementation

For a multiplayer real-time game, there are bound to be latencies which may cause disruption to the game play of the game. In this section, we will discuss about the different techniques adopted for various situations to mitigate the effect of latencies in Elemental Frenzy.

4.1 Short-circuiting

To reduce the perceived delay and thus improve the gameplay experience for the players, short-circuiting is implemented such that the client is 'smart' and will move the player's sprite immediately after collecting the 'move' event from the player, and then send the event to the server. The client does not wait for a response from the server before moving the player's sprite.

4.2 Artificial delay and Server-side prediction

Since the server - and not client - simulates the firing of eleballs upon receiving a player's mouse click event, when there is a considerable amount of latency between the client and the server, the lag can manifest very obviously to the player and reduce the game's playability and enjoy-ability. As such, we decided that the best strategy to overcome this was to use artificial delay on the client-side, where the player's sprite would play a shooting animation that would last for about half a second.

With the artificial delay introduced, the server receives the player's mouse click event about mid-way, or earlier, through the player's shooting animation. The server then performs prediction using the round-trip time (RTT) which it maintains for each connected player to decide when it needs to send the 'fire' message - which tells the clients to create the eleball - so that the player would see the eleball created just as the animation ends (illustrated in Figure 6 below in Section 4.3).

After implementing the artificial delay and the server-side prediction, we noticed that the lag of firing an eleball had almost completely vanished for one-way delay latencies of up to 200ms, showing the effect that animation and networking techniques have in 'tricking' the human mind. We have also included sound effects for firing eleballs to further mitigate the effects of high latencies from the players.

4.3 Local Perception Filter (LPF)

Unfortunately, while the animation in combination with artificial delay and server-side prediction did indeed reduce the perceived lag when firing eleballs, it was still very unplayable with one-way latencies of more than 100ms where eleballs become very hard to dodge. With just the techniques mentioned thus far, the player would jump over an eleball completely and still get hit by it because the server decides whether the eleball hits a player or not. In our architecture, the player does not get to decide.

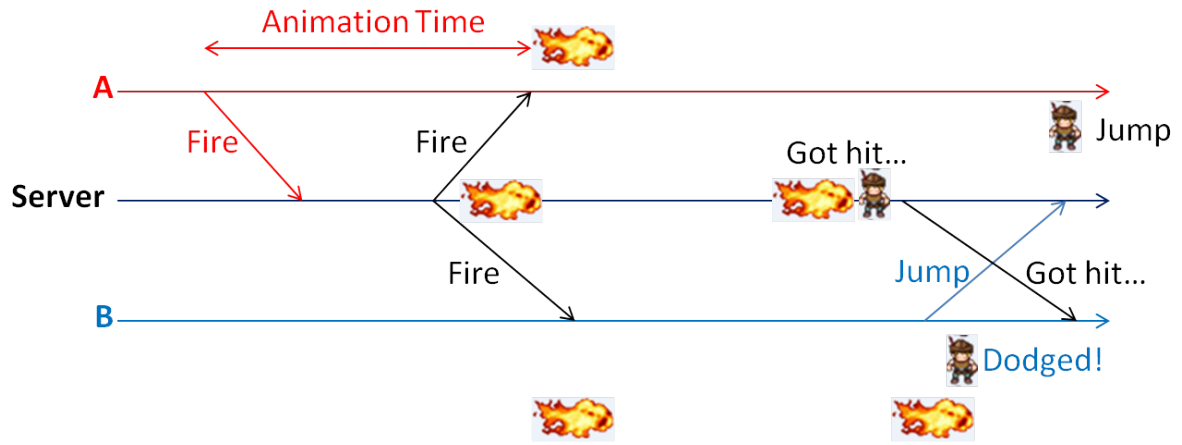

Figure 6: Without LPF: Server does not see the eleball-dodging player dodge the eleball

Figure 6 above shows the problem. Here, the issue was to synchronize what the player dodging the eleballs sees and what the server sees so that when the player sees himself dodge an eleball the server should see it too. For this, implementation of a LPF such that the scenario in *Figure 7* below is achieved was required.

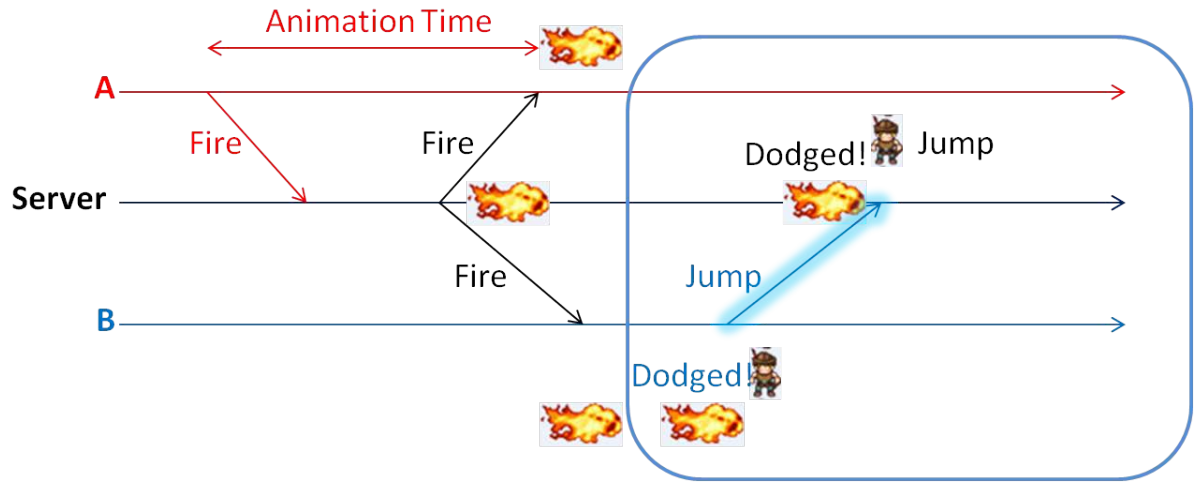

Figure 7: Server sees the eleball-dodging player actually dodge the eleball

In order to implement a LPF such that the server would see what the eleball-dodging player sees as illustrated in *Figure 7* above, the player B should be 'reading into the future' of the fired eleball so that it can send the 'jump' message at the right time so that the server simulates the jump at the right time. In this case, when the 'fire' message is received by player B, the eleball will be speed-boosted so that the 'reading into the future' can be done by player B to dodge eleballs smoothly. A short arbitrary value of 0.5s of time during which the eleball will be speed-boosted was chosen for our game as it is very important to the gameplay that eleballs can be dodged intuitively. The distance to be speed-boosted is simply calculated as (RTT * eleball velocity).

4.4 RTT calculation and Time stamping

In order to implement many of the solutions and networking techniques mentioned above, the server had to know the RTT to its players and the players to their server. To achieve this, the following techniques were employed:

- Timestamp synchronization by the clients with the server to synchronize their clocks with the server's. The time difference is calculated using the standard Network Time Protocol (NTP)
- Timestamp appending onto each message before sending out on both the client and the server sides.
- RTT calculation on both client and server sides. Formula used to update the RTT every time a new sample is calculated with the formula: **RTT = (α · Old_RTT) + ((1 – α) · New_Round_Trip_Sample)** with α = 0.9 to make new RTT samples affect the new RTT less.

4.5 Possible latencies capabilities

The game is currently playable even at high one-way latencies of up to about 200ms, and even up to 300ms, but naturally visual disruption and perceivable lag increases alongside the increased latencies.

5. Conclusion

All in all, from the development of Elemental Frenzy, our team has learned to implement the techniques taught in class albeit all the difficulties that we faced. Nonetheless, it was a great learning experience to all of us. We got to learn many different aspects of the game, from the aesthetics of the game to the back end development.

5.1 Future works

Currently, due to the limitations of the 2D game engine library, Quintus, that we used for the game, clients are not directly connected to the server. We sought to remove the dependency of the middle-man between the clients and the server so that we can give the game more scalability.