

CS4344 Networked and Mobile Gaming AY14/15 Semester 2

Group 6

Elemental Frenzy

Team members

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

# 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 situation in multiplayer gameplay.

## 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.

# 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.

## 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/>

## 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.

## 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.

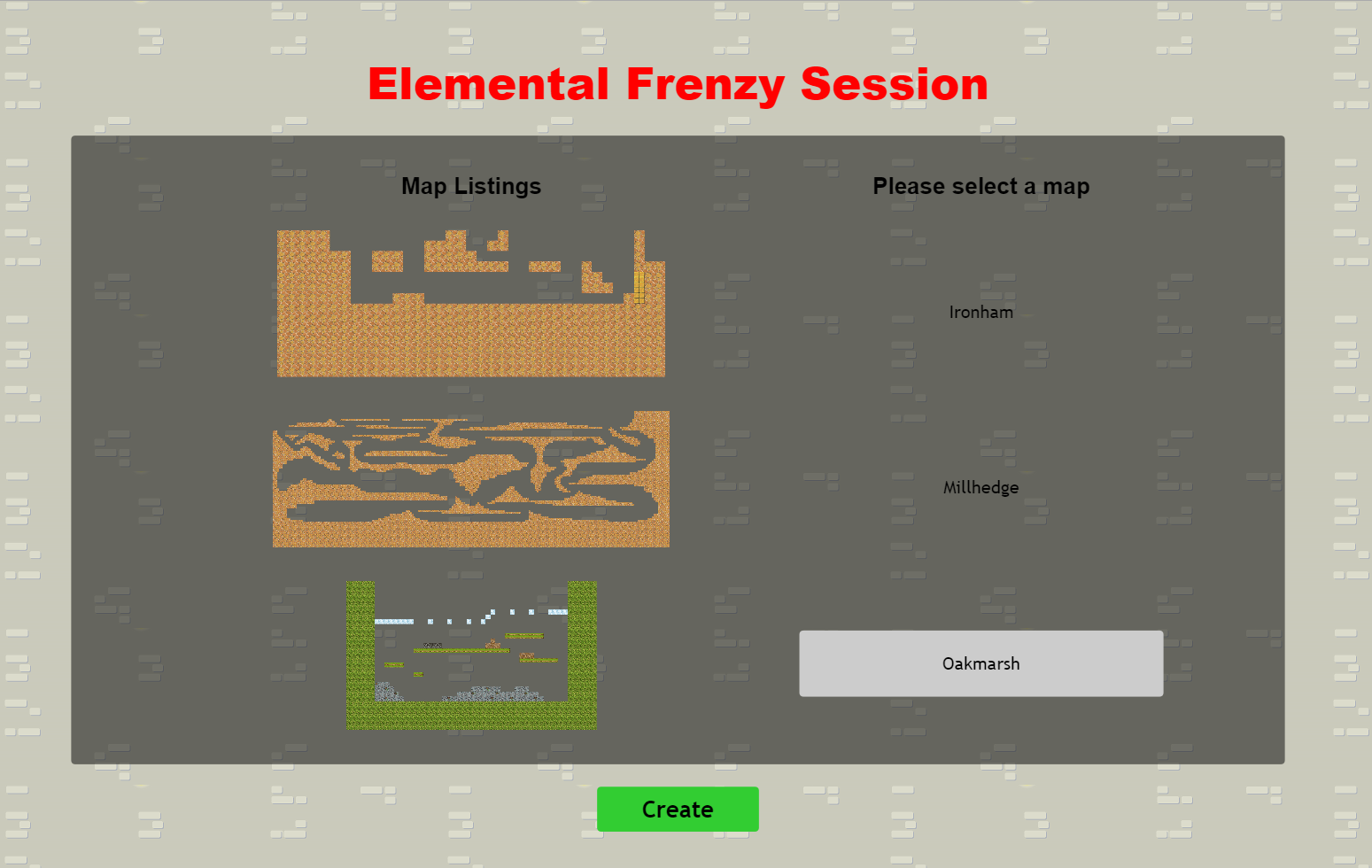
## Strategies to reduce bandwidth

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.

# Features

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

## Server



*Figure 1*

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*.

**

4

3

2

1

*Figure 2*

In the server’s view, you can notice several things (numbered):

1. Switch Map
   * As a host, you can choose to switch maps
2. Timer
   * The timer shows the remaining time for that round
3. See all players / focus on one player
   * By using the arrow keys, you can traverse around the map to see the situations
   * By click J/K keys, you can choose to toggle to focus view on a player
4. Mini Map
   * A mini map is useful to track players in bigger maps

## Client

* Game mechanics
* Splitting of workload
* Assets and artwork references
* Real-time
* Type of architecture
* Type of communication model
* How we synchronize states among players
* Strategies to reduce bandwith / power usage of the game
* How we ensure fairness

# Implementation

* Implemented in HTML5/Javascript
* Supports multiple game session capped at 5 due to network restrictions
* Game lobby which allows selection => player match making
* Techniques adopted for various situation
  + LPF for element ball
  + Short circuiting
* Smart client
* Library used
* Possible latencies that game still capable of handling
* Capable to be run on mobile platform, shake to change elements
* Problems faced and how we adopted techniques to mitigate issues