SUMO CLICKER

CI328 Internet Game Design and Development Assignment 2

Table of Contents

Introduction

In the following report, I will be detailing the process undertaken to complete the Sumo clicker project. This will include a summary of the game. This will be covering its rules, objectives, and gameplay. I will also give context to this gameplay using screenshots and captions.

Next, I will be covering how implementation was covered across the project's development and the flow of the code. This will have its own dedicated section to the creation of the network code was implemented and how interactions work between two players.

Finally, I will be reviewing my projects and detail into the best and worst aspects of the project and cover three areas that the project could be improved upon as to assess means of improving in both this project and also future projects.

Game summary

In Sumo clicker, you have a single objective: Push your opponent off the arena before they do. While on the surface this seems incredibly basic, there are layers that transform the game into more of a strategic battle. The player has only moving and blocking at their disposal which seem limited but both moving and blocking cost stamina. Stamina is a limited resource that does replenish but at delayed intervals making you have to consider when to act as if you run out stamina your actions will cost health and not only does this resource not replenish, if your health reaches 0 you lose the match anyway.

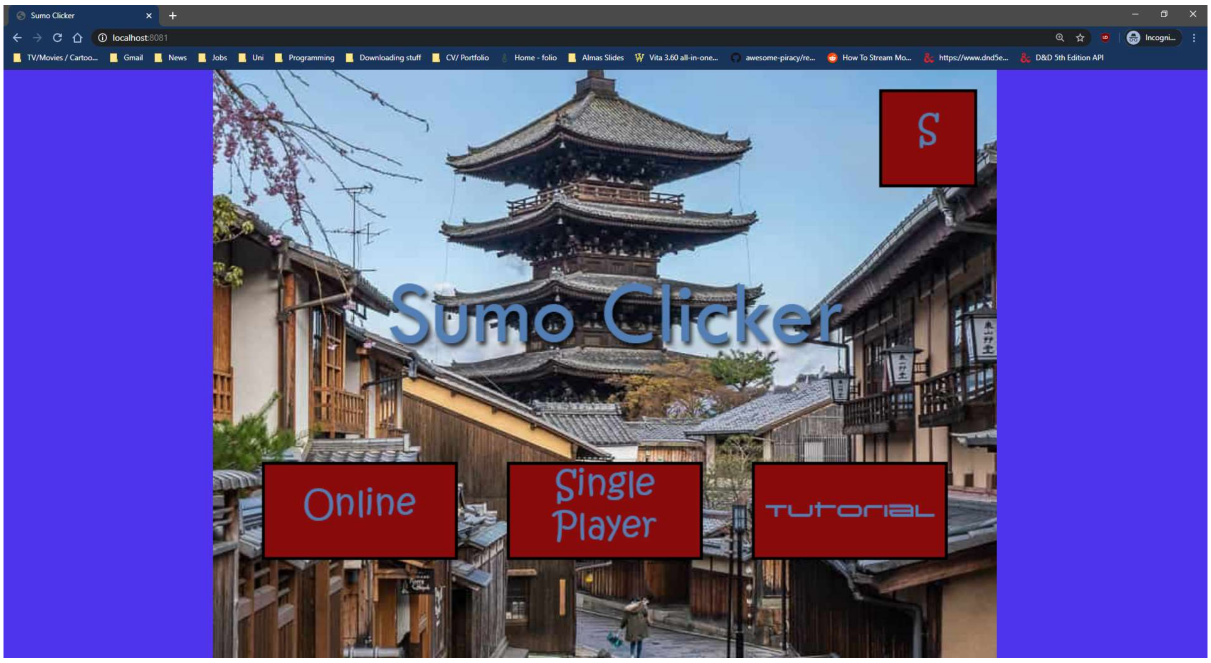
In Sumo clicker there is a single player mode that allows you to try out premade characters against a randomly selected character from the roster. Games against the AI can be difficult, especially on the harder difficulties as the stages get smaller and the ai reacts a lot faster the harder the difficulty, ensuring that the player will not only have to think but act fast to win on the hardest difficulty and opponents. While the single player offers training and experience with the core mechanisms of the game within a safe environment it offers little reward overall, which is where the multiplayer comes in.

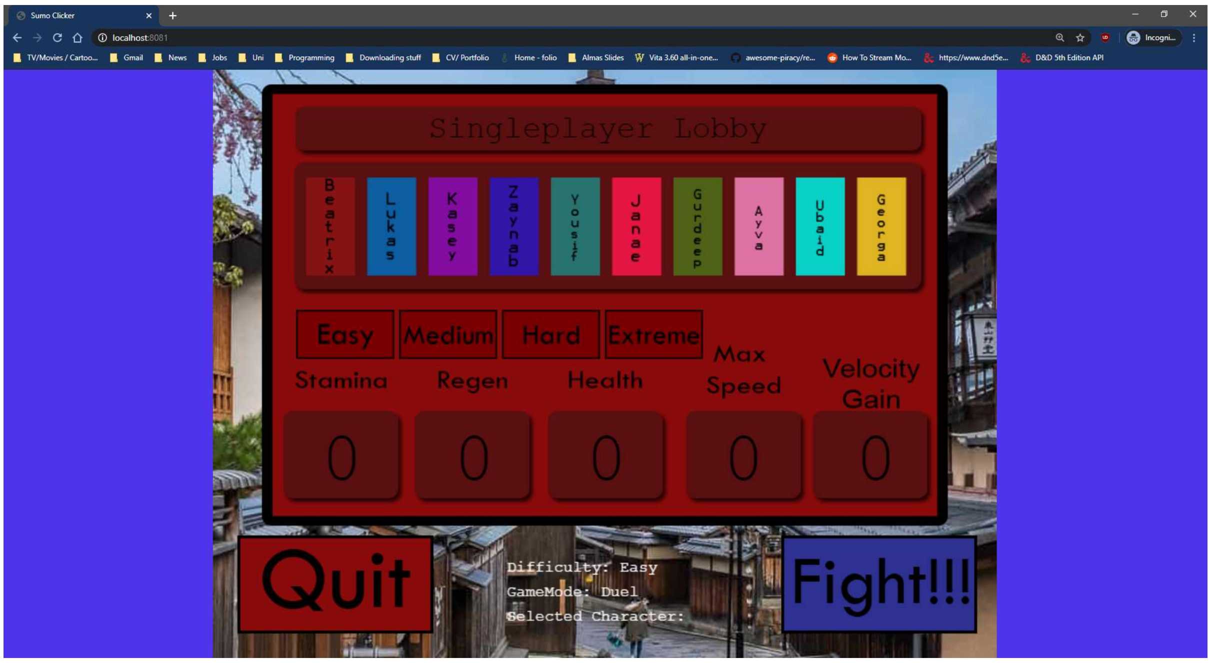
Within the online game mode, players must compete with other players to win matches so they can earn gold. The only use for this gold is to upgrade the players own character, which will persist between sessions and matches. This allows for the player to slowly increase the power of your own characters pushing ability. This even extends to the difficulty aspect of the game offering increased bonuses for wining the harder difficulties and even winning consecutively.

However, there is also a rogue-like aspect to this mode. Traditionally if you lost your health you would simply lose the match but within the Online mode losing all your health has a worse consequence of forcing a reset of your character back to the base level. The idea of this mechanic is to force the players to accept possibly losing on purpose to avoid losing their own character. This also opens the possibility of strategies where players aim to manipulate a player into Losing as to avoid their risk of losing.

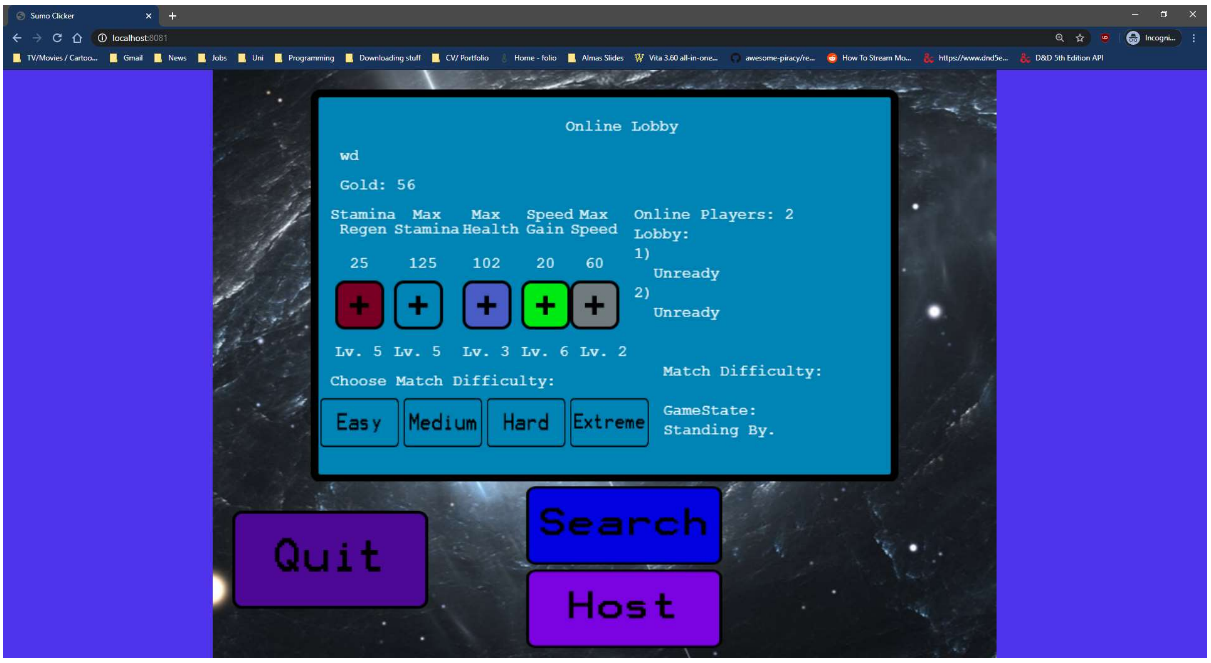
Finally, later difficulties still carry their bonuses and match changes from the single player counterpart, leaving it up to the player to what match difficulty they wish to search for. Either the more frantic or less rewarding easy mode or the dangerous and bountiful extreme difficulty duels.

Storyboard





Difficulty: Easy
GameMode: Duel
Selected Character:



Online Lobby

wd

Gold: 56

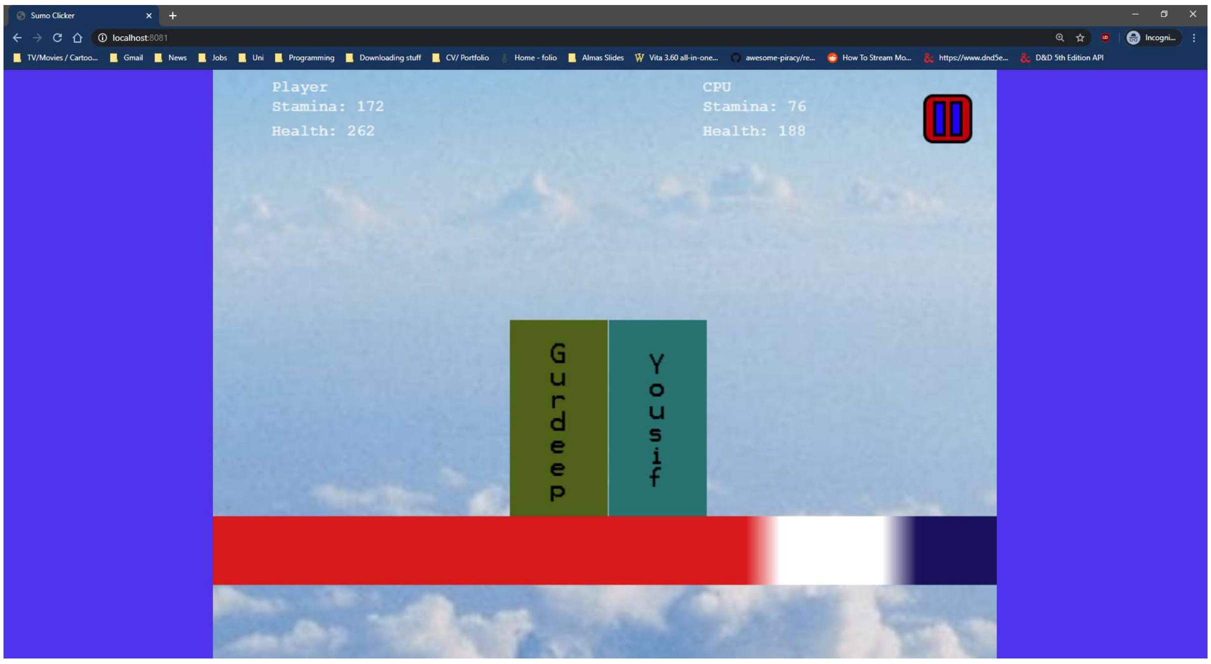
Stamina Regen	Max Stamina	Max Health	Speed Gain	Max Speed
25	125	102	20	60
Lv. 5	Lv. 5	Lv. 3	Lv. 6	Lv. 2

Choose Match Difficulty:

Online Players: 2
Lobby:
1)
Unready
2)
Unready

Match Difficulty:

GameState:
Standing By.



Player
Stamina: 172
Health: 262

CPU
Stamina: 76
Health: 188

Implementation specification

Sumo Clicker was implemented utilizing the Phaser 3 engine and the es6 denomination of JavaScript. This was selected due to es6 likening to programming languages I am familiar with and utilizing the Phaser engine for previous projects allows me to be familiar with the sub systems and features of Phaser allowed for an eased development process.

Development was broken down into three stages: Non-Gameplay, Offline Single player, and Online Multiplayer. Non-Game play development focused on creating core functions and UI aspects of the game. The core functions, such as one that calculated the centre of the Game object for eased asset manipulation, could be placed in the base class to which all scenes within sumo clicker extended from. This ensured that functions that were incredibly common could easily be accessed.

All data for these scenes were stored within the object's own scene class as using es6 meant that making the base class allowed to utilize class fields and variables to data to scenes using the object model that es6 provides.

The single player section focused on also utilizing the fields in the scene class to store all data that needed to be accessed or manipulated, this also ensured that data persisted between scene's where necessary such as for the management of the data between the main game and the UI scene running in parallel to both the single player game and its online counterpart.

Data for the Online aspect is handled differently as data needed to persist between browser sessions data would need to be saved in some capacity. While it was originally envisioned that the system would save the player's data within an encrypted json file, I opted to save this data utilizing cookies as the JSON file would not be able to guarantee any way of sever to client authentication which would be critical for storing the user data on the server. The Cookies were saved using the standard JS practice and manipulated using key functions to save and load the data upon transitioning into the online section of the game.

Server side, implementation focused on a call and response style of development utilizing socket.io to emit pings to the server and respective clients based on client states. The server would also server as a temporary database by storing key player information within a data object within the server itself. This could be queried using the individual socket id's to search for data between any players and utilizing a nested set of associated arrays to create a lobby system that could allow players to search for and host matches at ease.

Network utilisation

For a large amount of network interactions focused on instance-based client-side pinging, driven by the need to either designate a player as either a host or an opponent and work on passing data between the two types that exist within the same lobby. Almost all network interactions occur within the online lobby as the game is handled in real time and the input is simply pinged to the server as input is handled but we will cover this later on.

The initial interaction is direct connection to the server, this connects the server and client and stores the socket information within the scene to later communicate with the server. The main two interactions are the selection of to either host or search for a game. For ease of both development and understanding, hosting creates a lobby and designates you as a host of that lobby while effectively identical to other clients, this allows to ensure that finding a target to transmit data to would always be possible and accessible wear necessary.

Scene Starts and creates connection=> Server Registers listener functions to socket and passes back number of online players => Client displays information in appropriate location.

Once the lobby is created, the user will wait for a additional player to join and be updated on this by a series of pings that tell the user if a player has found their lobby, joined it and even readied up for the match. Once ready the user can initiate the match by click of button. This instead pings the server to ensure both players start the game at the same time. This is done like so:

User Searches for Lobby using difficulty as a param => Lobby either finds lobby or tells user it no lobby has been found => Lobby Data and client data is passed back and forth between the two clients to ensure they both can function.

For input and game state pings are made at key points such as the game beginning to synchronize the start of the match (accounting for load times) registering the end of the game and dealing with post game effects of the match. Input is also handled by creating pings to the server which are then passed to the opponent socket to be mimicked on the other players client. This is done like so:

User click/taps => Client Pings server saying it what action it did with lobby data =>server pings other client within the lobby => client activated corresponding input on this client.

Finally, the last network interactions are error handling and ensuring that later issue wouldn't arise using the lobby system by deleting all lobbies associated with the disconnecting socket and alerting any players in those lobbies that this is happening as to ensure they can search for another game or wait for another player to join their lobby.

Critical Review

In terms of the project, I believe that I have made a robust and interesting game that matches the brief but I do believe that aspects of the game could be improved and I will be concentration this section of the report dedicated to covering theses. I will also be detailing on the games successes as to ascertain key parts of the project that are beneficial to increase my skills within and those aspects that are integral to improve upon.

The first Aspect that could use work is the graphical aspect of the game. Many assets within the game are currently placeholders or stand ins to be replaced with more graphically impressive asset if time allowed for their creation. As I opted to focus on functionality as a primary focus it allowed me to finish the development of gameplay of the project but not the theming or aesthetics which is still a key part of game development.

Next is the lack of sounds within the project, while not entirely important the lack of any sounds within the game means their s no cohesion towards the games theming or Style. It also prevents a lot of accessible play as due to having to react quickly without sound cues it becomes a big issue.

Finally, and for a less superficial aspect to improve upon is the network handling aspect of the game. While functionally there are still issues that can arise between data transfer that cause issues for the game's presentation and occasionally functionality are tarnished due to not having redundancies for this issue. This could improve not only the game but also its longevity as these issues could make the player frustrated and stop playing completely.

In terms of beneficial implementations, I believe the client-side instancing is very robust relying on a cyclic system for the input of both the Host and opponent's movement inputs. This system ensured that all clients worked identically, and issues would be handleable within the client-side code where necessary.

Another good Implementation is the creation of the single player to use as a framework and a training mode. Due to the game's gameplay being straight forward and simple training would not be necessary but due to the risks of online gameplay, the users can have a stress-free environment as a preparation for the online. The single player also was implemented in a way as to ensure the functionality once the code was transplanted into the Online Game's implementation.

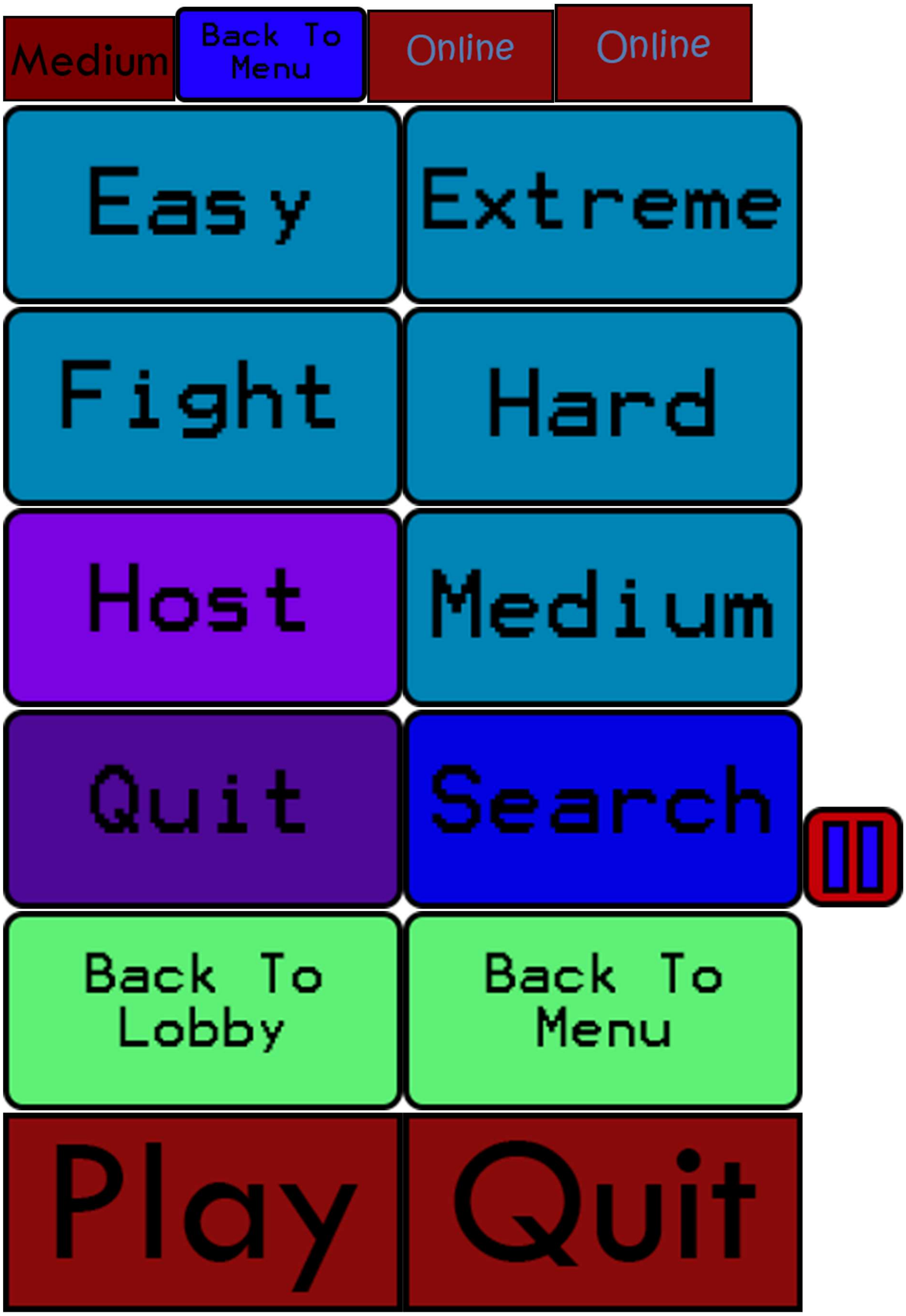
Finally, a big key implementation was the initial creation and extension of a base class to store critical and frequented functions within each of the Phaser scenes. This allowed for standardized designing, cookie management, error handling and data management regardless of what Phaser scene was active.

Assets

Below I will be including a repository of assets used within sumo clicker. All assets apart from the backgrounds were created by me using adobe photoshop.

Backgrounds

Buttons


Medium
Back To Menu
Online
Online
Easy
Extreme
Fight
Hard
Host
Medium
Quit
Search
Back To Lobby
Back To Menu
Play
Quit

Back to Menu

Reset Save Data

Resume

S

S

Single Player

Single Player

Tutorial

Tutorial

+ + + +

+ +

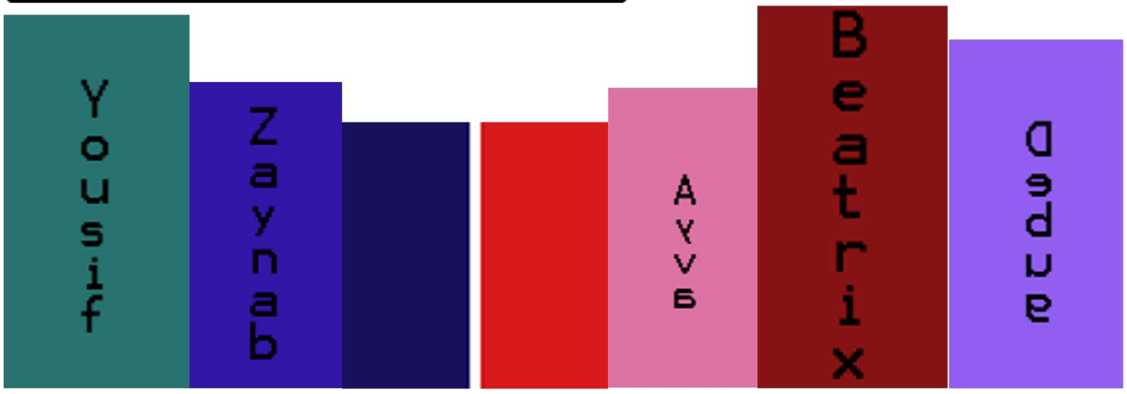
Back to Lobby

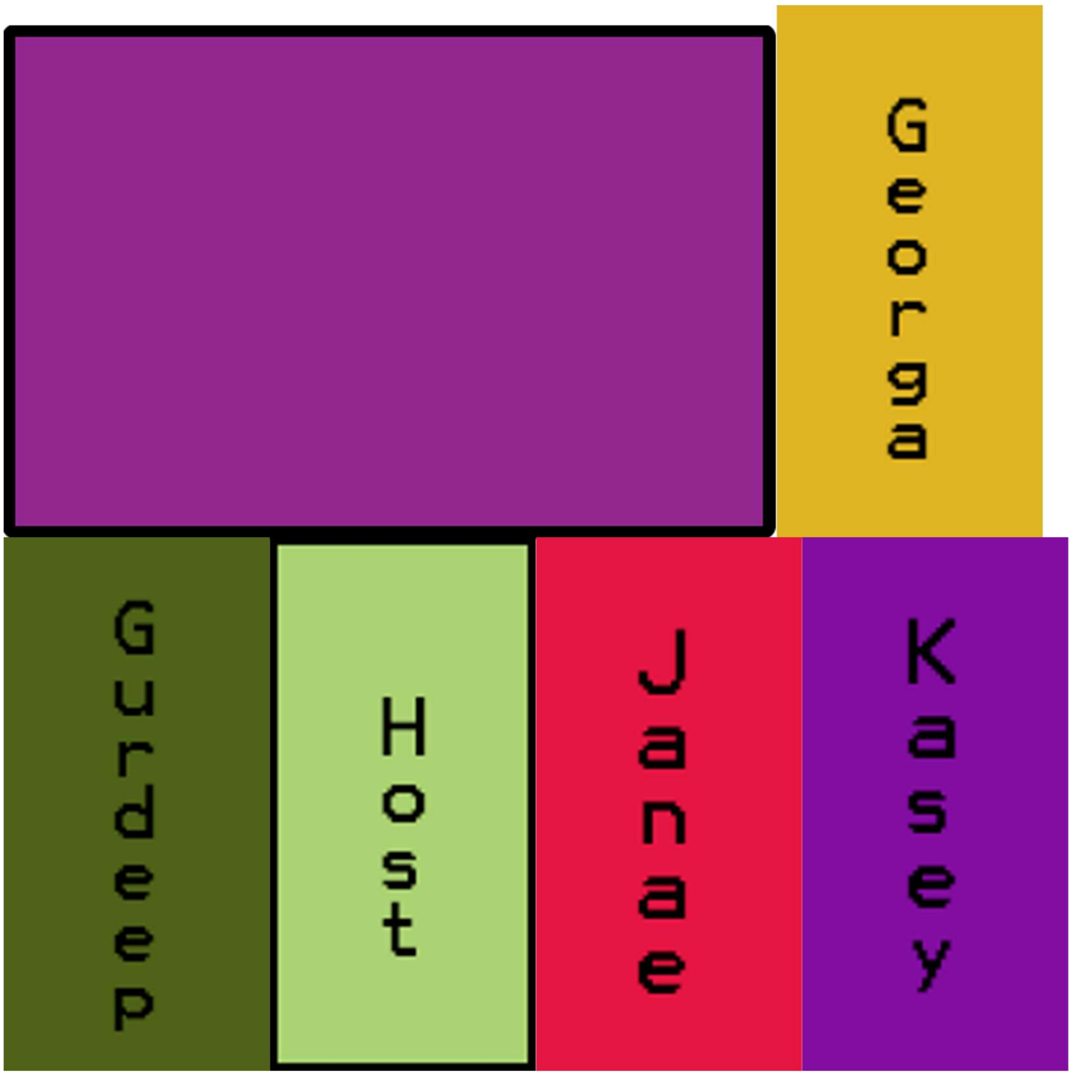
Back to Menu	Arena Battle
Back	Back to Menu
DEBUG	Duel
Easy	Extreme
Fight!!!	Hard

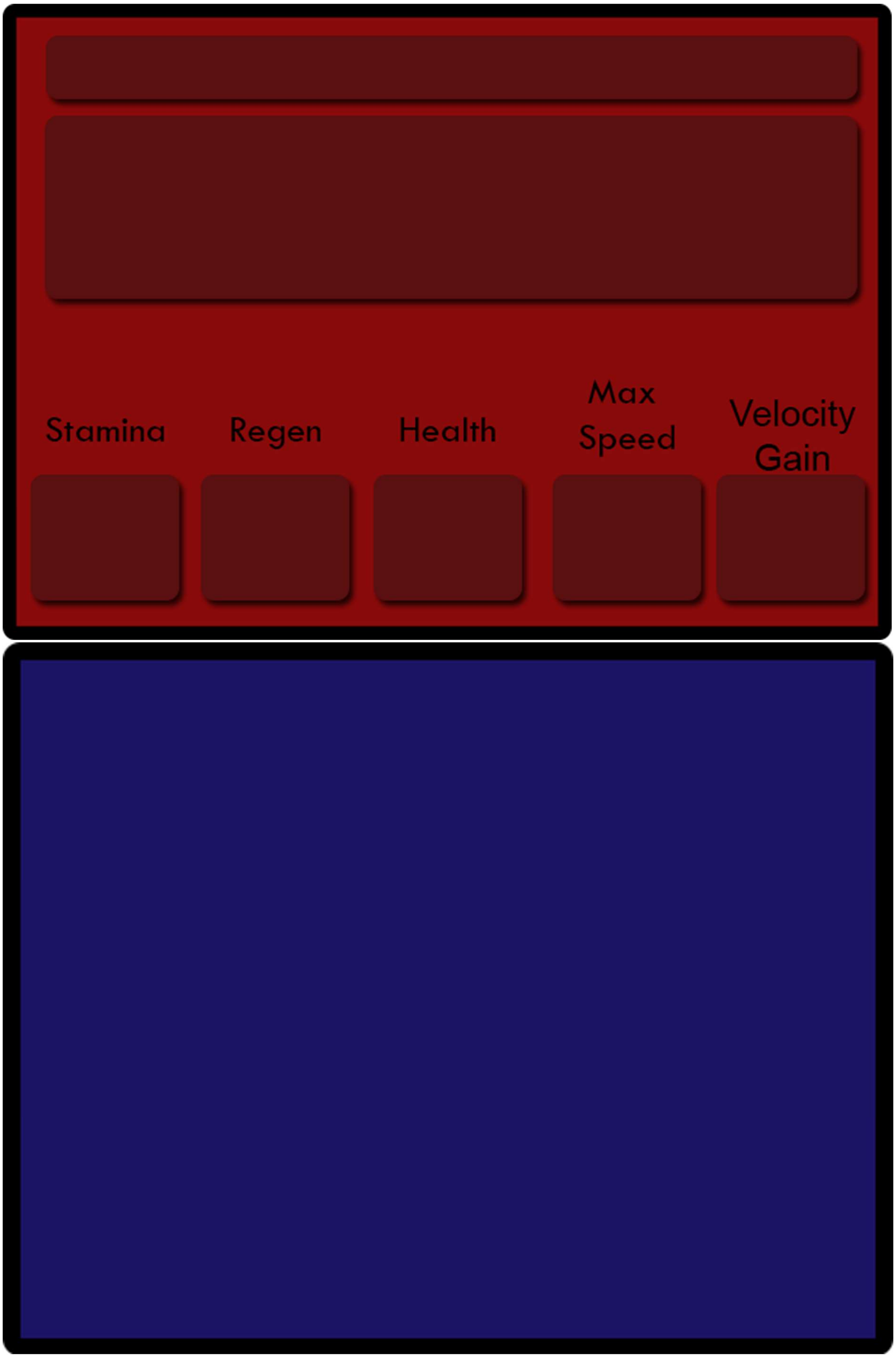
Leave Lobby ←

Back To Lobby

Images




Georga
Gurdeep
Host
Janae
Kasey


Stamina
Regen
Health
Max Speed
Velocity Gain

L
u
k
a
s

O
P
P

Sumo Clicker

Sumo Clicker

Ubaid

References

Renaux, J., 2017. *How to make a multiplayer online game with Phaser, Socket.io and Node.js.* [Online]
Available at: https://www.dynetisgames.com/2017/03/06/how-to-make-a-multiplayer-online-game-with-phaser-socket-io-and-node-js/
[Accessed 22 May 2020].

Westover, S., 2019. *Create a Basic Multiplayer Game in Phaser 3 with Socket.io – Part 1.* [Online]
Available at: https://gamedevacademy.org/create-a-basic-multiplayer-game-in-phaser-3-with-socket-io-part-1/
[Accessed 20 May 2020].