Match Making System

Technical Design Document

*Version 1.0*

*Omar Ibrahim*

*Website: omar47i.github.io/*

*Mobile: +85259393797*

# Table of Contents

[Table of Contents](#_Toc71494091)

[Design History](#_Toc71494092)

[Solution Overview](#_Toc71494093)

[Solution Summary](#_Toc71494094)

[Target Platform](#_Toc71494095)

[Development Overview](#_Toc71494096)

[Development Team](#_Toc71494097)

[Development Software](#_Toc71494098)

[External Code](#_Toc71494099)

[Solution Implementation](#_Toc71494100)

[Main Technical Requirements](#_Toc71494101)

[Architecture](#_Toc71494102)

[System Flow](#_Toc71494103)

[Graphics](#_Toc71494104)

[Audio](#_Toc71494105)

[Game Objects and Logic](#_Toc71494106)

# 

# 

# Design History

Version 1.0: First produced version of the match making system.

# 

# Solution Overview

--------------------------------------------------------------------------------------------------------------------

# Solution Summary

The match making system is responsible for match making the players based on the game mode they pick, which is PvP or 1v1, a team of 2v2, and a team of 3v3.

The solution load all the players in the JSON file, as the players enter the queue waiting to be matched, the matchmaker system tries to find a match for each player based on the SR (Skills Ranking) of both of them, there is also a team SR that is calculated and are matched in the same way as matching two player together, the solution includes UI stats elements that shows how the system works during run-time with different controls that can be used to monitor the matchmaking process.

## Target Platform

As a generic matchmaking solution, it can be used with any platform weather it’s a PC, Mac, or mobile platforms.

# Development Overview

--------------------------------------------------------------------------------------------------------------------

# Development Team

[Omar Ibrahim](https://omar47i.github.io/)

Match making system design and implementation, JSON data loading and parsing, and UI stats elements implementation.

# Development Software

Project is developed using Unity game engine 2020.3.7 LTS, Version control is handled by [Git](http://git-scm.com/) with the repository hosted on [GitHub](https://github.com/Omar47i/MatchMakingSystem).

# External Code

I haven’t used any plugin or external code, I used visual studio IDE for coding.

The solution has been made from scratch.

# Solution Implementation

--------------------------------------------------------------------------------------------------------------------

# Main Technical Requirements

The main technical requirements for the matchmaking system is straight forward:

* Adding a player to the queue.
* Perform a greedy algorithm to match players of similar SR.
* Try matching a player in 1v1 with another player.
* Try matching a player in 2v2 with another player or team.
* Try matching a player in 3v3 with another player or team.
* Create a match out of the two teams if possible.
* Create a match out of the two players in 1v1 mode.
* Compare players SR.
* Compare Team SR with player SR.
* Stats UI.
* Real-time stats as the matchmaker is in progress.

# Architecture

There is only one main module in the game which is the main scene, let’s call it main module.

Main module hosts core matchmaking logic and the stats UI

# System Flow

When the scene starts, the user is presented with a stats panel including different controls, to start the match making experience, we need to add players to the queue, the system reads all the players listed in the JSON file for us, we can add them to the queue by clicking on “Add 1 Player” to add one player for the selected mode, or we can click on “Add All Players” to add all the players in the queue for the selected mode, we can see the “Players In Queue” view is updated with added players with their SR displayed next to them, to start the match making process, we have two options, either perform one step of trying to match a player by clicking on “Find Match One Step”, or we can run the match maker continuously by clicking on “Toggle Find Match”, as the match maker matches players into teams and together, the three views will be updated based on the matches, and the stats also shows the number of players currently added to the queue plus the state of the match maker, whether it’s idle or running.

# Graphics

Game graphics is pretty simple, I used the default UI graphics as the solution has no gameplay elements.

# Audio

The game has no audio

# Game Objects and Logic

**GameManager**

The GameManager gameObject hold the GameManager.cs script which acts as the controller script for the whole solution, it exposes three values for the thresholds of the three game modes to be edited in the inspector.

**TeamsBuilder.cs**

TeamsBuilder script is the main algorithm for the matchmaking, it contains all the processing that needs to be done on individual players and teams to match them based on the criteria that I selected which is the SR difference, it’s also responsible for creating teams and matches.

**PlayerData.cs**

Holds the mapping of the player entity in the json file.

**Player.cs**

Holds the gameplay attributes for a player in the matchmaking system.

**UIController.cs**

The main script for displaying and updating the UI stats that shows how the match making system works in real-time.

**SampleDataParser.cs**

Parsing the players json file to be used in matchmaking.

**Singleton.cs**

A well written implementation of the Singleton design pattern