**Target Advanced Game**

Sommario

[1 Introduzione 4](#_Toc134128598)

[1.1 Informazioni sul progetto 4](#_Toc134128599)

[1.2 Abstract 4](#_Toc134128600)

[1.3 Scopo 4](#_Toc134128601)

[2 Analisi 5](#_Toc134128602)

[2.1 Analisi del dominio 5](#_Toc134128603)

[2.2 Analisi e specifica dei requisiti 5](#_Toc134128604)

[2.3 Use case 8](#_Toc134128605)

[2.4 Pianificazione 9](#_Toc134128606)

[2.5 Analisi dei mezzi 10](#_Toc134128607)

[2.5.1 Software 10](#_Toc134128608)

[2.5.2 Hardware 11](#_Toc134128609)

[3 Progettazione 11](#_Toc134128610)

[3.1 Design dei dati e database 11](#_Toc134128611)

[3.2 Design delle interfacce 12](#_Toc134128612)

[3.3 Design procedurale 19](#_Toc134128613)

[4 Implementazione 20](#_Toc134128614)

[4.1 Interfacce 20](#_Toc134128615)

[4.1.1 Main Scene 20](#_Toc134128616)

[4.1.2 Play Scene 21](#_Toc134128617)

[4.1.3 How to Play Scene 22](#_Toc134128618)

[4.1.4 Settings Scene 23](#_Toc134128619)

[4.1.5 Match History Scene 24](#_Toc134128620)

[4.1.6 Login Scene 25](#_Toc134128621)

[4.1.7 Create Account Scene 26](#_Toc134128622)

[4.2 Scripts 27](#_Toc134128623)

[4.2.1 UIButtonManager 27](#_Toc134128624)

[5 Test 28](#_Toc134128625)

[5.1 Protocollo di test 28](#_Toc134128626)

[5.1.1 Test Case Obbligatori 29](#_Toc134128627)

[5.2 Risultati test 34](#_Toc134128628)

[5.3 Mancanze/limitazioni conosciute 34](#_Toc134128629)

[5.3.1 Sito WEB 34](#_Toc134128630)

[6 Consuntivo 35](#_Toc134128631)

[7 Conclusioni 35](#_Toc134128632)

[7.1 Sviluppi futuri 35](#_Toc134128633)

[7.2 Considerazioni personali 35](#_Toc134128634)

[8 Glossario 35](#_Toc134128635)

[9 Bibliografia 36](#_Toc134128636)

[9.1 Sitografia 36](#_Toc134128637)

[10 Allegati 36](#_Toc134128638)

[Figura 1 - Use case 8](file:///E:\01%20I3BC\M306%20+%20Progetti\2Sem\TAG\3_Documentazione%20(word%20e%20pdf)\Documentazione_TAG.docx#_Toc134128639)

[Figura 2 - Diagramma di Gantt Preventivo 9](file:///E:\01%20I3BC\M306%20+%20Progetti\2Sem\TAG\3_Documentazione%20(word%20e%20pdf)\Documentazione_TAG.docx#_Toc134128640)

[Figura 3 – Diagramma E/R 11](#_Toc134128641)

[Figura 4 - Interfaccia iniziale 12](#_Toc134128642)

[Figura 5 - Schermata selezione partita 13](#_Toc134128643)

[Figura 6 - Interfaccia lobby 14](#_Toc134128644)

[Figura 7 - Interfaccia Impostazioni 15](#_Toc134128645)

[Figura 8 - Mappatura tasti (controller e tastiera) 16](#_Toc134128646)

[Figura 9 – Interfaccia base di gioco 17](#_Toc134128647)

[Figura 10 - Interfaccia puntatore attivo 17](#_Toc134128648)

[Figura 11 - Interfaccia scudo attivo 18](#_Toc134128649)

[Figura 12 - Interfaccia classifica 18](#_Toc134128650)

[Figura 13 - Scena principale 20](#_Toc134128651)

[Figura 14 - Scena scelta lobby 21](#_Toc134128652)

[Figura 15 - Scena visualizzazione tasti 22](#_Toc134128653)

[Figura 16 - Scena personalizzazione sensibilità 23](#_Toc134128654)

[Figura 17 - Scena Partite Passate 24](#_Toc134128655)

[Figura 18 - Scena di Login 25](#_Toc134128656)

[Figura 19 - Scena Creazione Account 26](#_Toc134128657)

[Figura 20 - UIButtonManager Variabili 27](#_Toc134128658)

[Figura 21 - UIButtonManager Metodi 28](#_Toc134128659)

# Introduzione

## Informazioni sul progetto

* Allievi: Simone Riva, Michael Dobeson, Stefano Ceschi
* Classe: I3AC-BC
* Docente responsabile: Michel Palucci
* Data inizio: 27.01.2023
* Data fine: 5.05.2023

## Abstract

*T.A.G. is a virtual recreation of the Laser Tag game, that is set in the future with a futuristic environment and characters.*

*In Laser Tag everyone is against everyone and you have to point your laser towards the enemy to disable his laser and earn a point.*

*In our game there will be a map with randomly generated obstacles where the player has to hit other enemies/other players with their laser pointer. You can also defend yourself using a shield to stop the laser from touching you.*

*The player is also able to see his scores in real time thanks to a leaderboard in the center of the map, he can also see the top statistics through his registered account on a website with a Database.*

## Scopo

Lo scopo del progetto è di avere una variazione virtuale del gioco laser tag con una generazione casuale di oggetti all’interno del campo di gioco per stimolare l’utente ad inventarsi una strategia di gioco diversa ad ogni partita.

# Analisi

## Analisi del dominio

Il gioco dovrebbe essere il più semplice possibile per gli utenti, andando a riprendere le caratteristiche principali del gioco reale. La particolarità del nostro gioco è la virtualizzazione del tutto tramite videogioco. Essendo uguale all’originale, gli utenti impareranno velocemente come si gioca.

## Analisi e specifica dei requisiti

Priorità 2/3 = opzionale

|  |  |
| --- | --- |
| ID: REQ-01 | |
| Nome | Movimento Giocatore |
| Priorità | 1 |
| Versione | 1.0 |
| Note | L'utente può usare dei comandi di tastiera o joystick per muovere il personaggio. |
| Sotto requisiti | |

|  |  |
| --- | --- |
| ID: REQ-02 | |
| Nome | Interfaccia Grafica |
| Priorità | 1 |
| Versione | 1.0 |
| Note | L'utente può navigare attraverso l'interfaccia di gioco usando tastiera o joystick. |
| Sotto requisiti | |
| 001 | Interfaccia UI per gestire tutte le impostazioni (es: audio, grafica, account, ...). |

|  |  |
| --- | --- |
| ID: REQ-03 | |
| Nome | Ambiente Partita |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Deve essere presente una mappa di gioco. |
| Sotto requisiti | |

|  |  |
| --- | --- |
| ID: REQ-04 | |
| Nome | Sito Web |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Deve essere presente un sito web. |
| Sotto requisiti | |
| 001 | **L’utente crea un account** |

|  |  |
| --- | --- |
| ID: REQ-05 | |
| Nome | Interazioni Giocatore |
| Priorità | 1 |
| Versione | 1.0 |
| Note | L'utente può mirare e puntare un laser ed eventualmente parare i laser degli altri usando i comandi di tastiera o joystick. |
| Sotto requisiti | |

|  |  |
| --- | --- |
| ID: REQ-06 | |
| Nome | Ostacoli Casuali |
| Priorità | 1 |
| Versione | 1.0 |
| Note | I muri si generano in maniera casuale all'inizio di ogni partita usando l'algoritmo di Manhattan Mapper. |
| Sotto requisiti | |

|  |  |
| --- | --- |
| ID: REQ-07 | |
| Nome | HUD |
| Priorità | 1 |
| Versione | 1.0 |
| Note | L'Heads-Up Display (HUD) mostrerà varie informazioni durante la partita. |
| Sotto requisiti | |
| 001 | Tempo rimasto di partita. |
| 002 | Punteggio |

|  |  |
| --- | --- |
| ID: REQ-08 | |
| Nome | AI Nemico |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Un'AI che si comporta come un giocatore. |
| Sotto requisiti | |

|  |  |
| --- | --- |
| ID: REQ-09 | |
| Nome | Database |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Il database conterrà tutte le tabelle utili per lo storage di dati e per il multiplayer. |
| Sotto requisiti | |
| 001 | Tabella Leaderboard Globale: sarà una tabella consultabile dal sito web dove verranno mostrati i migliori giocatori di sempre con il punteggio. |
| 002 | Tabella Leaderboard della Partita: conterrà i punteggi e i giocatori della partita corrente. |

|  |  |
| --- | --- |
| ID: REQ-10 | |
| Nome | Multiplayer |
| Priorità | 1 |
| Versione | 1.0 |
| Note | Il giocatore può collegarsi ad una sessione con altri giocatori. |
| Sotto requisiti | |

|  |  |
| --- | --- |
| ID: REQ-11 | |
| Nome | Algoritmo di Cell-Shading |
| Priorità | 2 |
| Versione | 1.0 |
| Note | La grafica del gioco è da cartone animata però in 3D. |
| Sotto requisiti | |

|  |  |
| --- | --- |
| ID: REQ-12 | |
| Nome | Video Clip |
| Priorità | 3 |
| Versione | 1.0 |
| Note | Le partite vengono registrate e mandati al sito web per essere visualizzati. |
| Sotto requisiti | |

In accordo con il committente durante la fase di sviluppo abbiamo deciso di eliminare il sito web e dare comunque la possibilità all’utente di creare l’account su Unity

|  |  |
| --- | --- |
| ID: REQ-13 | |
| Nome | Creazione Account |
| Priorità | 2 |
| Versione | 1.0 |
| Note | L’utente crea un account |
| Sotto requisiti | |

## Use case

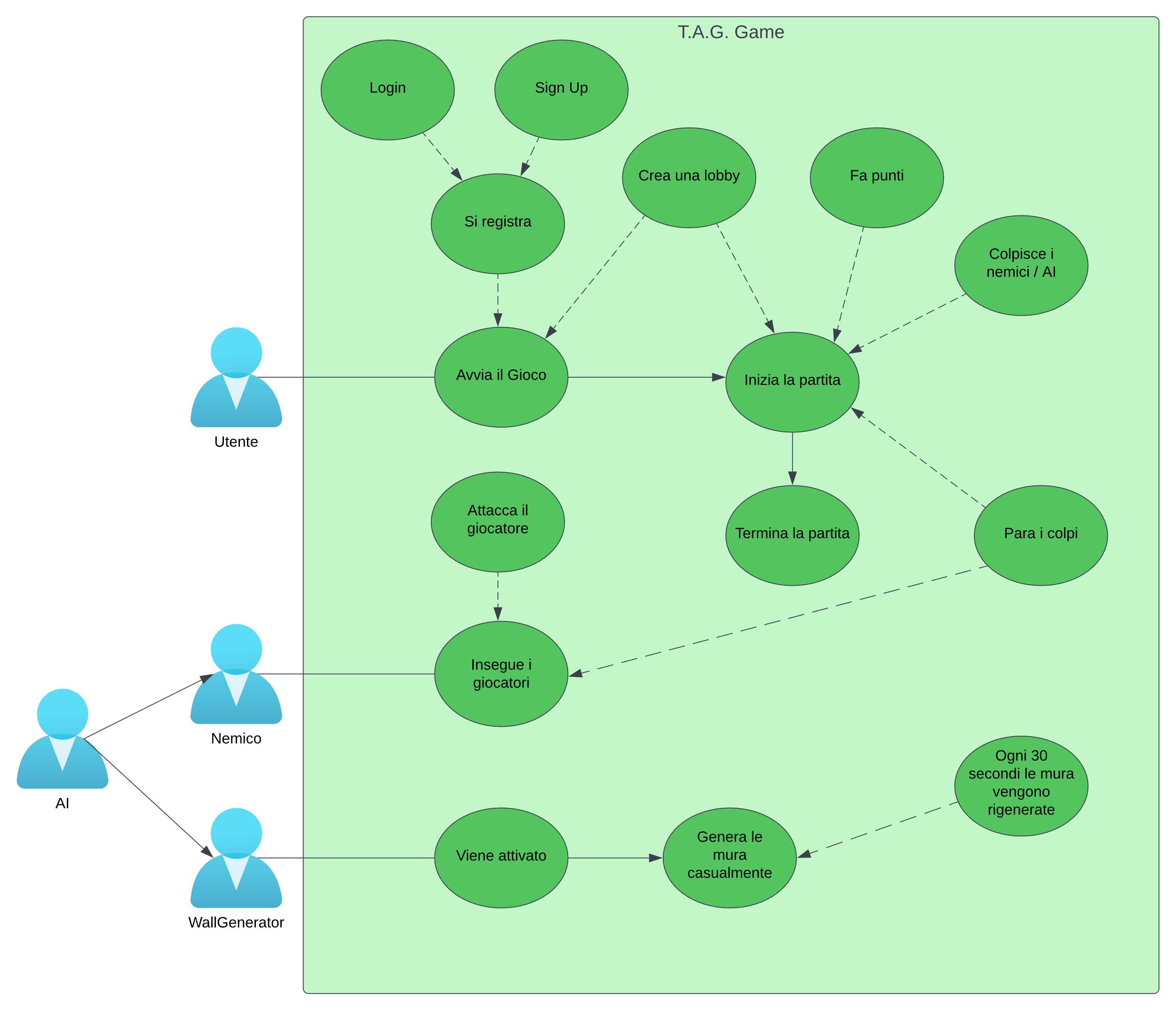
I casi d’uso rappresentano l’interazione tra i vari attori e le funzionalità del prodot

Figura 1 - Use case

## Pianificazione

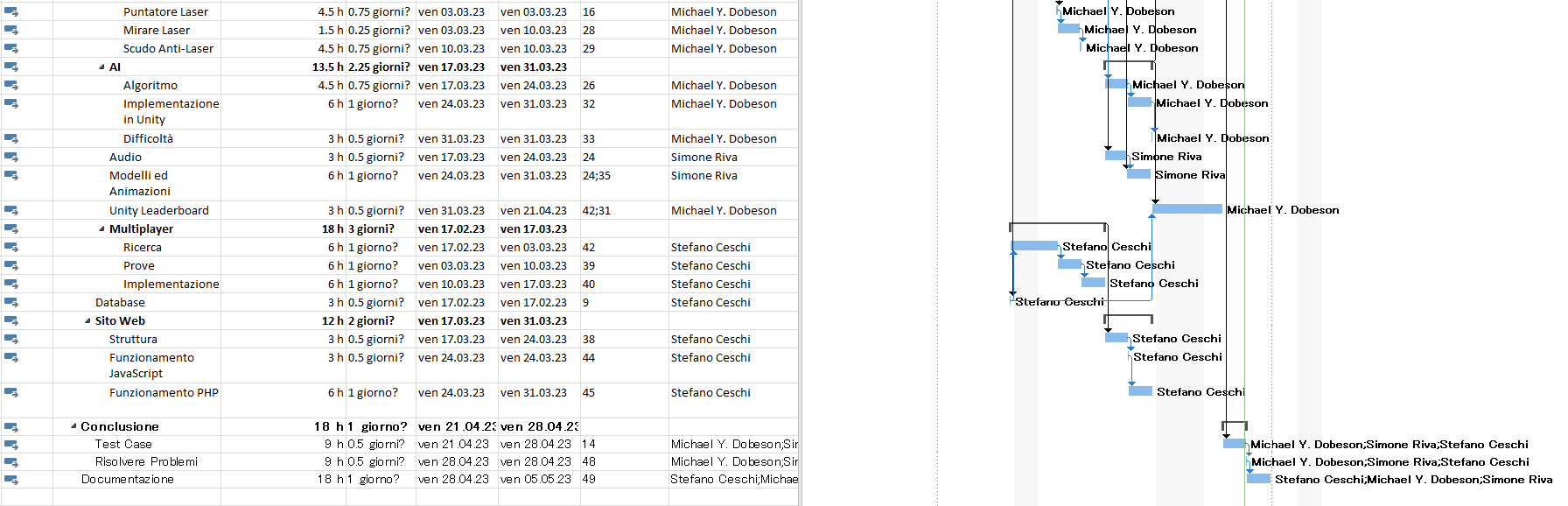


Figura 2 - Diagramma di Gantt Preventivo

## Analisi dei mezzi

***PC scolastici:***

* **CPU:** *Intel Core I7 7th generation*
* **GPU:** *Intel HD Graphics 730 (abbiamo avuto problemi con questa scheda video)*
* **RAM:** *16GB 3.60GHz*
* **DISCO:** *Western Digital 500GB (HDD)*

***Mezzi esterni:***

* 1x SSD SanDisk 500GB
* 1x SSD SanDisk 1TB
* 1x SSD Samsung 256GB

### Software

***Sviluppo gioco***

* Unity 2022.1.f1 (*personal license*)
* Netcode for GameObjects
* Visual Studio 2022 + 2019
* Blender

***Sviluppo WEB***

* PHPStorm (*school license*)
* Visual Studio Code
* MySQL

***Ambiente di test***

* XAMPP

### Hardware

Il prodotto è disegnato per poter funzionare fluentemente sulla maggior parte dei computer in circolazione, è però consigliato utilizzare hardware performante per evitare sbalzi di *framerate* oppure di prestazioni di rete.

Per poter usufruire del nostro prodotto è **consigliato** possedere una connessione ad internet stabile per evitare il *lag*.

Durante lo sviluppo potremmo utilizzare i PC scolastici descritti nel [capitolo 2.5](#_Analisi_dei_mezzi).

# Progettazione

## Design dei dati e database

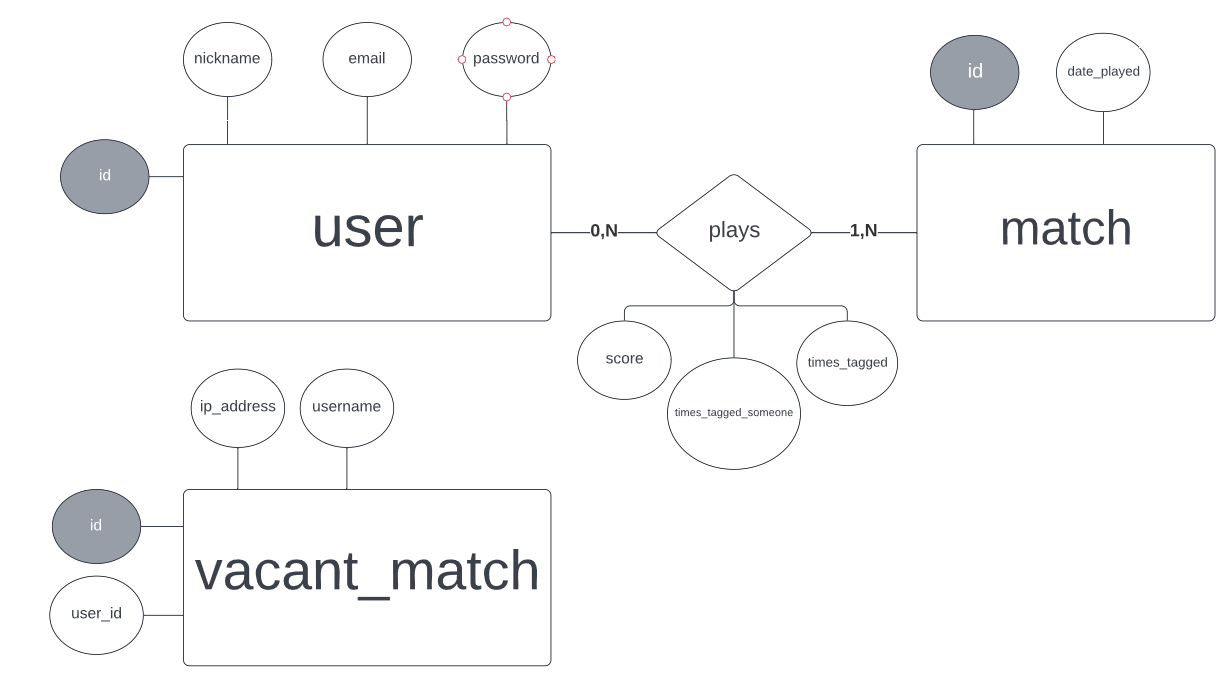


Figura 3 – Diagramma E/R

**DA RIVEDERE CON PALUCCI PER LA TABELLA NON RELAZIONATA**

## Design delle interfacce



Figura 4 - Interfaccia iniziale

La prima interfaccia visualizzata dall’utente comprende:

* Un bottone “Play” che fa iniziare la partita
* Un bottone “Settings” che permette all’utente di modificare delle impostazioni
* Un bottone “How to play” che spiega agli utenti i tasti per poter giocare
* Un bottone “Exit” che permette all’utente di uscire dal gioco



Figura 5 - Schermata selezione partita

Quando l’utente preme sul bottone “Play” arriva all’interfaccia di selezione partita che comprende:

* Il nome della lobby
* La quantità di giocatori già in partita
* Il bottone “Join” che permette all’utente di entrare in partita
* Il bottone “Create” che permette di creare una lobby



Figura 6 - Interfaccia lobby

Se l’utente preme sul bottone “Create” può creare una lobby che comprende:

* Un TextBox per l’inserimento del nome della lobby
* Il bottone “Create” per creare la lobby

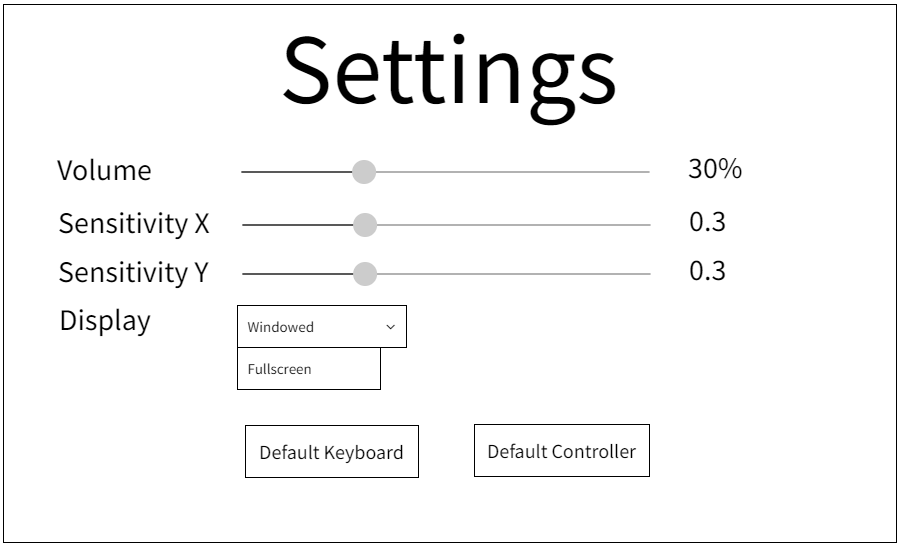


Figura 7 - Interfaccia Impostazioni

La schermata di impostazioni viene visualizzata quando l’utente preme sul bottone “Settings”, essa comprende:

* Uno slider pel la regolazione del volume
* Uno slider pel la regolazione della sensibilità orizzontale
* Uno slider pel la regolazione della sensibilità verticale
* Un dropdown per la regolazione del display (a finestra o schermo intero)
* Un bottone che regola cambia gli slider ad un valore predefinito per l’uso della tastiera
* Un bottone che regola cambia gli slider ad un valore predefinito per l’uso della tastiera

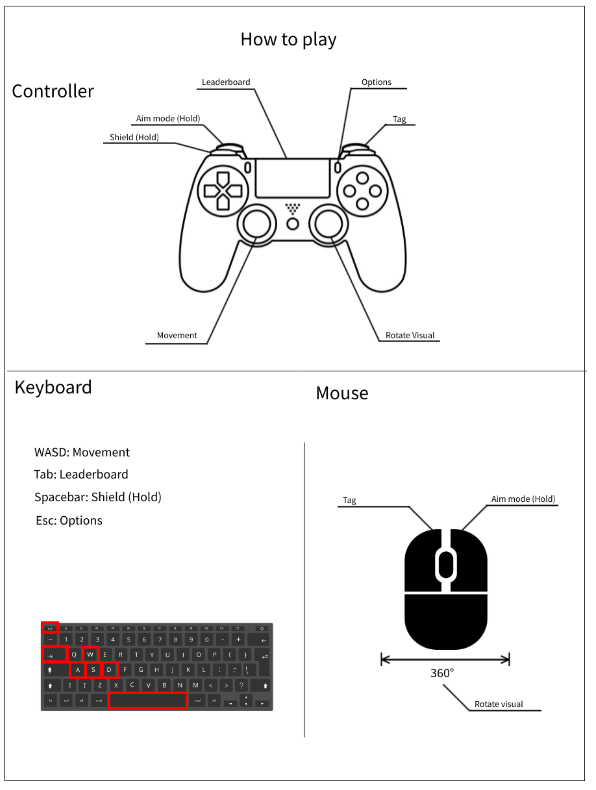


Figura 8 - Mappatura tasti (controller e tastiera)

La schermata How to play viene visualizzata quando l’utente preme sul bottone “How to play”, questa schermata permette all’utente di poter visualizzare i tasti per giocare (sia controller che tastiera).

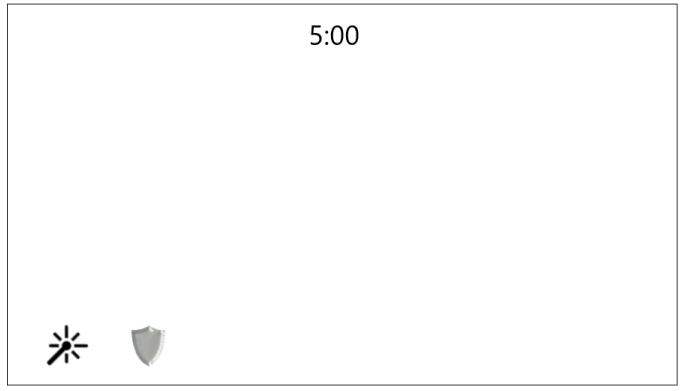


Figura 9 – Interfaccia base di gioco

Questa interfaccia appare quando il giocatore preme il bottone “Play”. Vengono visualizzati la possibilità di utilizzare il puntatore laser e lo scudo tramite due tasti del controller/tastiera, inoltre il player vede l’ambiente di gioco. Il tempo in alto è la durata della partita



Figura 10 - Interfaccia puntatore attivo

Questa interfaccia appare quando il giocatore sta utilizzando il puntatore laser impedendo di utilizzare lo scudo, inoltre il player vede l’ambiente di gioco. Il tempo in alto è la durata della partita

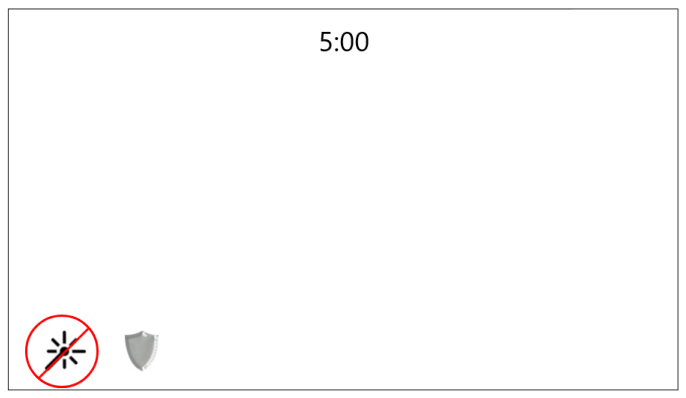


Figura 11 - Interfaccia scudo attivo

Questa interfaccia appare quando il giocatore sta utilizzando lo scudo impedendo di utilizzare il puntatore laser, inoltre il player vede l’ambiente di gioco. Il tempo in alto è la durata della partita

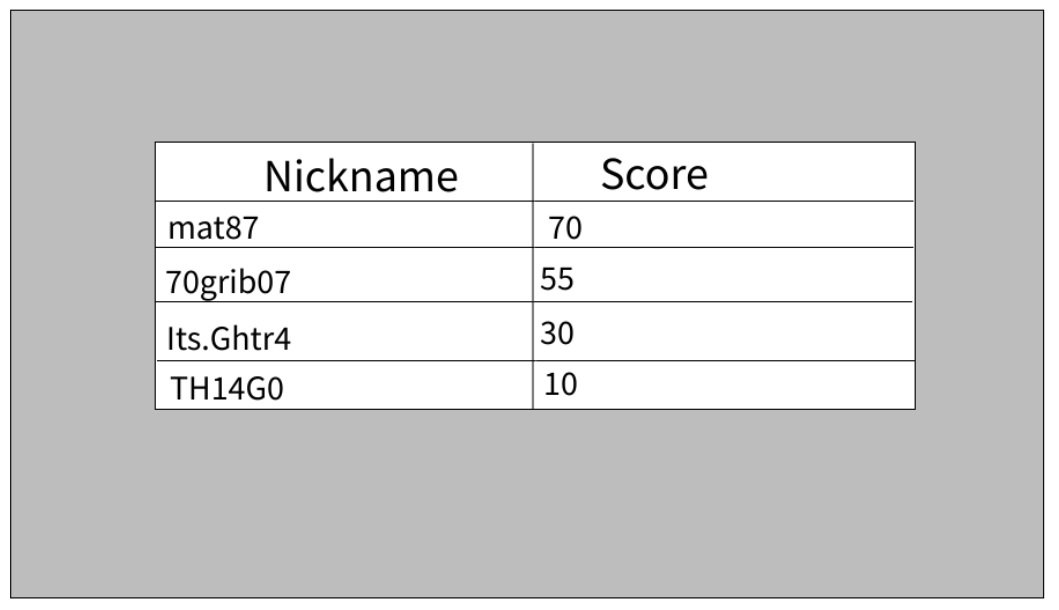


Figura 12 - Interfaccia classifica

Questa schermata appare durante la partita quando il player schiaccia su “Tab”, viene visualizzata la classifica della partita. Inoltre il background diventa grigio.

## Design procedurale

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

## Interfacce

### Main Scene



Figura 13 - Scena principale

Questa scena è la prima scena che visualizza l'utente subito dopo aver visto il video introduttivo, essa comprende:

* il Bottone “Play” che consente di visualizzare le lobby
* il Bottone “Play offline” che consente di giocare contro tre bot
* il Bottone “How to play” che consente di visualizzare i tasti sia di controller che di mouse e tastiera
* il Bottone “Settings” che consente di impostare le sensibilità di gioco
* il Bottone “Match history” che consente di visualizzare le partite passate
* il Bottone “Exit” che consente di uscire dal gioco
* il Bottone “Login/Logout” che consente di accedere è rispettivamente uscire con il proprio account
* il Bottone “Credits” che visualizza i ”titoli di coda” del gioco

### Play Scene



Figura 14 - Scena scelta lobby

Quando l'utente clicca sul bottone play dopo aver fatto login vedrà questa schermata, essa è composta da:

* in alto a destra il Bottone “Load lobbies” che permette di visualizzare le partite non ancora iniziate
* in alto a sinistra un Bottone che permette di tornare indietro la scena iniziale
* il Bottone “Join” che permette all'utente di accedere alla partita dell'utente chiamato “User”
* il Bottone “Create” che permette di creare la partita a nome dell'utente che lo preme

### How to Play Scene

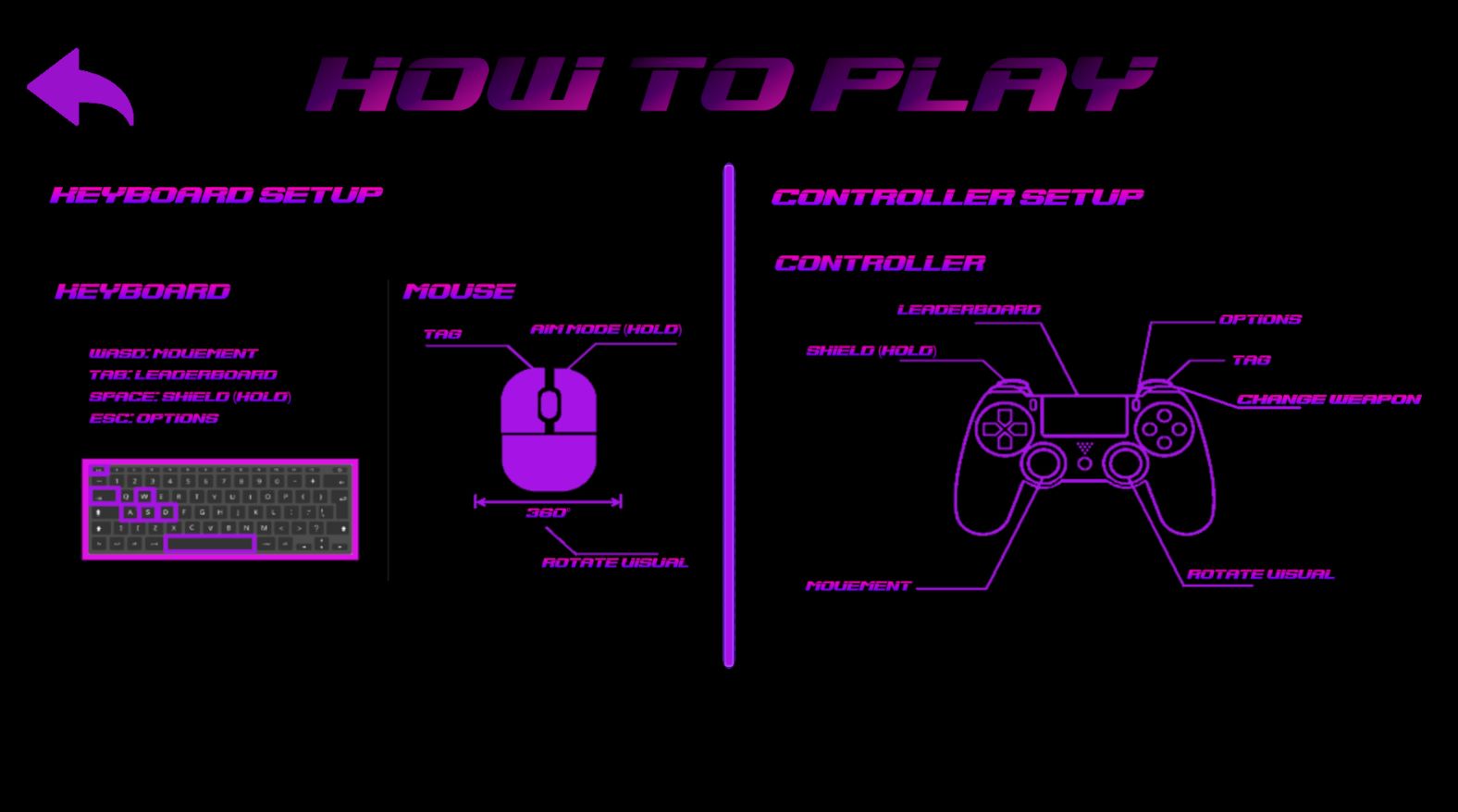


Figura 15 - Scena visualizzazione tasti

La schermata how to play verrà visualizzata quando l'utente clicca sul bottone how to play presente nella scena iniziale, essa è composta dalla mappatura dei tasti sia di mouse e tastiera sia del controller per permettere all'utente di visualizzare i tasti. Inoltre in alto a sinistra è presente un bottone per tornare alla scena iniziale.

### Settings Scene

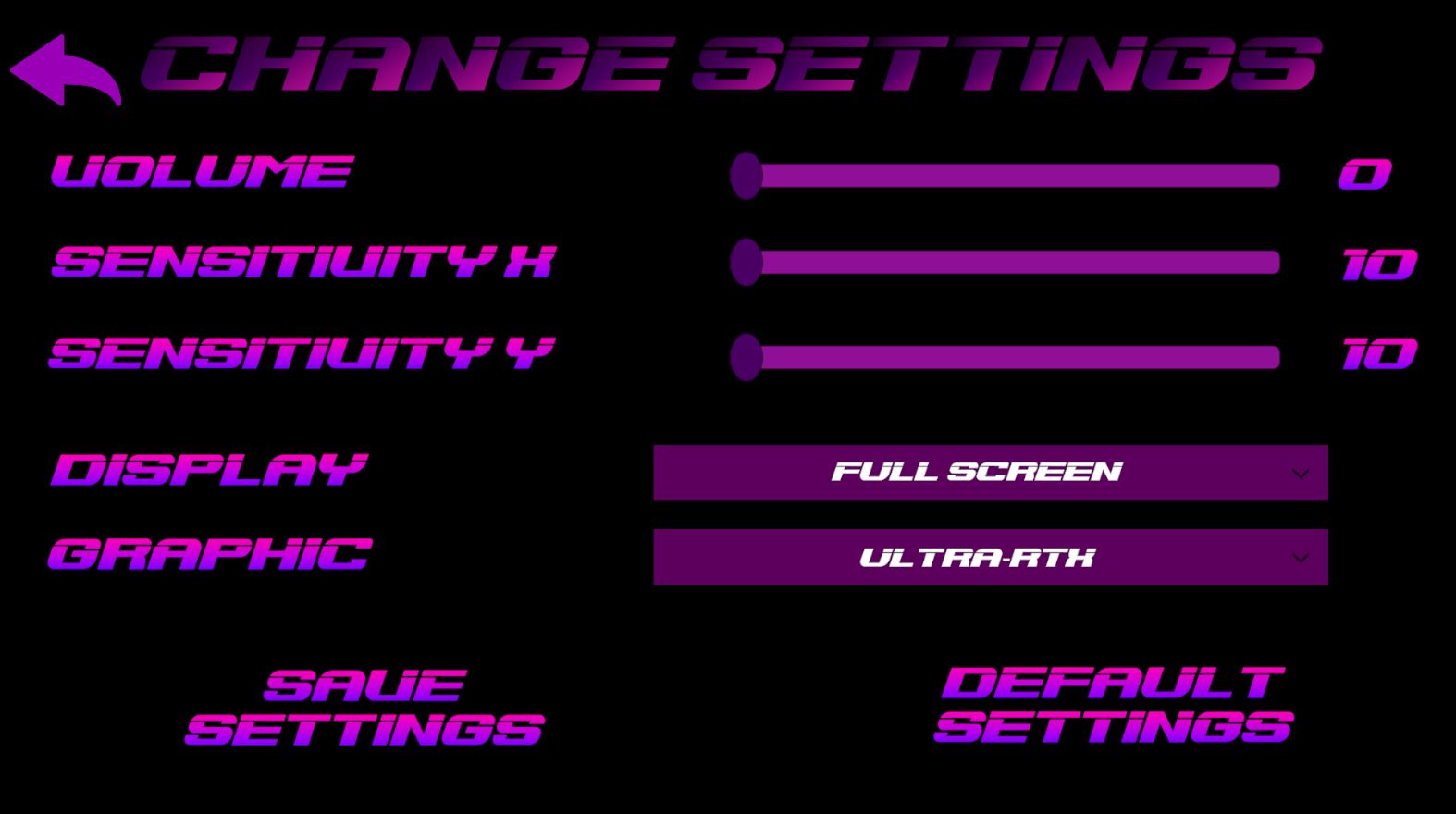


Figura 16 - Scena personalizzazione sensibilità

La scena “Change Settings” viene visualizzata quando l'utente premerà sul bottone settings nella scena iniziale, essa comprende:

* Uno Slider “Volume” che permette di modificare il volume di gioco
* Uno Slider “Sensitivity X” che permette di modificare la sensibilità X
* Uno Slider “Sensitivity Y” che permette di modificare la sensibilità Y
* Un Dropdown “Display” che permette di modificare il tipo di finestra (Full screen o Windowed)
* Un Dropdown “Graphics” che permette di modificare il tipo di garafica
* Un Pulsante “Save Settings” che permette di salvare le impostazioni selezionate
* Un Pulsante “Default Settings” che imposta automaticamente le impostazioni predefinite
* In alto a sinistra un Pulsante che permette di tornare alla scena iniziale

### Match History Scene

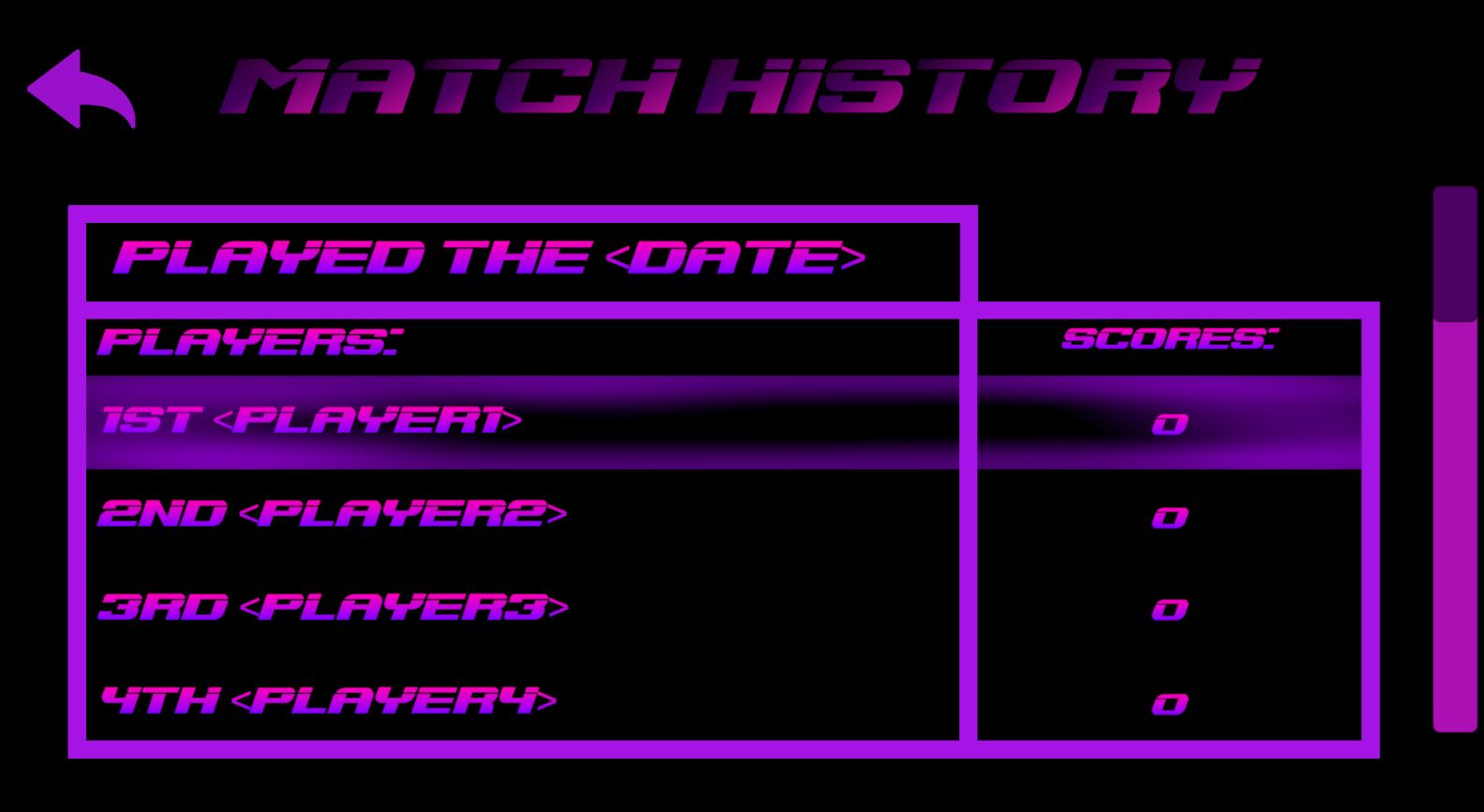


Figura 17 - Scena Partite Passate

La scena “Match History” viene visualizzata quando l'utente preme sul bottone “Match History” nella scena iniziale essa è composta da una serie di tabelle contenenti tutte le informazioni riguardanti le partite passate, inoltre contiene in alto a sinistra un bottone che permette di tornare indietro alla scena iniziale.

### Login Scene



Figura 18 - Scena di Login

La schermata “Login” verrà visualizzata quando l'utente premerà sul pulsante “Login/Logout” presente nella scena iniziale, essa comprende:

* Un Textbox dove l'utente deve inserire la propria email
* Un Textbox dove l'utente deve inserire la password che può visualizzare anche in chiaro tramite il bottone “Show/Hide Password”
* Il pulsante “Sign in” che permette all'utente di accedere all'account
* Il pulsante “Create un account” che permette all'utente di creare un account
* In alto a sinistra un pulsante che permette di tornare indietro alla scena iniziale

### Create Account Scene

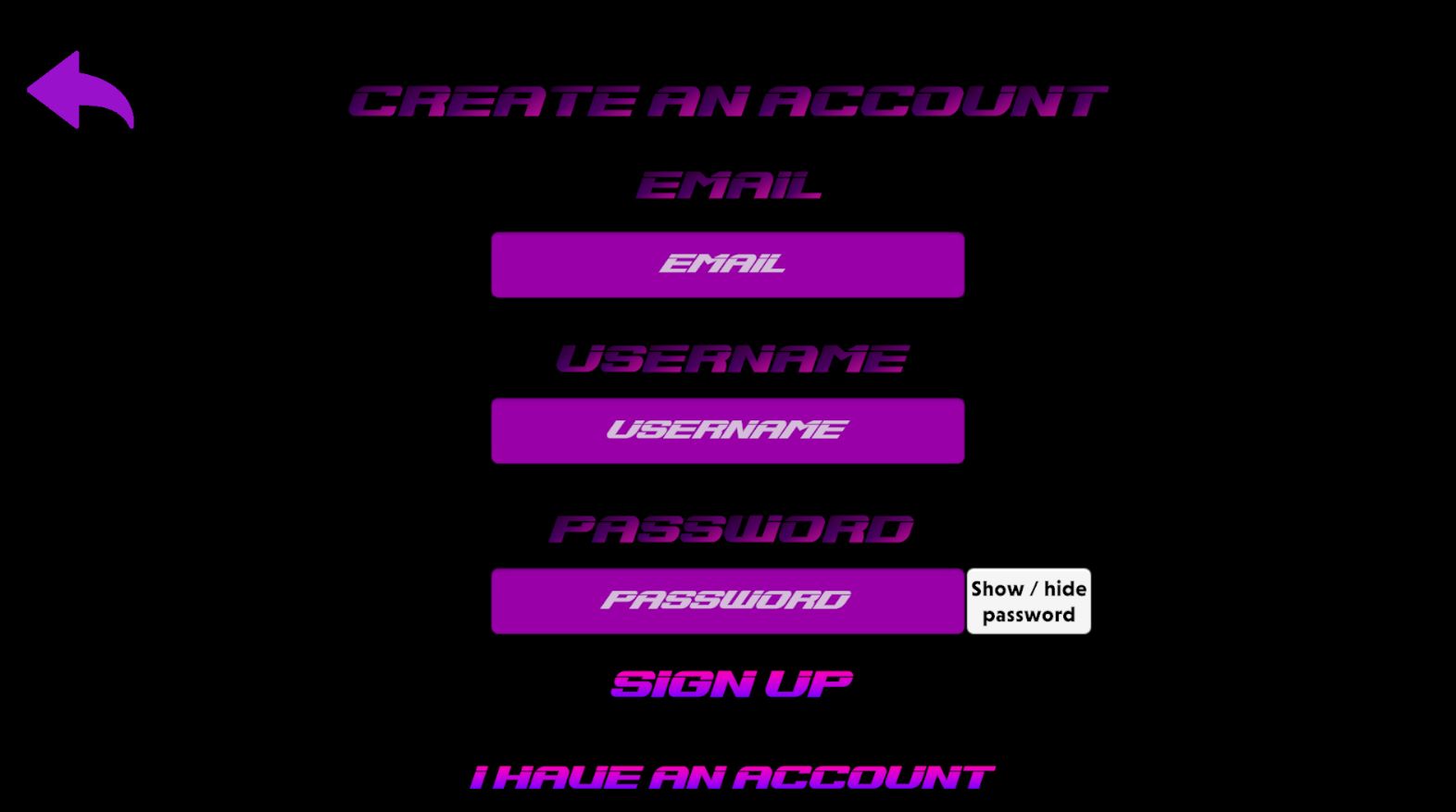


Figura 19 - Scena Creazione Account

La schermata di creazione account viene visualizzata quando l'utente preme sul pulsante “Create an Account” nella scena di login, essa comprende:

* Un Textbox dove l'utente deve inserire la propria email
* Un Texbox dove l'utente deve inserire il proprio nickname
* Un Textbox dove l'utente deve inserire la propria password che può visualizzare anche in chiaro tramite il bottone “Show/Hide Password”
* Il bottone “Sign up” che permette all'utente di creare l'account
* il bottone I have an account che permette all'utente di andare alla scena di login
* In alto a sinistra un pulsante che permette di tornare indietro alla scena iniziale

## Scripts

### NetManager

Questa classe gestisce la creazione di istanze di gioco (quindi gestisce la creazione Host e Client)

#### Metodo Start()

private void Start()

{

textIP.text = "IP: " + GetLocalIPAddress();

string mode = PlayerPrefs.GetString("multiplayerMode");

print(mode);

switch (mode)

{

case "server":

Debug.LogError("Impossible to instance a server! Not Supported!");

Application.Quit();

break;

case "host":

textIP.text += " (HOST)";

//Setto le impostazioni di rete

NetworkManager.Singleton.GetComponent<UnityTransport>().SetConnectionData(

"0.0.0.0", //dove connettersi. è host, quindi non server. lasciare 0.0.0.0

(ushort)6973, //porta di ascolto

PlayerPrefs.GetString("connectIp") //indirizzo ip del host.

);

//Avvio il processo di rete e mi connetto

NetworkManager.Singleton.StartHost();

break;

case "client":

textIP.text += " (CLIENT)";

//Setto le impostazioni di rete

NetworkManager.Singleton.GetComponent<UnityTransport>().SetConnectionData(

PlayerPrefs.GetString("connectIp"), //indirizzo ip del host.-

(ushort)6973 //porta del host

);

//Avvio il processo di rete e mi connetto

NetworkManager.Singleton.StartClient();

break;

}

}

In questo metodo, a seconda del **playerprefs** che si è passato dallo script StartGame.cs viene avviato un *Singleton* per l’host oppure viene reindirizzato al *Singleton*per la creazione del client.

#### GetLocalIPAddress

//serve a ottenere l'indirizzo privato.

//in futuro cambiarlo e ottenere quello pubblic

public static string GetLocalIPAddress()

{

foreach (NetworkInterface ni in NetworkInterface.GetAllNetworkInterfaces())

{

if (ni.NetworkInterfaceType != NetworkInterfaceType.Loopback && ni.OperationalStatus == OperationalStatus.Up)

{

IPInterfaceProperties ipProps = ni.GetIPProperties();

if (ipProps.GatewayAddresses.Count > 0)

{

foreach (UnicastIPAddressInformation ip in ipProps.UnicastAddresses)

{

if (ip.Address.AddressFamily == AddressFamily.InterNetwork)

{

return ip.Address.ToString();

}

}

}

}

}

return "Indirizzo IP locale non trovato!";

}

Questo metodo ritorna l’indirizzo IP locale della scheda ethernet principale (quindi ignora quelle secondarie)

### NetworkMatchManager

Questa classe si occupa di gestire il match lato server, per questo eredita da **NetworkBehavior.**

#### Metodi

* StartMatch: si occupa di avviare il match
* EndMatch: si occupa di terminare la partita
* StartMatchServerRpc: (eseguito solo dal server) 🡪 Si occupa di richiamare una coroutine per avviare la partita
* EndMatchServerRpc: (eseguito solo dal server) 🡪 Si occupa di chiamare la coroutine per terminare la partita.

[ServerRpc]

private void EndMatchServerRpc()

{

DamageManager dm = null;

foreach (GameObject item in GameObject.FindGameObjectsWithTag("Player"))

{

if (item.GetComponent<NetworkPlayer>().IsOwner)

{

dm = item.GetComponent<DamageManager>();

break;

}

}

scores = dm.GetPlayerScores();

SetScoresClientRpc(scores);

isMatchStarted.Value = false;

InsertScoresOnDbClientRpc() }

* StartGameClientRpc: (eseguito solo dal client) 🡪 Si occupa di gestire lo start del match lato client
* InsertScoresOnDbClientRpc: (eseguito solo dal client) 🡪 Si occupa di gestire l’inserimento dei dati dei punteggi nel DB

### ServerLife

Si occupa di gestire la durata della partita, quando la classe NetworkMatchManager avvia la partita il timer scende, quando arriva a zero viene notificata la classe antecedentemente menzionata.

#### Update()

private void Update()

{

if (FindObjectOfType<NetworkMatchManager>().isMatchStarted.Value && IsHost)

{

timeRemaining.Value -= Time.deltaTime;

if(timeRemaining.Value <= 0)

FindObjectOfType<NetworkMatchManager>().EndMatch();

}

if(FindObjectOfType<NetworkMatchManager>().isMatchStarted.Value)

text.text = "Time remaining: " + (int)timeRemaining.Value + " seconds";

}

Questo metodo si occupa di aggiornare il tempo di gioco e di visualizzarlo.

### DamageManager

Questa classe si occupa di gestire lato server tutto quello che riguarda la ricezione del laser, ovvero toglie punti vita al giocatore.

#### Metodi

* Awake: istanzia tutte le **NetworkVariables**
* Update: aggiorna la vista della vita lato client
* GetPlayerScores: ottiene tutti i punteggi dei giocatori
* PlayerHittedServerRpc: (eseguito dal server) 🡪 è richiamato dai client quando sanno di aver colpito qualcuno con il laser
* AddDeathPointsServerRpc: (eseguito dal server) 🡪 è richiamato dai client quando un client sa che ha fatto terminare i punti vita ad un altro giocatore. Aggiunge i punti
* UpdateTextClientRpc: (eseguito dai client) 🡪 è richiamato dal metodo Update() e aggiorna i testi dei punti vita
* PlayerTaggedBySomeoneClientRpc (eseguito dai client) 🡪 è richiamato quando un player colpisce quancuno

### NetworkPlayer

Questa classe si occupa di gestire il singolo player all’interno della rete di **Netcode for GameObjects**

#### Start

private void Start()

{

\_camera = GetComponentInChildren<Camera>();

\_audioListener = GetComponentInChildren<AudioListener>();

\_playersManagement = FindObjectOfType<PlayersManagement>();

if (!IsOwner) //Se chi richiama questo metodo non è il proprietario allora...

{

\_audioListener.enabled = false; //...viene disabilitato l'audiolistener

\_camera.enabled = false; //...viene disabilitata la camera

\_playersManagement.enabled = false; //...viene disabilitato questo script

//vengono disabilitati questi componeti

GetComponent<NetworkObject>().enabled = false;

GetComponent<NetworkPlayer>().enabled = false;

GetComponent<DamageManager>().enabled = false;

GetComponent<PlayerMotor>().enabled = false;

GetComponent<PlayerLook>().enabled = false;

return;

}

NetworkId = GetComponent<NetworkObject>().NetworkObjectId; //vado a prendere dal server il mio id

UserDbId = PlayerPrefs.GetInt(PlayerPreference.USER\_ID);

\_playersManagement.ClientConnectedServerRpc(NetworkId); //avviso il server che mi sono connesso

}

In questo metodo vengono cercati ed eventualmente disabilitati componenti al GameObject. Vengono disabilitato solo se il GameObject del player è un clone (ovvero non quello che sto controllando).

#### Metodi

* Update: si occupa di controllare se il server si sta disconnettendo
* DisconnectAllClientsServerRpc: si occupa di disconnetter ogni client se il server crasha o termina la partita
* EndAllConnectionsClientRpc: è il metodo richiamato da quello precedente. Disconnette il client in maniera effettiva
* *Override* OnNetworkDespawn: è l’override del metodo bas:

public override void OnNetworkDespawn()

{

if (IsClient && IsLocalPlayer)

{

SceneManager.LoadScene((int)SceneToId.gameEnded);

NetworkManager.Singleton.Shutdown();

}

}

### PlayersManagement

Questa classe si occupa di gestire i players in rete

#### ClientConnectedServerRpc

[ServerRpc(RequireOwnership = false)]

public void ClientConnectedServerRpc(ulong clientId) //il player si è connesso

{

GameObject[] rawList = GameObject.FindGameObjectsWithTag("Player");

GameObject clientObj = null;

foreach (GameObject obj in rawList)

{

if(obj.GetComponent<NetworkObject>().NetworkObjectId == clientId)

{

clientObj = obj;

break;

}

}

try

{

players.Add(clientId, clientObj);

UpdateDictCount();

Debug.Log("Client connected, id: " + clientId);

ulong cId = clientId;

if (clientId == 1)

cId = 0;

else

cId -= (cId - 1);

switch (cId)

{

case 0:

player1Id.Value = cId;

break;

case 1:

player2Id.Value = cId;

break;

case 2:

player3Id.Value = cId;

break;

case 3:

player4Id.Value = cId;

break;

default:

Debug.LogError("Error, clientId > 4! Actual ID: " + cId);

break;

}

Debug.Log("Joined client " + cId);

}

catch { }

}

Questo metodo è chiamato dal client quando si connette

#### ClientDisconnectedServerRpc

[ServerRpc(RequireOwnership = false)]

public void ClientDisconnectedServerRpc(ulong clientId, bool serverCrashed) //client disconnesso

{

if (!IsOwner)

return;

Debug.Log("Client disconnected, id: " + clientId);

if (!serverCrashed)

{

players.Remove(clientId);

UpdateDictCount();

return;

}

Debug.LogError("Server crashed!");

}

Questo metodo viene richiamato quando il client si disconnette. Server anche a sapere se la disconnessione è dovuta al server che è crashato.

#### GetPlayers

public IDictionary<ulong, GameObject> GetPlayers()

{

return players;

}

Questo metodo ritorna il dizionario di player (contiene il loro GameObject e il NetworkObjectId 🡪 è un id che viene assegnato al GameObject per riconoscerlo in rete).

### NetMatchManager

Questo script gestisce lato client il match.

#### Metodi

* Start: si occupa di una lista di players solo se viene eseguito dall’istanza del giocatore e non dai cloni

### NetLaserSystem

Questa classe è quasi identica alla sua controparte non net (PlayerLaserSystem) ma con la differenza che qui quando un player viene colpito esegue questo codice:

// If it's another player/AI deal damage

if (rayHit.collider.CompareTag("Enemy") || rayHit.collider.CompareTag("Player"))

{

//Tells server to decrement player's life

FindObjectOfType<DamageManager>().PlayerHittedServerRpc(

rayHit.collider.GetComponent<NetworkPlayer>().NetworkObjectId,

transform.parent.parent.parent.GetComponent<NetworkPlayer>().NetworkObjectId);

// Display bullet holes effect very briefly

DisplayBulletHoleEffect(0.2f);

}

### MapGenerator

Questa classe si occupa di generare la mappa di gioco in modo casuale.

#### GenerateMapServerRpc

[ServerRpc(RequireOwnership = false)]

public void GenerateMapServerRpc()

{

for (int i = 0; i < walls.Length; i++)

{

wallsBefore[i] = walls[i]; //copio l'array precedente

walls[i] = UnityEngine.Random.Range(0, 100) < PROBABILITY; //genero la nuova mappa

}

serializedWalls.Value = ArrayToString(walls, ','); //serializzo l'array nella net var

serializedWallsBefore.Value = ArrayToString(wallsBefore, ','); //serializzo l'array nella net var

StartCoroutine(SendCommandToClientCooldown()); //attesa

}

Questo metodo server per generare l’array. Lo serializziamo in una stringa perché nei metodi ClientRpc / ServerRpc e nelle NerworkVariables non possono essere passati oggetti ma solo i tipi primitivi e alcune classi di Unity serializzabili.

#### ApplyConfigurationClientRpc

[ClientRpc]

private void ApplyConfigurationClientRpc()

{

string[] serializedWallsArray = serializedWalls.Value.ToString().Split(",");

string[] serializedWallsBeforeArray = serializedWallsBefore.Value.ToString().Split(",");

walls = new bool[WALL\_COUNT];

wallsBefore = new bool[WALL\_COUNT];

wallInfoDebug.text = "ClientWalls:" + "\r\n" + serializedWalls.Value; //debug

for (int i = 0; i < walls.Length; i++) //vado a riformare l'array di bool dalla stringa serializzata

{

if (serializedWallsArray[i].Equals("True"))

{

walls[i] = true;

}

else

{

walls[i] = false;

}

if (serializedWallsBeforeArray[i].Equals("True"))

{

wallsBefore[i] = true;

}

else

{

wallsBefore[i] = false;

}

}

for (int i = 0; i < walls.Length; i++) //controllo lo stato precedente e decido se abbassare o alzare il muro

{

if (walls[i] == wallsBefore[i])

{

continue;

}

if(walls[i] && !wallsBefore[i])

{

StartCoroutine(TranslateObject(wallsPhysic[i], .2f, Vector3.zero, 0, deltaH, 0));

}

else

{

StartCoroutine(TranslateObject(wallsPhysic[i], .2f, Vector3.zero, 0, -deltaH, 0));

}

}

}

Questo metodo si occupa di *applicare* la configurazione generata dal metodo precedente.

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

### Test Case Obbligatori

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-001  REQ-01 | Nome: | Movimento Giocatore |
| Descrizione: | L'utente può usare dei comandi di tastiera o joystick per muovere il personaggio. | | |
| Procedura: | 1. L’utente apre il gioco 2. Inizia una partita 3. Si sposta con controller o tastiera | | |
| Risultati attesi: | L’utente è in grado di spostarsi in giro per la mappa sia con controller che con tastiera | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-002  REQ-02 | Nome: | Interfaccia grafica |
| Descrizione: | L'utente può navigare attraverso l'interfaccia di gioco usando tastiera o joystick. | | |
| Procedura: | 1. L’utente apre il gioco 2. Seleziona I vari oggetti della schermata | | |
| Risultati attesi: | L’utente è in grado di interagire con gli oggetti delle schermate sia con controller che con tastiera | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-003  REQ-03 | Nome: | Ambiente Partita |
| Descrizione: | Deve essere presente una mappa di gioco. | | |
| Procedura: | 1. L’utente apre il gioco 2. Preme sul bottone “Play” 3. Entra in partita | | |
| Risultati attesi: | L’utente visualizza la mappa di gioco con il personaggio | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-004  REQ-04 | Nome: | Sito web |
| Descrizione: | Deve essere presente un sito web | | |
| Procedura: | 1. L’utente apre un browser 2. Entra nel sito web | | |
| Risultati attesi: | L’utente visualizza il sito web | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-005  REQ-13 | Nome: | Creazione account |
| Descrizione: | L’utente crea un account su Unity | | |
| Procedura: | 1. L’utente entra nel gioco 2. Crea un account | | |
| Risultati attesi: | L’utente ha un account personale su Unity | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-006  REQ-05 | Nome: | Interazioni giocatore |
| Descrizione: | L'utente può mirare e puntare un laser ed eventualmente parare i laser degli altri usando i comandi di tastiera o joystick. | | |
| Procedura: | 1. L’utente avvia una partita 2. Mira e punta un avversario 3. Si para con lo scudo | | |
| Risultati attesi: | L’utente è in grado di utilizzare tutte le mosse del personaggio | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-007  REQ-06 | Nome: | Ostacoli casuali |
| Descrizione: | I muri si generano in maniera casuale all'inizio di ogni partita usando l'algoritmo di Manhattan Mapper. | | |
| Procedura: | 1. L’utente avvia una partita 2. Visualizza la posizione casual dei muri | | |
| Risultati attesi: | L’utente può giocare su delle mappe tutte diverse | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-008  REQ-07 | Nome: | HUD |
| Descrizione: | L'Heads-Up Display (HUD) mostrerà varie informazioni durante la partita. | | |
| Procedura: | 1. L’utente avvia una partita 2. L’utente guarda il tempo scorrere | | |
| Risultati attesi: | L’utente può giocare per un tot di tempo una determinate partita | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-009  REQ-07 | Nome: | Incremento punteggio leaderboard |
| Descrizione: | Ad ogni colpo del puntatore laser su un avversario il punteggio aumenta | | |
| Procedura: | 1. L’utente avvia una partita 2. Punta un avversario 3. Guarda nella leaderboard il punteggio | | |
| Risultati attesi: | L’utente può visualizzare il suo punteggio incrementato ad ogni colpo corretto | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-010  REQ-08 | Nome: | AI Nemico |
| Descrizione: | Un'AI che si comporta come un giocatore. | | |
| Procedura: | 1. L’utente avvia una partita 2. L’utente gioca senza altri 3 giocatori 3. Vengono generate le AI che giocano come una persona | | |
| Risultati attesi: | L’utente può giocare anche con solamente AI | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-011  REQ-09 | Nome: | Database |
| Descrizione: | Il database contiene tutte le tabelle utili per lo storage di dati e per il multiplayer. | | |
| Procedura: | 1. L’utente avvia una partita 2. Punta un avversario 3. Controlla che il punteggio si sia incrementato | | |
| Risultati attesi: | L’utente può visualizzare il suo punteggio incrementato ad ogni colpo corretto | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-012  REQ-09 | Nome: | Tabella Leaderboard Globale |
| Descrizione: | tabella consultabile dal sito web dove verranno mostrati i migliori giocatori di sempre con il punteggio | | |
| Procedura: | 1. L’utente Accede al sito web 2. Va nella sezione leaderboard 3. Controlla la leaderboard | | |
| Risultati attesi: | L’utente può visualizzare la classifica dei migliori punteggi | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case:  Riferimento: | TC-013  REQ-10 | Nome: | Multiplayer |
| Descrizione: | Il giocatore può collegarsi ad una sessione con altri giocatori. | | |
| Procedura: | 1. L’utente seleziona una lobby 2. Avvia una partita 3. Vede gli altri giocatori | | |
| Risultati attesi: | L’utente può giocare con altri giocatori | | |

## Risultati test

|  |  |  |  |
| --- | --- | --- | --- |
| TC-xxx | Risultato | Descrizione errore | Risolto? |
| TC-001 | Passato |  |  |
| TC-002 | Passato |  |  |
| TC-003 | Passato |  |  |
| TC-004 | Non Passato | Non fatto in accordo con il Committente |  |
| TC-005 | Passato |  |  |
| TC-006 | Passato |  |  |
| TC-007 | Passato |  |  |
| TC-008 |  |  |  |
| TC-009 |  |  |  |
| TC-010 | Passato |  |  |
| TC-011 | Passato |  |  |
| TC-012 |  |  |  |
| TC-013 |  |  |  |

## Mancanze/limitazioni conosciute

### Sito WEB

L’idea iniziale era che il prodotto sarebbe stato accompagnato da un sito web dove sarebbe stato possibile gestire il proprio account e vedere i risultati dei match (anche in live). Durante il progetto ci siamo accorti di un nostro piccolo errore nella progettazione, quindi non saremo riusciti a terminare la parte web del progetto. Abbiamo quindi deciso, in accordo con il committente, di eliminare il sito web e di integrare queste funzioni direttamente nell’applicativo.

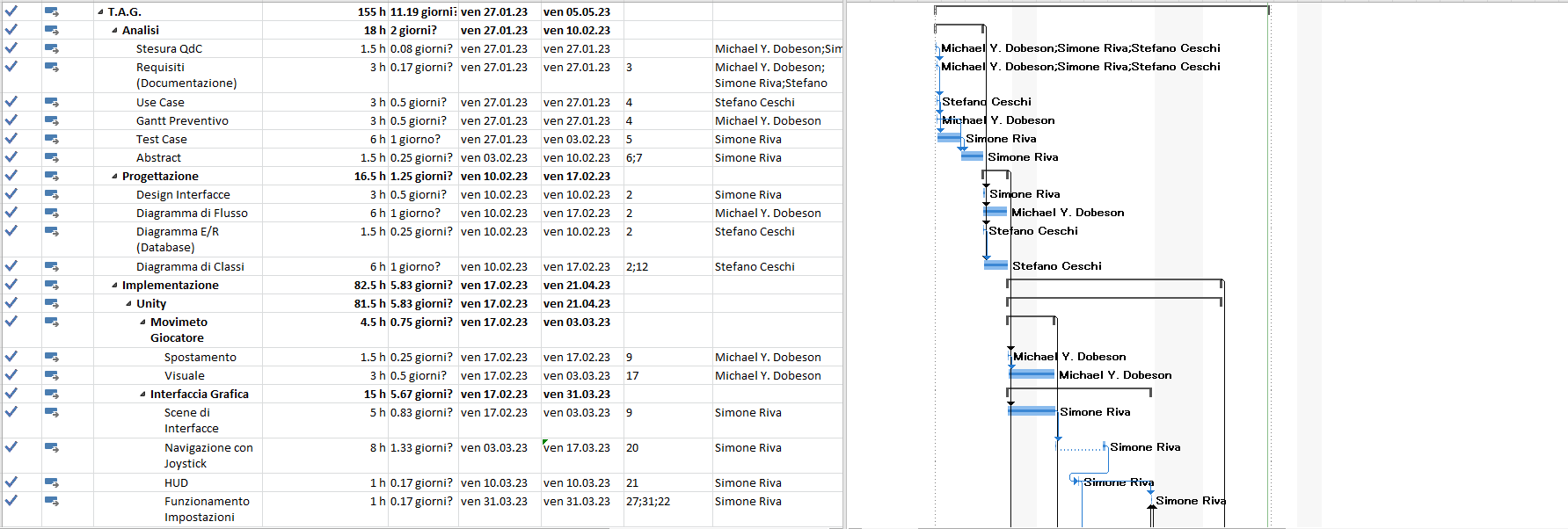
### Funzionamento Impostazioni

Per quanto riguarda il funzionamento delle impostazioni non abbiamo fatto in tempo ad implementarlo, i pulsanti e gli slider funzionano ma non impostano al giocatore le cose.

### Funzionamento in WAN

L’idea era quella di fare funzionare tutto in WAN, però non siamo riusciti perché da scuola non si può accedere all’esterno e l’API che volevamo usare non funzionava.

# Consuntivo



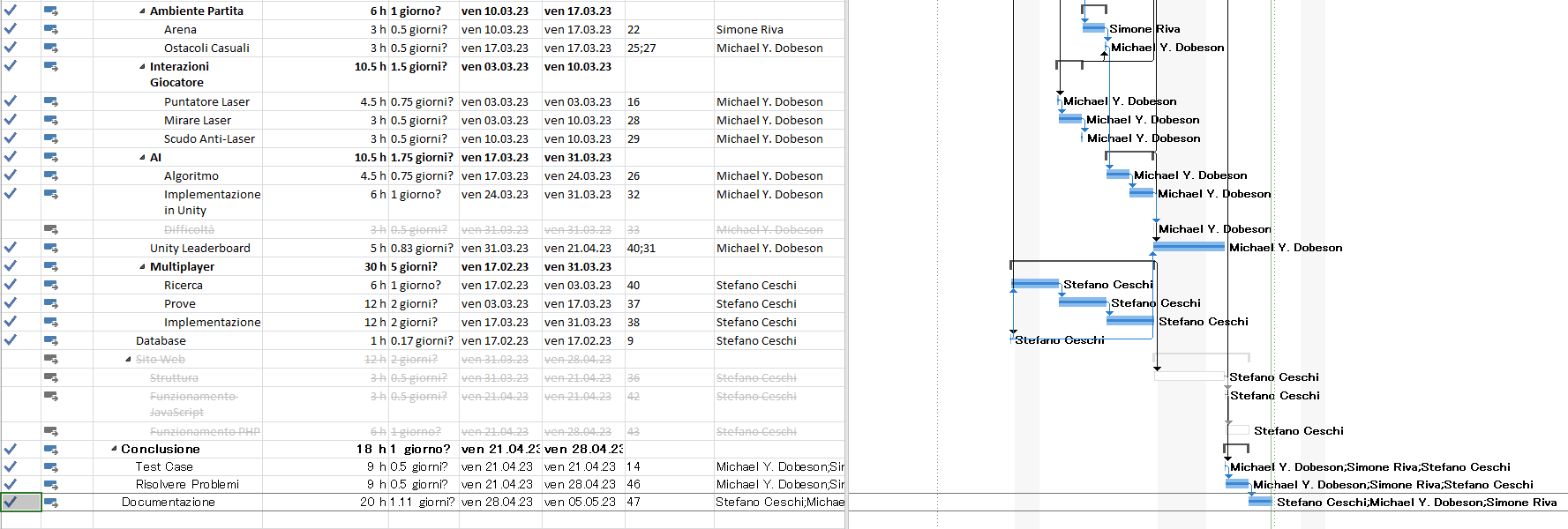


Figura 22 - Gantt Consuntivo

Al termine del progetto il Gantt è cambiato, infatti abbiamo impiegato il doppio del tempo sia per la navigazione con il controller sia per l’implementazione del multiplayer.  
Inoltre abbiamo impiegato meno tempo per la realizzazione del DataBase.

# Conclusioni

## Sviluppi futuri

In futuro si può migliorare la grafica, applicare il cell-shading, sviluppare il sito web, creare un sistema per un LAN party e sicuramente correzioni di bug.

## Considerazioni personali

### Stefano Ceschi

Vorrei concludere questo progetto dicendo prima di tutto che ho veramente imparato molto, soprattutto quello che riguarda il networking dei GameObjects. Questo progetto mi è servito particolarmente per capire che nessuno di noi tre è onnisciente, mi spiego meglio: siamo partiti con l’idea di fare un giocone, pieno di funzioni e con addirittura un sito web, ma ci siamo resi conto circa 3 settimane prima che non avremmo finito se avessimo continuato a quel ritmo. Io mi sono forse concentrato troppo sullo sviluppo e poco sulla documentazione, ne sono consapevole ma voglio ribadire che, a mio avviso, è anche così che si imapara.

# Glossario

|  |  |
| --- | --- |
| Termine | Descrizione |
| AI | Intelligenza Artificiale |
| Cell-shading | Stile non fotorealistico di visualizzazione di modelli 3D |
| Framerate | Sequenza di fotogrammi |
| Lag | Tempi di scambio dati troppo lunghi risetto a quelli desiderati |
| Leaderboard | Classifica |
| GameObject | Oggetto di gioco presente in Unity |
| NetworkBehavior | È la classe base di Netcode (da cui ogni script eredita) |
| ServerRpc | Dice al PC che quel metodo deve essere eseguito solo dal server / host |
| ClientRpc | Dice al PC che quel metodo deve essere eseguito solo dal client |
| NetworkVariable | Variabile che è sincronizzata nella rete Netcode |

# Bibliografia

## Sitografia

<https://lucid.app> 28.04.2023

<https://docs.unity3d.com> 28.04.2023

<https://stackoverflow.com> 28.04.2023

<https://learn.microsoft.com/en-us/docs/> 28.04.2023

<https://api.ipify.org/> 28.04.2023

<https://mockflow.com/> 28.04.2023

<https://www.youtube.com/watch?v=Y3WNwl1ObC8> 14.04.2023

# Allegati

<https://github.com/SimoneRivaSAMT/TAG> 05.05.2023

<https://trello.com/b/T5sMZ7dY/development> 05.05.2023