

# Integrazione di funzionalità interattive in un ambiente di realtà virtuale multi-utente

**Relatore:** *Prof. Giuseppe Vizzari*

**Co-relatrice:** *Prof.ssa Daniela Briola*

**Relazione della prova finale di**  
*Ghezzi Davide*  
*Matricola 846114*

Anno Accademico 2021 - 2022

*Alla mia famiglia che mi ha sempre supportato ed incoraggiato  
nelle mie scelte e nello studio.*

*Ai miei amici che ogni giorno hanno condiviso con me momenti di gioia,  
sostenendomi nei periodi di sconforto e non voltandomi mai le spalle.*

*A tutti i professori e ai ricercatori che hanno contribuito alla mia  
formazione culturale e professionale.*

# Indice

<b>Indice</b>	<b>3</b>
<b>Elenco delle figure</b>	<b>5</b>
<b>Glossario</b>	<b>7</b>
<b>1 Introduzione</b>	<b>8</b>
1.1 Requisiti . . . . .	9
1.2 Realizzazione . . . . .	9
1.3 Organizzazione dei Capitoli . . . . .	10
<b>2 Analisi e Progettazione</b>	<b>11</b>
2.1 Specifica dei Requisiti . . . . .	11
2.2 Analisi dei casi d'uso . . . . .	12
2.2.1 Autenticazione . . . . .	14
2.2.2 Create Room . . . . .	14
2.2.3 Join Room . . . . .	15
2.2.4 Chat Vocale . . . . .	15
2.2.5 Chat di testo . . . . .	15
2.2.6 Richiesta di parlare . . . . .	16
2.2.7 Impostazioni . . . . .	16
2.2.8 Mute/Unmute Utente . . . . .	16
2.2.9 Creazione/Eliminazione di bandierine . . . . .	17
2.3 Diagrammi Workflow . . . . .	18
2.3.1 Autenticazione . . . . .	19
2.3.2 Create Room . . . . .	20
2.3.3 Join Room . . . . .	21
<b>3 Tecnologie Abilitanti</b>	<b>23</b>
3.1 Realtà Virtuale . . . . .	23
3.1.1 Headset . . . . .	24
3.2 Unity . . . . .	25
3.2.1 Interfaccia Utente . . . . .	26
3.2.1.1 Hierarchy . . . . .	27

3.2.1.2	Game and Scene . . . . .	27
3.2.1.3	Inspector . . . . .	28
3.2.1.4	Project . . . . .	29
3.2.1.5	Console . . . . .	30
3.3	C# . . . . .	31
3.3.1	.NET Framework . . . . .	31
3.3.2	MonoBehaviour . . . . .	33
3.3.3	Librerie Utilizzate . . . . .	34
3.3.3.1	Librerie Interne . . . . .	34
3.3.3.2	Librerie Photon . . . . .	35
3.4	Photon Engine . . . . .	35
3.4.1	Implementazione di PUN in Unity . . . . .	36
<b>4</b>	<b>Funzionalità dell'utente</b>	<b>38</b>
4.1	XR Origin . . . . .	38
4.2	Creazione e distruzione di oggetti . . . . .	39
4.2.1	Object_Spawner . . . . .	40
4.2.2	Object_Destroyer . . . . .	42
4.3	Unity Canvas e Interfaccia Utente . . . . .	45
4.4	Chat interattiva . . . . .	45
4.5	Abilitazione e disabilitazione audio utenti . . . . .	49
<b>5</b>	<b>Dimostrazioni d'uso</b>	<b>52</b>
5.1	Identificazione . . . . .	52
5.2	Lobby . . . . .	54
5.3	Generic Room . . . . .	55
5.3.1	Chat di testo e Impostazioni . . . . .	57
5.3.2	Creazione/Eliminazione di bandierine . . . . .	57
<b>6</b>	<b>Sviluppi futuri</b>	<b>61</b>
6.1	Sviluppi Futuri . . . . .	61
6.1.1	Visibilità dei Raggi . . . . .	61
6.1.2	Avatar realistici . . . . .	61
6.1.3	Ambientazione esistente . . . . .	62
6.1.4	Sistema di Registrazione e Autenticazione . . . . .	62
<b>Bibliografia e Sitografia</b>		<b>63</b>

# Elenco delle figure

2.1	Diagramma completo dei casi d'uso . . . . .	13
2.2	Diagramma del caso d'uso Autenticazione . . . . .	14
2.3	Diagramma dei casi d'uso Join Room e Create Room . . . . .	15
2.4	Diagramma del caso d'uso Text Chat che include Richiesta di Parlare . . . . .	16
2.5	Diagramma dei casi d'uso Voice Chat, Mute/Unmute Utente, Impostazioni . . . . .	17
2.6	Diagramma del caso d'uso Creazione/Eliminazione Bandierina . . . . .	18
2.7	Diagramma Workflow di Autenticazione . . . . .	19
2.8	Diagramma Workflow di Create Room . . . . .	20
2.9	Diagramma Workflow di Join Room . . . . .	21
3.1	Oculus Quest 2 con i relativi controller . . . . .	24
3.2	Logo di Unity . . . . .	25
3.3	Interfaccia utente di Unity 2021.3.15f1 . . . . .	26
3.4	Hierarchy Panel di Unity Editor . . . . .	27
3.5	Game and Scene Panel di Unity Editor . . . . .	28
3.6	Inspector Panel di Unity Editor . . . . .	29
3.7	Project Panel di Unity Editor . . . . .	30
3.8	Console Panel di Unity Editor . . . . .	30
3.9	Architettura della piattaforma .NET Framework . . . . .	32
3.10	Tempismo di chiamata dei metodi <i>Awake</i> e <i>Start</i> . . . . .	33
3.11	Struttura di Photon Engine . . . . .	35
3.12	Dashboard delle applicazioni Photon Realtime . . . . .	36
3.13	Photon Server Settings . . . . .	37
4.1	XR Origin nella Hierarchy del progetto . . . . .	38
4.2	Componenti del LeftHand Controller . . . . .	39
4.3	Modello della bandierina che viene creata all'interno della scena . . . . .	40
4.4	Modello del pannello su cui vengono create le bandierine . . . . .	40
4.5	Object_Spawner in dettaglio . . . . .	41
4.6	Diagramma di flusso del metodo <i>Update()</i> della classe Object_Spawner . . . . .	42
4.7	Object_Destroyer e DestroyObject in dettaglio . . . . .	43
4.8	Diagramma di flusso del metodo <i>Update()</i> della classe Object_Destroyer . . . . .	44
4.9	Join Room Canvas . . . . .	45
4.10	Chat Room Canvas . . . . .	46

4.11	Open Chat Canvas . . . . .	46
4.12	Diagramma di sequenza per Join Room . . . . .	47
4.13	Diagramma di sequenza per Open Chat . . . . .	48
4.14	Diagramma di sequenza per Chat Room . . . . .	49
4.15	Apri Impostazioni Canvas . . . . .	50
4.16	Schermata Utenti Canvas . . . . .	50
4.17	Diagramma di sequenza per l'attivazione e disattivazione del microfono . . . . .	51
5.1	Schermata di errore per lo studente . . . . .	52
5.2	Schermata di errore per il docente . . . . .	53
5.3	Lobby docente . . . . .	54
5.4	Lobby studente . . . . .	55
5.5	Visualizzazione fra utenti . . . . .	56
5.6	Funzionamento della Chat e delle Impostazioni . . . . .	57
5.7	Scena prima del piazzamento della bandierina . . . . .	58
5.8	Scena dopo il piazzamento della bandierina lato docente . . . . .	58
5.9	Scena dopo il piazzamento della bandierina lato studente . . . . .	59
5.10	Scena prima della distruzione della bandierina . . . . .	59
5.11	Scena dopo la distruzione della bandierina lato docente . . . . .	60
5.12	Scena dopo la distruzione della bandierina lato studente . . . . .	60

# Glossario

**Asset Store** Sito Web da cui si scaricano ed installano i pacchetti Unity non presenti nello Unity Registry (<https://assetstore.unity.com/>)

**Database** In informatica, archivio di dati strutturato in modo da razionalizzare la gestione e l'aggiornamento delle informazioni e da permettere lo svolgimento di ricerche complesse

**File System** Sistema di archiviazione del sistema operativo. Ha la funzione di gestire lo spazio nelle memorie ausiliarie del computer (es. hard disk, disco fisso esterno, pen drive, dvd-rom, ecc...)

**Networking** In informatica, sistema di collegamento in rete di più elaboratori e utenti, comprendente le piattaforme, i sistemi operativi, i protocolli e le architetture di rete

**Render** Il processo che permette di ottenere, a partire da un modello tridimensionale elaborato al computer, un'immagine artificiale molto realistica

**Runtime System** Il runtime system di un programma (o di un linguaggio di programmazione) è l'insieme dell'hardware e del software necessario come piattaforma per l'esecuzione di quel programma (o dei programmi scritti in quel linguaggio)

**Script** Programma o sequenza di istruzioni che viene interpretata o portata a termine da un altro programma (invece che dal processore come nei linguaggi compilati)

**Unity Registry** Sezione dell'Editor di Unity da cui si scaricano ed installano determinati pacchetti

# Capitolo 1

## Introduzione

Nel corso degli ultimi vent'anni abbiamo assistito ad uno sviluppo esponenziale della tecnologia che ha portato una fetta sempre più grande di pubblico a disporre di hardware con potenza di calcolo elevata ad un costo relativamente contenuto.

Basti pensare, ad esempio, alle TV o telefoni cellulari disponibili agli inizi degli anni 2000 che non possono assolutamente competere con i rispettivi prodotti disponibili attualmente sul mercato.

Di conseguenza, le persone al giorno d'oggi hanno anche a disposizione software molto più complessi e avanzati rispetto al recente passato, grazie ai quali possono affacciarsi sempre più frequentemente a nuove tecnologie.

Una delle tecnologie che ha prepotentemente preso piede negli ultimi anni è la **Realtà Virtuale**.

La Realtà Virtuale (*Virtual Reality*) è una realtà simulata, un mondo digitale dove si viene immersi indossando un apposito visore, che avvolge totalmente l'utente, andando a mascherare del tutto la percezione, quantomeno visiva, del mondo fisico intorno a lui.

È una tecnologia impiegata soprattutto in ambito videoludico, ma è molto utile anche in ambito didattico, perché è possibile vivere delle esperienze educative grazie alla riproduzione fedele di luoghi storici come monumenti, luoghi di culto, siti archeologici oppure luoghi naturali come fondi oceanici, sentieri montuosi, siti geologici.

Al giorno d'oggi, esistono molteplici applicazioni in realtà virtuale in grado di far vivere all'utente le esperienze viste in precedenza e in cui può anche muoversi per esplorare l'ambiente o interagire con esso.<sup>1</sup>

Queste applicazioni, seppur molto utili per l'apprendimento didattico individuale, presentano un grosso problema: non supportano la presenza di più utenti, ma, come ben sappiamo, in un'aula scolastica solitamente sono presenti un docente e circa venti studenti.

Di conseguenza, le caratteristiche principali che un'applicazione multi-utente deve avere sono:

- Percezione contemporanea degli utenti e delle loro azioni;
- Interazione con l'ambiente, con modifiche visibili a tutti;
- Possibilità di interagire con gli altri utenti attraverso una chat di testo o vocale (data la difficoltà d'uso di meccanismi di input-output tradizionali mentre si utilizza un visore).

---

<sup>1</sup><https://www.lifewire.com/virtual-reality-tourism-4129394> in questo sito web sono illustrate alcune applicazioni di questo tipo

Lo scopo del progetto è innanzitutto quello di analizzare le soluzioni tecnologiche e programmatiche per risolvere il problema della mono-utenza, con il fine di rendere effettivamente utili questo tipo di applicazioni per l'apprendimento scolastico, infine quello di ricreare un ambiente che mostri il funzionamento delle suddette soluzioni tramite la realizzazione di un prototipo.

## 1.1 Requisiti

Con le premesse esposte in precedenza, si è voluto realizzare un prototipo di un ambiente di realtà virtuale multi-utente che potesse ospitare circa una ventina utenti, di cui un docente.

Ogni utente ha la percezione visiva e uditiva di ciò che lo circonda, compresi gli altri utenti, per ricreare al meglio la sensazione di presenza e partecipazione all'interno dell'aula virtuale.

Inoltre, al docente è stata data la possibilità di poter disattivare o attivare il microfono di uno o più utenti a suo piacimento ed è stata introdotta anche la possibilità per ogni altro utente di richiedere la parola, proprio come uno studente quando in un aula scolastica '*alza la mano*'.

La grossa differenza rispetto ad un'aula scolastica risiede nell'ambiente in cui tutti gli utenti sono immersi.

In questo prototipo gli utenti verranno immersi in un ambiente naturale con un castello medioevale esplorabile, questo modello è stato scelto semplicemente per dare un'idea delle possibilità grafiche e farsi un'idea dei costi computazionali, senza aver valutato approfonditamente un contesto disciplinare formativo.

L'ambiente in questione potrebbe essere utile, per esempio, per una lezione di storia, arte o architettura.

Questa particolarità ha portato alla luce un'ulteriore requisito, ovvero la possibilità per il docente di interagire con l'ambiente circostante.

Per il docente, durante la simulazione, può essere molto utile segnalare o indicare particolari luoghi all'interno dell'ambiente, a tale scopo è stata pensata la possibilità di poter piazzare e rimuovere delle bandierine dai suddetti luoghi.

Per concludere, visto la mancata possibilità di disporre di un ambiente virtuale che riproduce fedelmente un ambiente reale, l'ambiente sviluppato è a scopo illustrativo e ancora allo stato primordiale.

## 1.2 Realizzazione

Il prototipo è stato realizzato in collaborazione con due colleghi del Dipartimento di Informatica Sistemistica e Comunicazione dell'Università degli studi di Milano-Bicocca, Emanuele Sapiro e Lorenzo Iacopetta.

Le prime fasi di sviluppo sono state caratterizzate da ricerche approfondite che hanno poi portato ad una suddivisione dei compiti per accelerare la fase di sviluppo del software.

In particolare, i miei colleghi hanno approfondito la parte di *Networking*, cioè di connessione tra i dispositivi degli utenti, e della chat vocale, attraverso l'uso delle librerie: **Photon PUN** e **Photon Voice**.

La parte sviluppata da me, invece, riguarda: l'interazione con l'ambiente da parte del docente,

l'implementazione di una chat testuale e l'implementazione di una schermata per l'attivazione e la disattivazione del microfono degli utenti.

### 1.3 Organizzazione dei Capitoli

In questa relazione sono presenti altri 5 Capitoli oltre a questo capitolo introduttivo.

Nel **Cap.2** sarà presentata la fase di analisi e progettazione del prototipo.

Nel **Cap.3** si analizzeranno le Tecnologie abilitanti, ovvero gli strumenti che sono stati utilizzati per la progettazione e sviluppo del prototipo.

Nel **Cap.4** verrà illustrata la parte del prototipo sviluppata dal sottoscritto.

Nel **Cap.5** verrà mostrato il funzionamento delle parti fondamentali del prototipo attraverso alcune dimostrazioni d'uso.

Nel **Cap.6** saranno presentati alcuni possibili sviluppi futuri del prototipo e delle sue funzionalità.

# Capitolo 2

## Analisi e Progettazione

Come anticipato nel capitolo introduttivo, il progetto prevede la creazione di un'aula virtuale in cui il docente e gli studenti possono comunicare tra loro e interagire con l'ambiente circostante.

In questo capitolo verrà illustrata prima la fase di analisi, attraverso la specifica dei requisiti e dei casi d'uso, successivamente la fase di progettazione, con i diagrammi di ***workflow*** (flussi di lavoro).

### 2.1 Specifica dei Requisiti

L'analisi dei requisiti funzionali, ha fatto emergere alcune delle proprietà che il sistema deve necessariamente avere per soddisfare la richiesta.

La prima proprietà riguarda gli attori del sistema, secondo la specifica, infatti, è necessario distinguere due attori:

- Docente
- Studente

Questa distinzione implica la necessità di introdurre un sistema di autenticazione in modo tale da poter identificare l'utente e il suo ruolo all'interno dell'applicazione.

Una seconda proprietà emersa riguarda il tipo di interazioni che gli attori possono avere tra loro, perciò è stato introdotto un mezzo di comunicazione fra utenti connessi.

Infine, è stato necessario introdurre un sistema che permetesse agli utenti di interagire con l'ambiente circostante, in particolare il docente deve avere la possibilità di indicare un punto di interesse all'interno della scena virtuale.

Questa breve descrizione delle funzionalità richieste ci permette di identificare più in dettaglio i requisiti funzionali:

- Il sistema deve permettere l'identificazione dell'utente, in modo da poterne stabilire il ruolo;
- Il sistema deve consentire agli attori di accedere ad uno spazio comune e di potersi vedere reciprocamente;
- All'interno dello spazio comune il sistema deve fornire uno strumento di comunicazione agli utenti;

- All'interno dello spazio comune il sistema deve prevedere la possibilità, da parte del docente, di indicare un punto di interesse.

Un forte vincolo imposto all'applicazione è rappresentato dall'assenza di un'iscrizione o registrazione. Per semplicità, si è deciso di trascurare questa funzionalità aggiuntiva che necessiterebbe di un Database, per memorizzare i dati di accesso degli utenti, e di una funzionalità per l'accesso ai dati. Le credenziali di autenticazione sono specificate in un file json.

## 2.2 Analisi dei casi d'uso

La specifica dei requisiti indica chiaramente la presenza di due attori che condividono alcune funzionalità comuni, pertanto è stata creata una generalizzazione, chiamata Utente, di questi due attori.

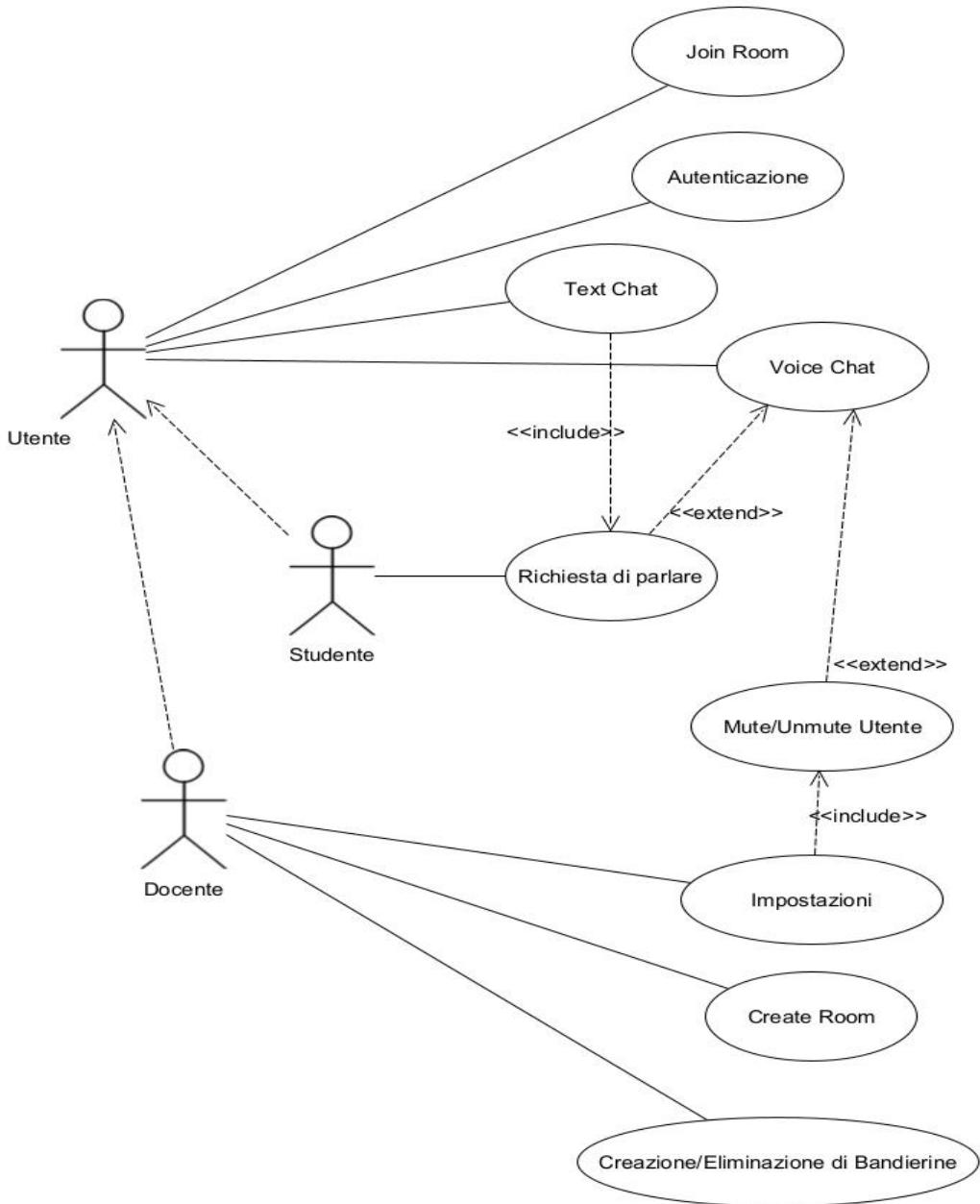


Figura 2.1: Diagramma completo dei casi d'uso

Dalle specifiche dei requisiti sono stati identificati 9 casi d'uso, ovvero la descrizione di un insieme di interazioni, tra un utente ed un sistema, che consentono all'utente di raggiungere un obiettivo o di svolgere un compito:

- Autenticazione;
- Create Room;
- Join Room;

- Chat Vocale;
- Chat di testo;
- Richiesta di parlare;
- Impostazioni;
- Mute/Unmute Utente;
- Creazione/Eliminazione di bandierine.

### 2.2.1 Autenticazione

Dalle specifiche risulta evidente la necessità di distinguere gli utenti in due attori, perciò è stato introdotto un meccanismo di autenticazione necessario per la verifica dell'identità dell'utente.

Il caso d'uso *Autenticazione* inizia in seguito all'avvio dell'applicazione, dopodiché l'attore inserisce le proprie credenziali, ovvero *username* e *password* e clicca sul bottone *login*.

Se le credenziali sono errate, il sistema mostrerà un messaggio di errore, se invece l'autenticazione ha successo, l'attore verrà indirizzato verso una nuova scena che differisce a seconda del ruolo in cui si è identificato.

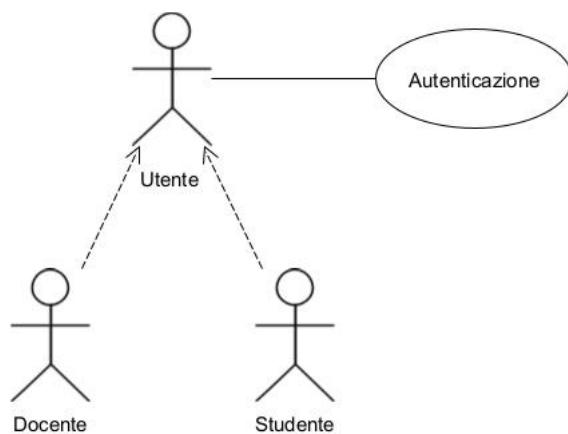


Figura 2.2: Diagramma del caso d'uso Autenticazione

### 2.2.2 Create Room

Il caso d'uso *Create Room* ha come precondizione aver effettuato l'autenticazione con successo con il ruolo di docente.

Il sistema mostra all'attore una schermata con la lista delle aule già create, se presenti, e offre la possibilità di creare una nuova aula, assegnandole un nome.

In seguito alla creazione della stanza, il docente viene automaticamente indirizzato alla scena relativa all'aula appena creata.

Pertanto, sia che il docente crei una nuova stanza oppure scelga di entrare in un'aula già presente, il sistema caricherà la scena corretta per l'attore.

### 2.2.3 Join Room

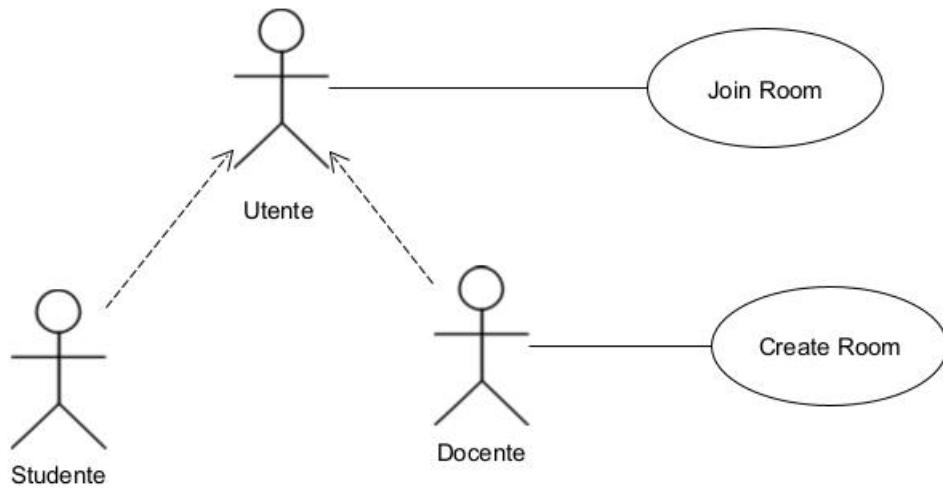


Figura 2.3: Diagramma dei casi d'uso Join Room e Create Room

Il caso d'uso *Join Room* ha come precondizione aver effettuato l'autenticazione con successo con uno dei ruoli resi a disposizione dal sistema.

Il sistema presenta una schermata in cui viene mostrato l'elenco delle aule già disponibili.

L'utente può scegliere tra le aule presenti ed entrare in una di queste, cliccando direttamente sul nome dell'aula.

Nel caso di un docente, l'ingresso in una stanza verrà effettuato non solo attraverso la scelta tra quelle disponibili, ma anche dopo aver effettuato la creazione di una nuova stanza.

### 2.2.4 Chat Vocale

Il caso d'uso *Chat Vocale* ha come precondizione l'accesso ad uno spazio condiviso da due o più utenti.

In questa fase non sono richieste particolari azioni da parte degli attori, ma risulta importante evidenziare come essi ottengano valore da questo caso d'uso, infatti è da questo che gli utenti ottengono la possibilità di comunicare tra loro tramite i microfoni e le cuffie.

Non appena gli attori accedono alla stessa aula, il sistema permette loro di comunicare e di vivere un'esperienza audio realistica attraverso la tecnologia **audio 3D**, che consente all'ascoltatore di percepire il suono da ogni direzione.

### 2.2.5 Chat di testo

Il caso d'uso *Chat di testo* ha come precondizione l'accesso ad uno spazio condiviso da due o più utenti.

Il sistema garantisce una comunicazione network attraverso un'interfaccia utente nella quale è possibile scrivere dei messaggi e inviarli agli altri utenti.

## 2.2.6 Richiesta di parlare

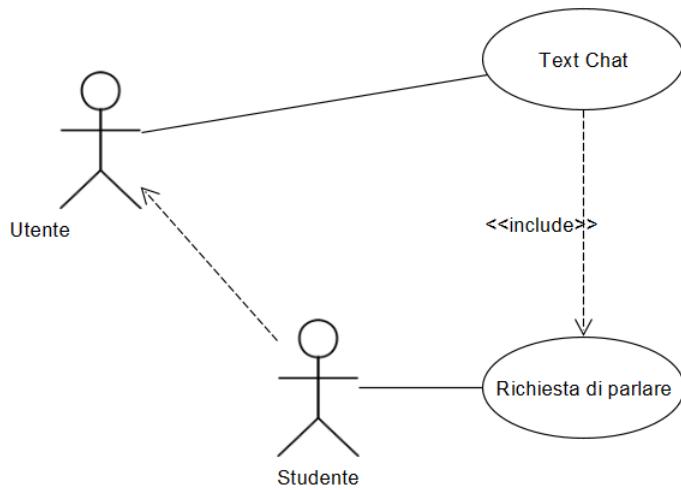


Figura 2.4: Diagramma del caso d’uso Text Chat che include Richiesta di Parlare

Il caso d’uso *Richiesta di Parlare* ha come precondizione l’accesso ad uno spazio condiviso da due o più utenti e che l’attore di questo caso d’uso abbia il ruolo di studente.

Come detto in precedenza, all’interno dell’aula, il docente avrà l’autorità di concedere la parola ad uno studente, a tutti, oppure togliere questo diritto a sua discrezione.

Questo potrebbe portare ad una comunicazione unilaterale in cui gli studenti sono attori passivi della comunicazione.

Per ovviare a questo possibile inconveniente, gli studenti hanno la possibilità di richiedere la parola. Per fare ciò, nella schermata della chat di testo presentata dal sistema, sarà presente un bottone che invierà sulla chat una notifica, della forma: ‘*Nome studente* richiede di parlare’, a tutti gli utenti. A questo punto il docente può decidere se concedere la parola allo studente che l’ha richiesta.

## 2.2.7 Impostazioni

Il caso d’uso *Impostazioni* ha come precondizione l’accesso ad uno spazio condiviso da due o più utenti e che l’attore di questo caso d’uso abbia il ruolo di docente.

Il sistema mostrerà al docente una schermata, che possiede l’elenco dei nomi dei giocatori connessi alla stanza, nella quale sono abilitate le funzioni del caso d’uso *Mute/Unmute Utente*.

## 2.2.8 Mute/Unmute Utente

Il caso d’uso *Mute/Unmute Utente* ha come precondizione l’accesso ad un’aula da parte di più utenti e che l’attore di questo caso d’uso abbia il ruolo di docente.

All’interno dell’aula, il docente avrà l’autorità di concedere la parola ad uno studente, a tutti, oppure togliere questo diritto a sua discrezione.

Il docente avrà la possibilità di vedere l'elenco degli studenti presenti nell'aula e, accanto ai loro nomi, sarà presente un bottone che permetterà di dare e togliere la parola al relativo studente.

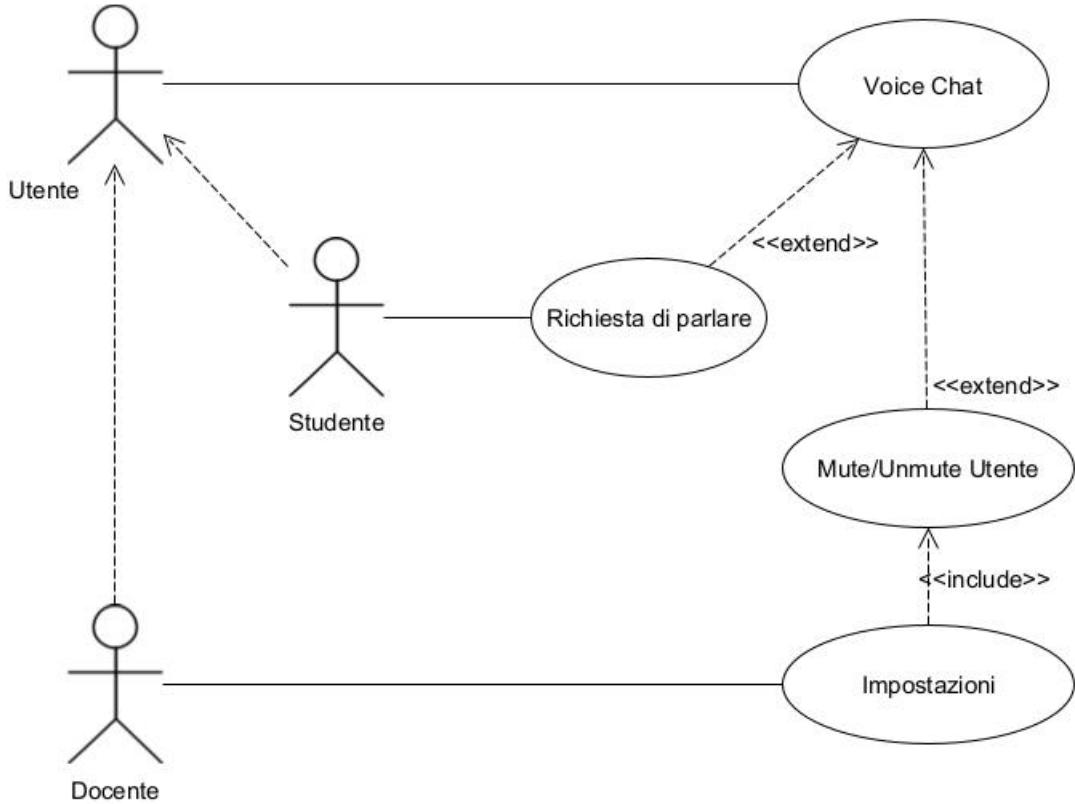


Figura 2.5: Diagramma dei casi d'uso Voice Chat, Mute/Unmute Utente, Impostazioni

### 2.2.9 Creazione/Eliminazione di bandierine

Il caso d'uso *Creazione/Eliminazione di bandierine* ha come precondizione l'accesso ad un'aula da parte di più utenti e che l'attore di questo caso d'uso abbia il ruolo di docente.

Il caso d'uso modella l'interazione con l'ambiente circostante, ovvero con gli elementi presenti nell'aula. Questa interazione si è tradotta nella possibilità di inserire, in alcuni punti prestabiliti, una bandierina. L'attore abilitato a questo caso d'uso deve indicare in direzione dell'area prestabilita e, cliccando all'interno di quell'area, può generare la bandierina.

Ad un successivo click, all'interno dell'area, la bandierina viene rimossa.

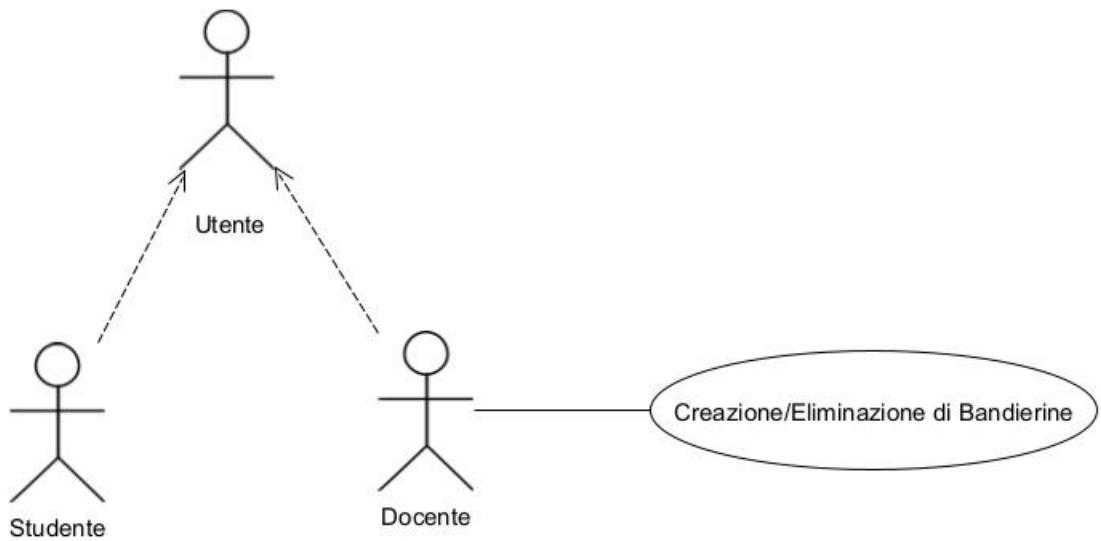


Figura 2.6: Diagramma del caso d'uso Creazione/Eliminazione Bandierina

## 2.3 Diagrammi Workflow

La definizione dei casi d'uso porta a scomporre il progetto nelle sue parti costituenti.

Ognuna di queste risponde ad una specifica necessità emersa dall'analisi dei requisiti.

Molti casi d'uso si sono tradotti quasi direttamente in un diagramma *workflow*, che mostra il flusso delle diverse attività in un caso d'uso.

In particolare sono stati definiti 3 diagrammi *workflow* che corrispondono a 3 casi d'uso descritti in precedenza:

- Autenticazione Workflow;
- Create Room Workflow;
- Join Room Workflow.

L'analisi di questi 3 casi d'uso, più il caso d'uso *Voice Chat*, verrà poi svolta più dettagliatamente dai colleghi Emanuele Sapiro e Lorenzo Iacopetta nelle loro Relazioni, visto che riguarda la parte di cui si sono occupati nel progetto.

I rimanenti casi d'uso verranno analizzati dettagliatamente in un capitolo a parte, il **Cap.4**, per evidenziare meglio la parte del progetto svolta dal sottoscritto.

### 2.3.1 Autenticazione

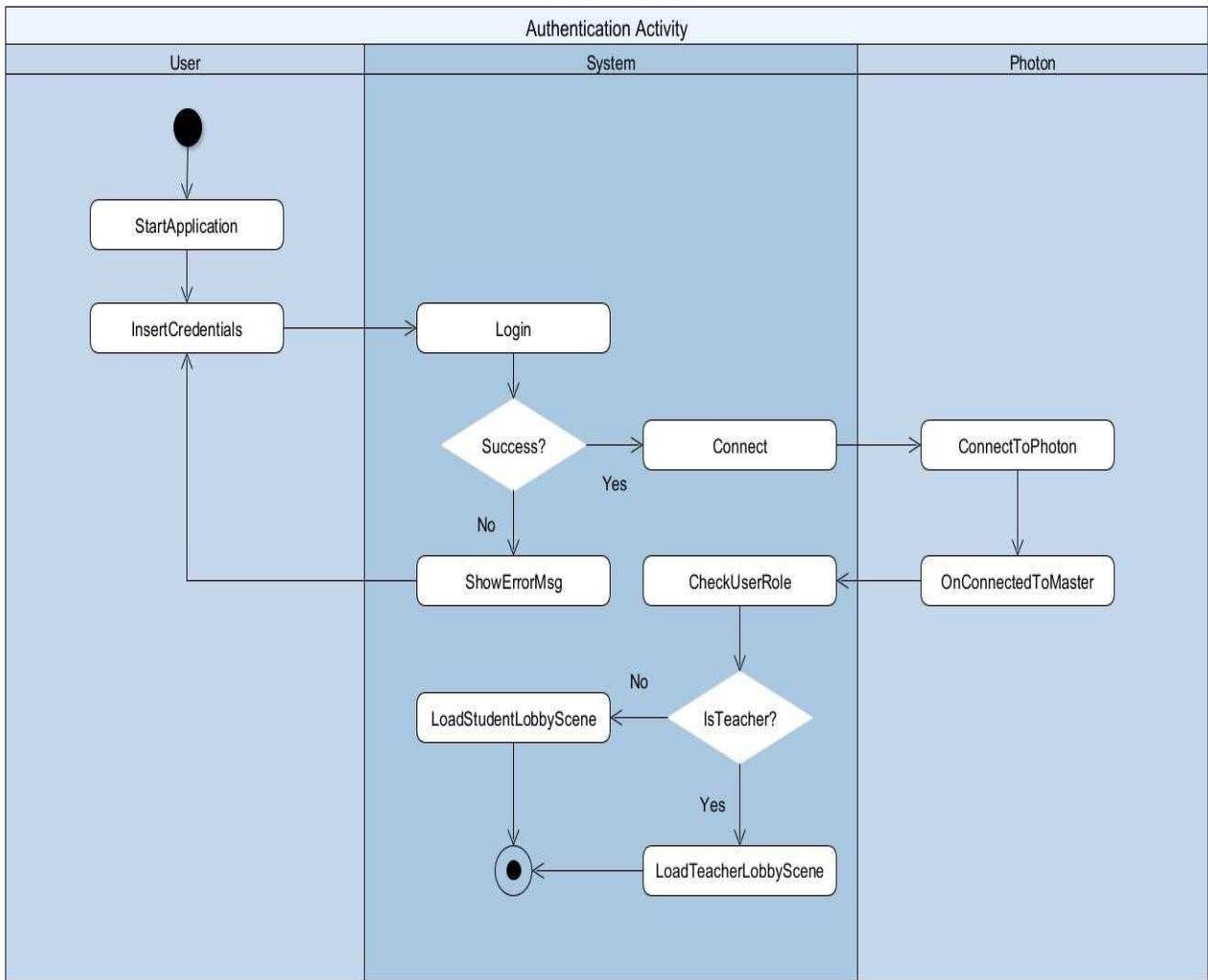


Figura 2.7: Diagramma Workflow di Autenticazione

L'autenticazione degli utenti ricopre un ruolo importante all'interno dell'applicazione, perché, in base al ruolo dell'attore, il sistema caricherà scene diverse e si avranno diritti e funzioni diverse.

Quando l'applicazione viene avviata, il sistema mostrerà all'utente una schermata in cui è possibile inserire le proprie credenziali ed effettuare l'accesso.

In seguito al click dell'attore sul pulsante *Login*, il sistema deve verificare che le credenziali siano corrette.

Se l'utente non esiste nella base dati, il sistema deve restituire un messaggio di errore all'utente che può ripetere l'operazione, altrimenti il sistema si connette ad un server esterno (*server Photon*).

Il server Photon possiede un identificativo, nascosto all'utente, che deve essere necessariamente impostato correttamente dal programmatore per poter permettere la comunicazione tra applicazioni diverse (e quindi utenti diversi), perciò le applicazioni con lo stesso identificativo si conserveranno allo stesso server di Photon.

In seguito alla connessione al server, il sistema verifica il ruolo dell'utente che si è appena collegato. Se si tratta di un docente, l'applicazione caricherà la scena dedicata ai docenti che permetterà, oltre ad accedere ad aule già presenti, la creazione di una nuova aula.

Altrimenti, se l'attore è uno studente, il sistema caricherà la scena dedicata agli studenti, in cui è possibile scegliere ed entrare in un'aula.

Il flusso termina quando il sistema carica la scena appropriata per l'utente che si è collegato.

### 2.3.2 Create Room

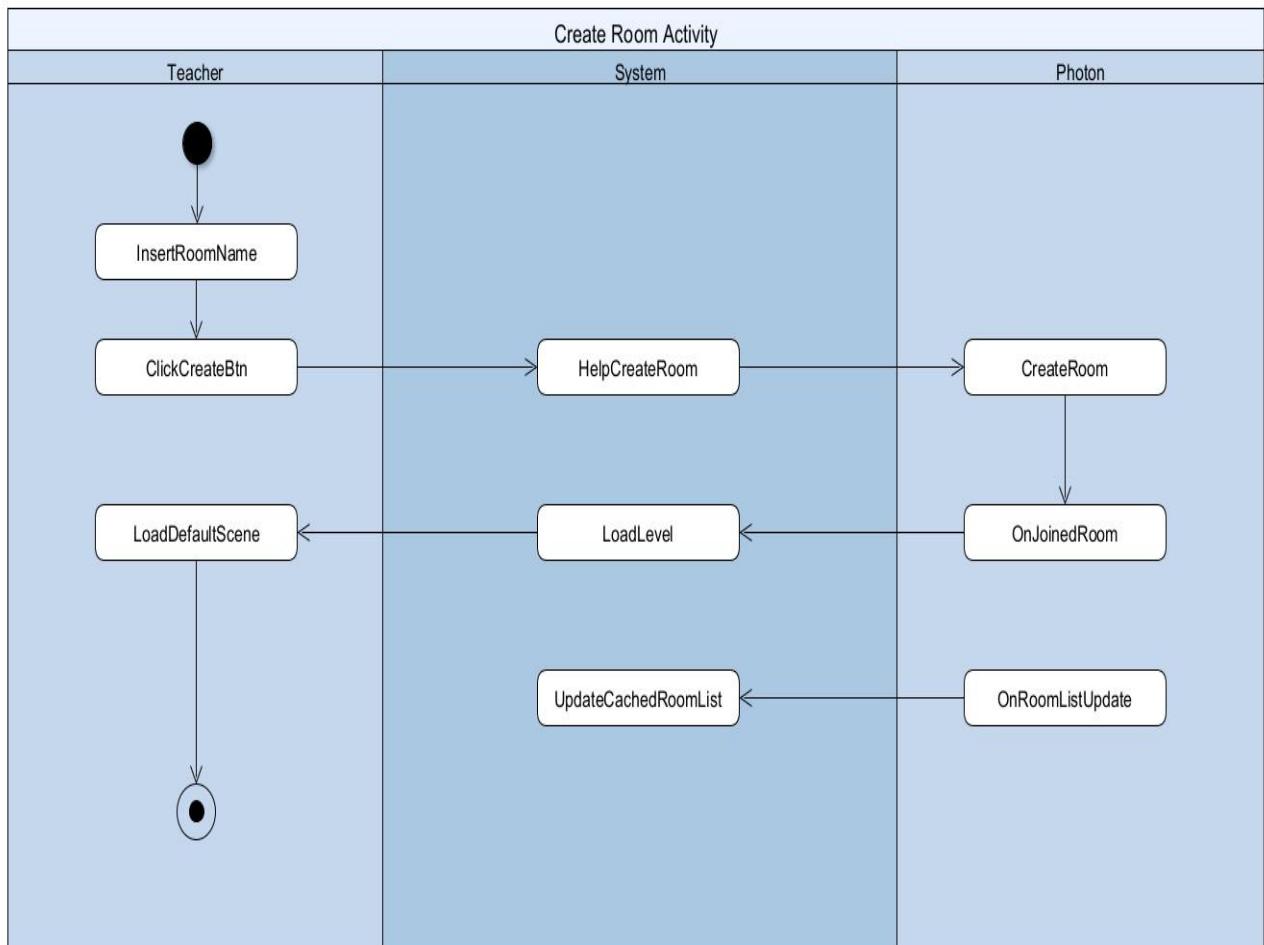


Figura 2.8: Diagramma Workflow di Create Room

In questo diagramma sono mostrate le attività che vengono svolte durante la creazione della stanza da parte del docente.

Il docente, innanzitutto, deve inserire un nome che verrà assegnato all'aula e cliccare sul bottone *Create Room*.

Il sistema, a questo punto, invierà la richiesta di creazione di una stanza al server Photon che creerà un'istanza sul proprio server etichettata con il nome che il docente ha assegnato all'aula.

Dopodiché, il server Photon aziona una funzione di ingresso, per il docente, alla stanza appena creata, infine il sistema, sull'applicazione del docente, caricherà la scena in cui poi gli utenti andranno ad interagire.

La stanza creata dal docente sarà poi visibile a tutti gli altri utenti che accederanno alla scena adibita all'ingresso delle stanze virtuali, grazie all'operazione *OnRoomListUpdate* del server Photon.

### 2.3.3 Join Room

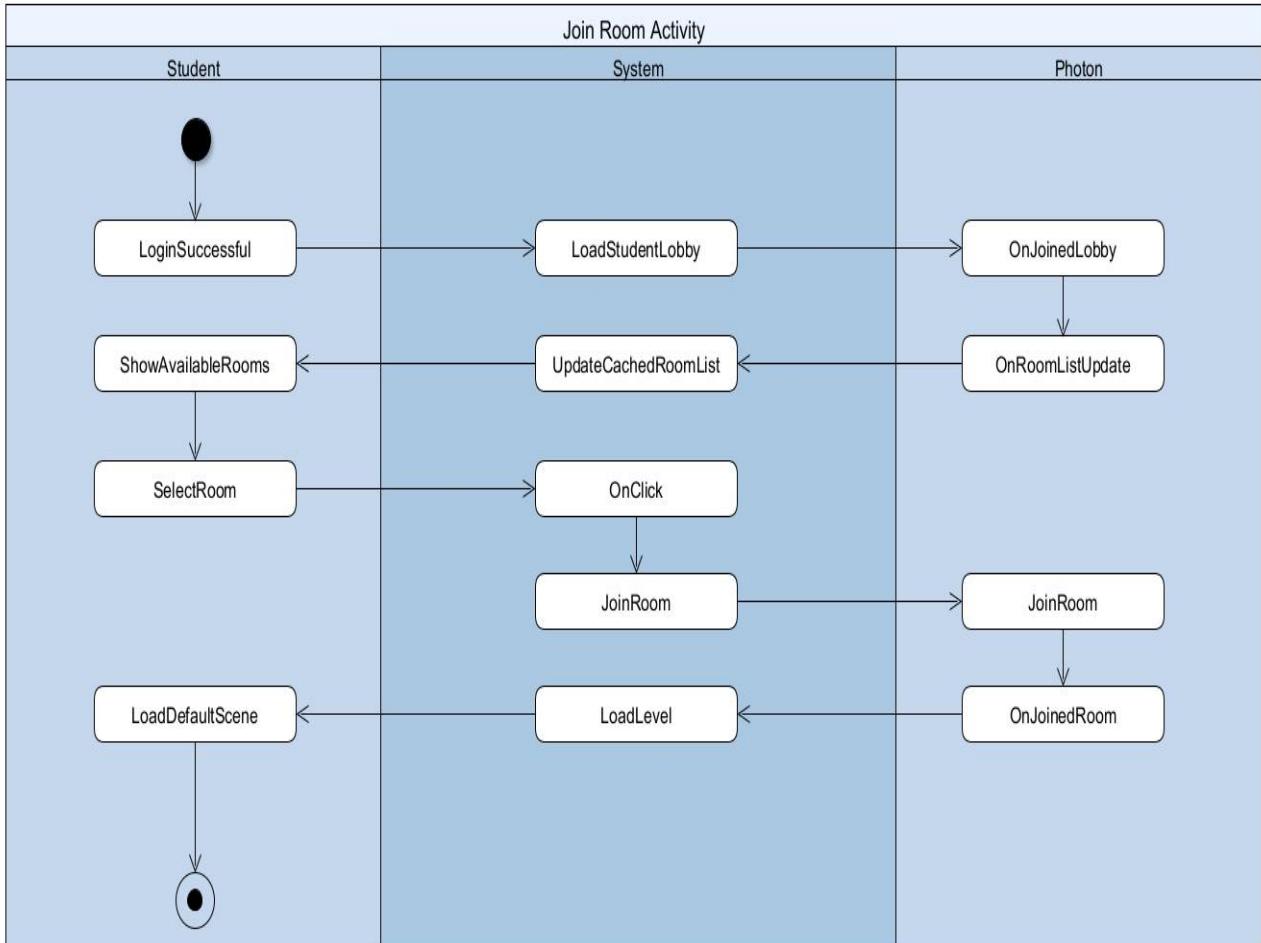


Figura 2.9: Diagramma Workflow di Join Room

In questo diagramma sono illustrate le attività svolte durante l'operazione di collegamento ad una stanza creata.

Come già accennato in precedenza, gli studenti visualizzeranno una scena in cui compare solo l'elenco delle aule disponibili, mentre gli insegnanti avranno anche la possibilità di creare una nuova aula.

Come analizzato nel diagramma *workflow* di Create Room, si tratta di una distinzione importante delle attività dell'applicazione rispetto a quelle del server Photon.

L'accesso al server Photon, infatti, è del tutto trasparente all'utente perché è gestito dal server esterno.

Il compito del sistema, in questo caso, sarà quello di caricare e mostrare all'utente la lista delle stanze disponibili a cui si può collegare.

Una volta caricate le aule disponibili, l'utente può sceglierne una, e al click, l'applicazione invierà la richiesta di ingresso all'aula al server Photon.

Sarà poi il sistema, in seguito all'attività del server, a caricare la scena relativa all'aula selezionata. Quindi, ogni applicazione possiede una o più scene da caricare, quello che permette la comunicazione tra utenti è la connessione allo stesso server Photon.

Gli utenti potranno vedersi e interagire tra loro nella scena finale grazie alla presenza di un *Avatar*: ogni utente possiede un *Avatar* che è stato progettato con delle componenti *software* in grado di renderlo percepibile a tutti gli altri utenti collegati alla stessa stanza virtuale.

# Capitolo 3

## Tecnologie Abilitanti

In questo capitolo verranno descritti gli strumenti e le tecnologie che sono stati utilizzati per la realizzazione dei requisiti software illustrati nel **Cap.2**.

### 3.1 Realtà Virtuale

La Realtà Virtuale[1] è un ambiente esclusivamente digitale, creato da uno o più computer, che simula la realtà effettiva e la ricrea in modo non tangibile, che viene visualizzato dall'utente tramite l'utilizzo di strumenti particolari, detti *VR Headset*.

L'obiettivo principale di questa tecnologia è quello di ricreare un ambiente che stimoli il più possibile i 5 sensi, in modo da rendere il tutto molto più realistico.

L'architettura necessaria per poter usufruire in maniera completa della Realtà Virtuale è composta da:

- Visori che abbiano determinate caratteristiche quali, ad esempio, un campo visivo dai 100 ai 110 gradi, un *frame rate* (frequenza di immagini proiettate al secondo) compreso tra un minimo di almeno 60fps ed un massimo di 120fps per evitare una visione a scatti fastidiosa agli occhi
  - Un giroscopio che consenta, insieme ad un accelerometro ed ad un magnetometro, il cosiddetto *Head Tracking*, ovvero lo spostamento dell'immagine seguendo esattamente i movimenti del capo lungo i quattro punti cardinali e con tempi di risposta dai cinquanta millisecondi ai trenta millisecondi.
- Tutto questo è sviluppato volutamente per far sì che l'utente possa interagire e “vivere” all'interno della Realtà Virtuale.
- La presenza all'interno del visore di un sistema audio professionale multicanale che offre la sensazione di suoni che provengono da tutte le direzioni e che consentano il cosiddetto effetto doppler (con l'aumentare del suono in avvicinamento ed il diminuire in allontanamento), sia da un sofisticato sistema di puntamento ad infrarossi che consente di leggere il movimento oculare (il cosiddetto *Eye Tracking*) rendendo ancora più realistica l'immersione nell'ambiente virtuale mediante la creazione della profondità di campo.

Nell'introduzione è stata illustrata l'utilità della Realtà Virtuale in ambito didattico, ma può essere impiegata in molteplici campi, come ad esempio nella medicina.

Nel campo medico la Realtà Virtuale sta diventando uno strumento non soltanto formativo, ma anche terapeutico ed operativo, con l'epilogo più famoso ad aprile 2016 al Royal London Hospital dove si è svolta in collegamento con l'India la prima operazione chirurgica della storia in realtà virtuale trasmessa in tempo reale.

### 3.1.1 Headset



Figura 3.1: Oculus Quest 2 con i relativi controller

Un *Virtual Reality Headset*[2], o *VR Headset*, è un dispositivo elettronico, detto visore, che permette la visione di applicazioni sviluppate per la Realtà Virtuale.

Esistono due categorie di *VR Headset*, quelli che necessitano di un collegamento ad un dispositivo Desktop e i dispositivi portatili con hardware dedicato.

I primi sono i più indicati per lo sviluppo di ambienti in Realtà Virtuale sofisticati, visto che un PC è dotato di una potenza di calcolo mediamente superiore a quella di un dispositivo con hardware dedicato, ma di fatto ciò che l'utente vede è un semplice Render in tempo reale del mondo virtuale.

I *VR Headset* portatili con hardware integrato, o *standalone* non offrono una resa grafica alla pari dei visori descritti precedentemente, ma permettono comunque di immergere l'utente in un'ambientazione verosimile, garantendo il vantaggio della portabilità.

Il prototipo sviluppato è stato pensato per essere utilizzabile sul modello di visore rappresentato nella **Figura 3.1**.

L'Oculus Quest 2 è un modello di visore *standalone*, con esso è possibile non solo giocare ai giochi appositamente pensati per la Realtà Virtuale, ma anche guardare film come se si fosse al cinema, crearsi la propria configurazione multi-monitor per lavorare, visualizzare foto e video a 360 gradi, e fare tutto quello che ci viene in mente sempre immersi in un mondo virtuale.

Il visore si può configurare attraverso l'app *Oculus*, scaricabile nel proprio *smartphone*, e, una volta

indossato, ci ritroveremo subito all'interno della *home*, dalla quale potremo avviare tutte le *app* e i giochi in nostro possesso.

Nel mondo virtuale si può interagire con i *controller*. *Oculus Touch*, le funzionalità pensate per gli utenti in questo prototipo si basano, perciò, sulle specifiche tecniche di questi *controller*.

## 3.2 Unity



Figura 3.2: Logo di Unity

**Unity**[3] è un motore grafico per lo sviluppo di applicazioni 3D e 2D, anche in Realtà Virtuale, nato nel 2005.

La progettazione di applicazioni è basata sul linguaggio di programmazione C#(*C sharp*), i cui *Script* vengono gestiti tramite il software Microsoft Visual Studio.

L'Editor di *Unity* è supportato da Windows, MacOS e Linux, inoltre le applicazioni che si possono sviluppare sono supportate da piattaforme di vario tipo per esempio:

- Mobili: IOS, Android;
- Desktop: Windows, MacOS, Linux;
- Web: WEBGL;
- Console: XBOX, Playstation(PS4, PS5), Nintendo Switch;
- Virtual Reality/Extended Reality: Oculus, PlayStation VR, Google's ARCore, Apple's ARKit, Windows Mixed Reality, Google Cardboard.

### 3.2.1 Interfaccia Utente

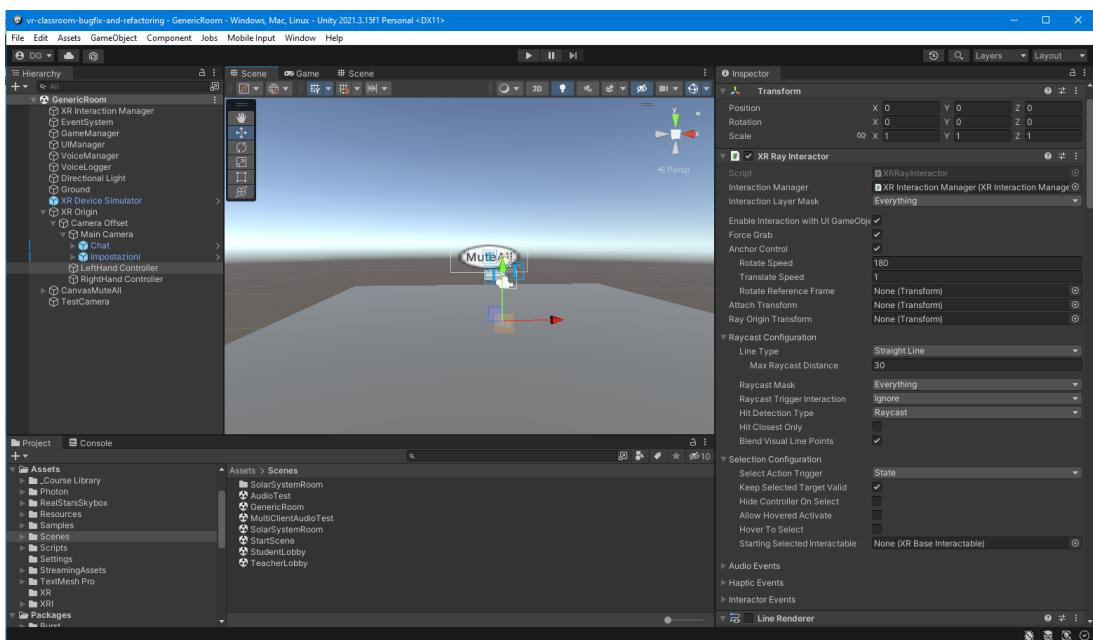


Figura 3.3: Interfaccia utente di Unity 2021.3.15f1

L’interfaccia utente[4] di *Unity*, detta anche Unity Editor, è composta da 5 sezioni principali: Hierarchy, Game and Scene, Inspector, Project e Console.

### 3.2.1.1 Hierarchy

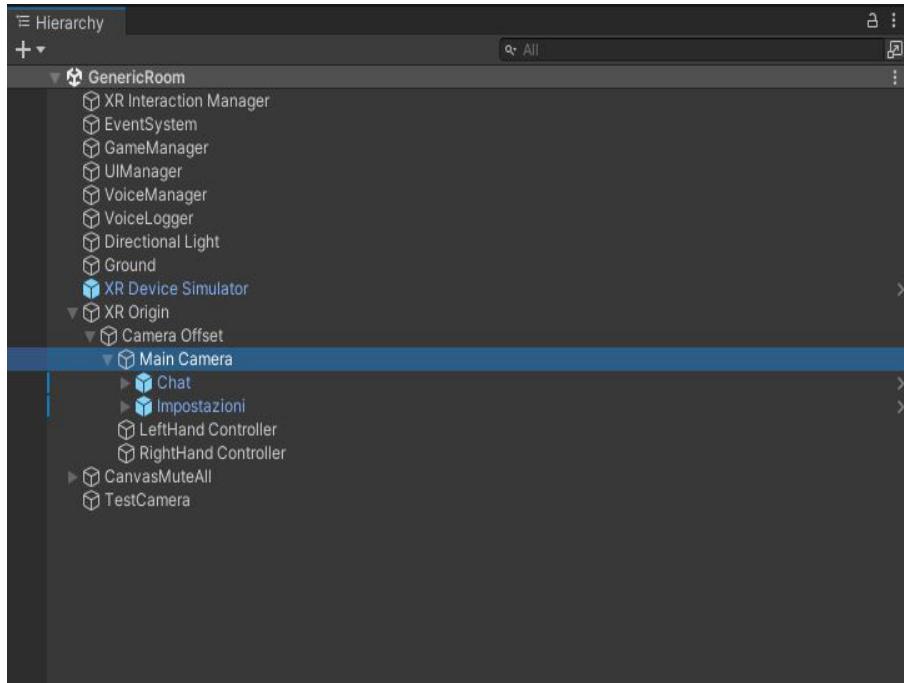


Figura 3.4: Hierarchy Panel di Unity Editor

La Hierarchy[5] è la sezione dell'Editor in cui sono elencati tutti gli oggetti della scena e la relazione che c'è tra essi.

La relazione tra oggetti è denominata gerarchia, o parentela, ovvero un oggetto può essere un 'genitore' o 'figlio' di un altro oggetto.

È lo stesso tipo di relazione si può trovare tra cartelle nel File System di un sistema operativo.

L'oggetto principale, il genitore, trasmette le proprie componenti e proprietà agli oggetti figli, quindi ogni modifica effettuata sull'oggetto genitore verrà applicata anche all'oggetto figlio.

La relazione di parentela può anche essere presente su più livelli, ossia un oggetto figlio può avere un oggetto figlio e così via.

### 3.2.1.2 Game and Scene

La Scena[6] è la sezione dell'Editor che rappresenta l'ambiente applicativo vero e proprio, in essa, infatti, sono visibili gli oggetti, detti *GameObject*, che sono stati creati dal programmatore.

Nella Scena, inoltre, si ha una visione interattiva dell'ambiente applicativo, ovvero è possibile posizionare, selezionare e manipolare gli oggetti.

Gli oggetti di scena possono essere di vario tipo per esempio: modelli 3D come cubi e sfere, modelli dotati di *texture*, *controller* per il giocatore, luci per l'ambientazione e una telecamera per rendere visibili tutti gli oggetti della scena, oltre alla scena stessa, nell'applicazione finale.

Dalla Scena si può entrare nella Game Mode[7], quella sezione dell'Editor in cui è possibile testare ciò che si è creato nella Scena, come ad esempio i movimenti del giocatore e dei vari oggetti implementati nella Hierarchy.

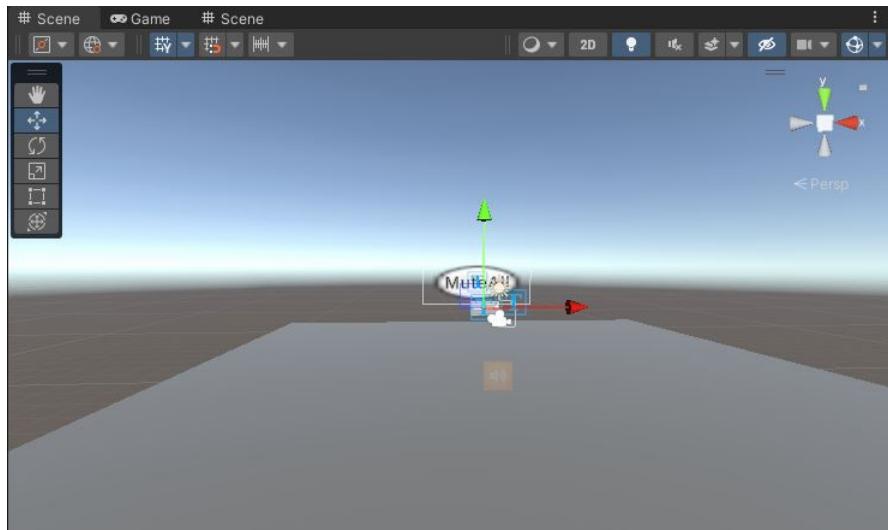


Figura 3.5: Game and Scene Panel di Unity Editor

### 3.2.1.3 Inspector

L'Inspector[8] è la sezione dell'Editor dedicata alla gestione delle componenti degli oggetti *Unity*, che possono essere di vario tipo, per esempio: script C#, Transform, Text, Audio, Video ecc... Sono le componenti dell'oggetto che determinano le funzionalità che ha all'interno della scena, per cui è essenziale applicare le componenti giuste agli oggetti per potergli permettere di compiere le azioni desiderate.

Ogni componente possiede dei parametri, che variano a seconda del tipo di componente, che posso

essere modificati con mouse e tastiera tramite l’interfaccia utente.

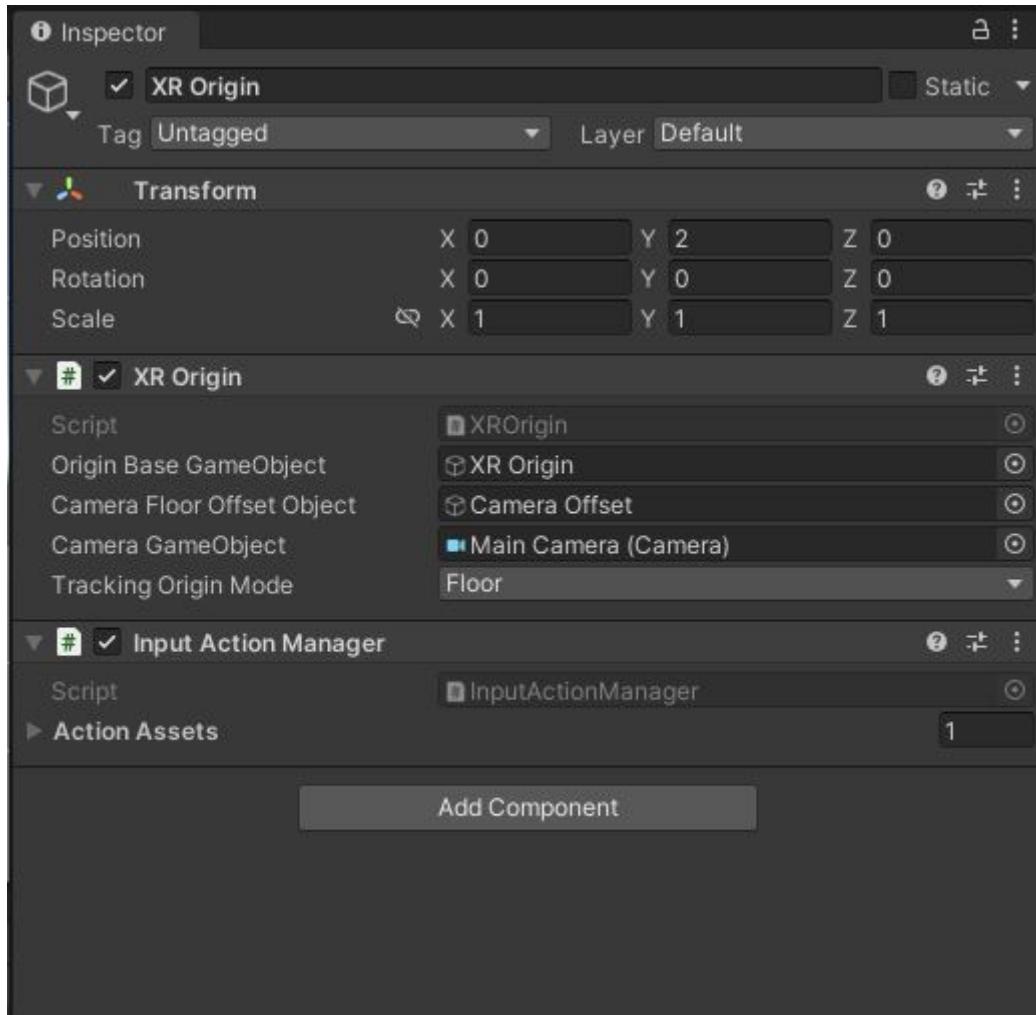


Figura 3.6: Inspector Panel di Unity Editor

È questo uno dei vantaggi che si ha nello sviluppo di applicazioni con *Unity*, ovvero avere un’interfaccia interattiva nella quale si possono effettuare modifiche su oggetti e componenti che hanno effetto immediato sulla Scena.

#### 3.2.1.4 Project

Project[9] è la sezione dell’Editor che mostra la cartella ‘Assets’ del progetto creato, che contiene tutti gli elementi che possono essere utilizzati all’interno del progetto.

Oltre agli elementi che si possono creare con *Unity*, è possibile creare e importare elementi dell’Assets dall’esterno, come per esempio modelli 3D, file Audio, file Video, immagini o qualsiasi altro file compatibile con *Unity*.

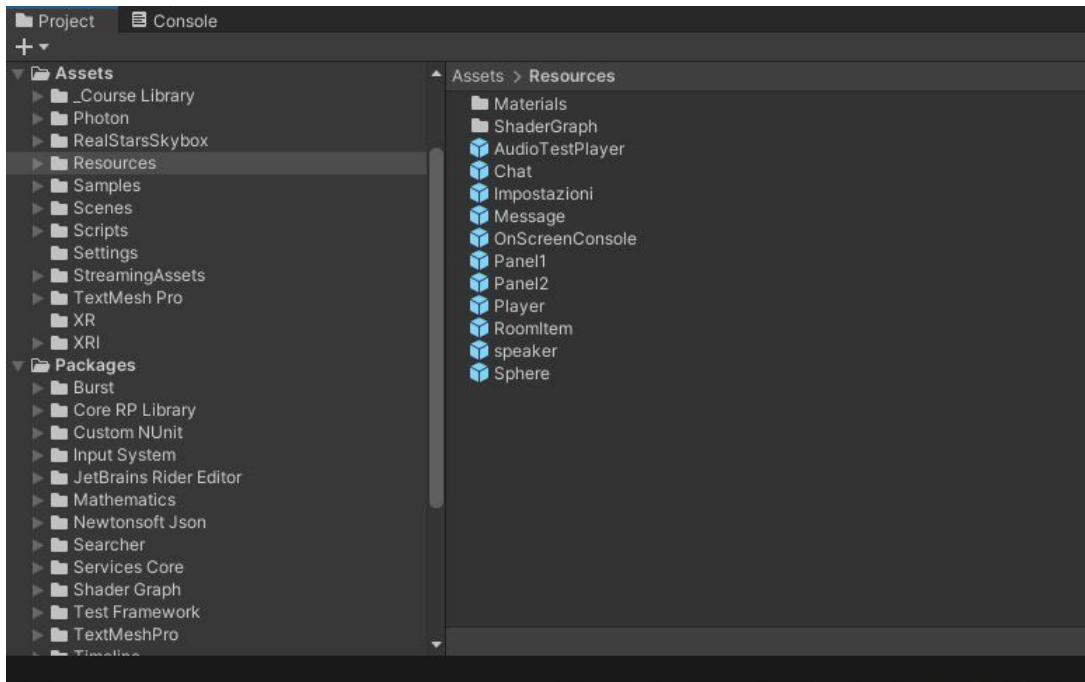


Figura 3.7: Project Panel di Unity Editor

### 3.2.1.5 Console

La Console[10] è la sezione dell'Editor in cui si possono trovare i messaggi di errore, con il simbolo rosso, messaggi di avvertimento (*Warning*) e altri messaggi generati da *Unity*.

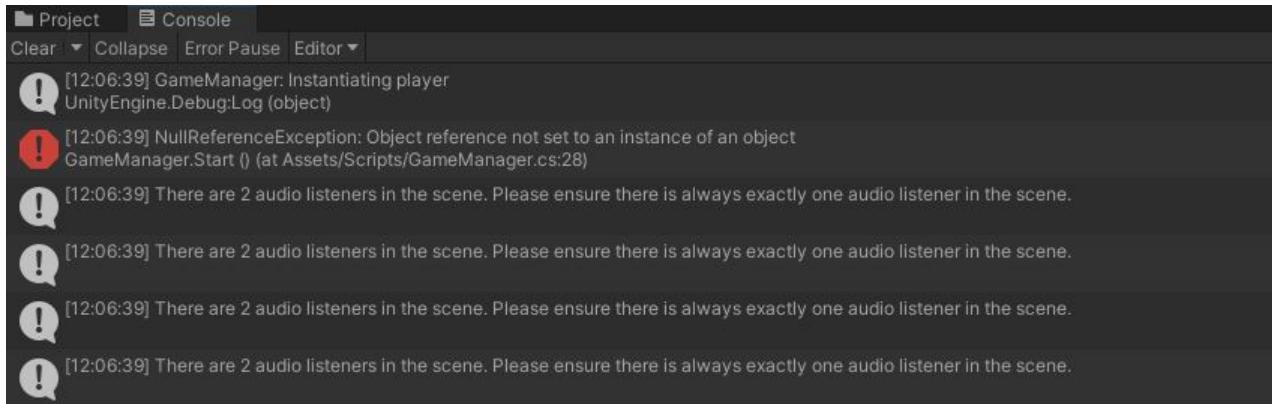


Figura 3.8: Console Panel di Unity Editor

Si possono anche generare manualmente dei messaggi sulla Console utilizzando i comandi *Debug.Log*, *Debug.LogWarning*, *Debug.LogError*.

### 3.3 C#

Il C#[11] è un linguaggio di programmazione multi-paradigma, sviluppato da Microsoft, che supporta tutti i concetti della programmazione orientata agli oggetti, quindi encapsulamento, ereditarietà e polimorfismo.

La sintassi e la struttura del C# prendono spunto da vari linguaggi nati precedentemente, in particolare Delphi, C++, Java e Visual Basic.

#### 3.3.1 .NET Framework

I programmi C# vengono eseguiti su *.NET Framework*[11], un componente di Windows composto da:

- *Common Language Runtime* (CLR);
- Librerie di classi di *.NET Framework*.

Il *Common Language Runtime* rappresenta la base di *.NET Framework* e può essere considerato come un agente che gestisce il codice in fase di esecuzione (simile alla *Java Virtual Machine*), fornendo servizi di base quali gestione della memoria (*Garbage Collector*), gestione di thread, servizi remoti, networking e attivando una rigida indipendenza dai tipi e altre forme di accuratezza del codice che garantiscono sicurezza ed efficienza.

Il codice destinato al *runtime* è definito codice gestito, mentre quello non destinato al *runtime* è definito codice non gestito, per esempio una libreria dinamica (DLL) scritta in C/C++.

Il CLR è responsabile dell'acquisizione del solo codice gestito, della compilazione dello stesso in codice macchina e quindi della sua esecuzione.

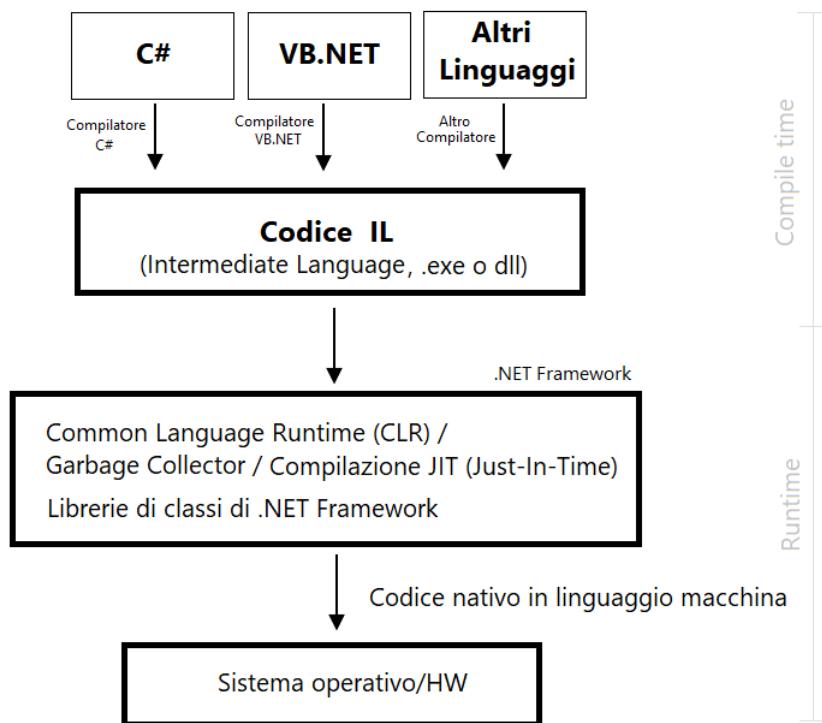


Figura 3.9: Architettura della piattaforma .NET Framework

Nella **Figura 3.9** sono illustrate le relazioni, in fase di compilazione (*Compile Time*) e di esecuzione (*Runtime*), tra i file di codice sorgente C#, o degli altri linguaggi, le librerie di classi di *.NET Framework*, gli *assembly* e il CLR.

Il codice sorgente scritto in C# viene compilato in un linguaggio intermedio (*Intermediate Language*, IL).

Il codice IL viene archiviato sul disco in un file eseguibile denominato *assembly*, in genere con estensione EXE o DLL.

Quando viene eseguito il programma C#, l'*assembly* viene caricato nel CLR, e se i requisiti di sicurezza sono soddisfatti, il CLR esegue una compilazione *Just-In-Time* (JIT) per convertire il codice IL in istruzioni del linguaggio macchina nativo.

Le librerie di classi sono una raccolta completa e orientata agli oggetti di tipi riusabili, da utilizzare per lo sviluppo di applicazioni, da quelle tradizionali a riga di comando o con interfaccia utente grafica (GUI, *Graphical User Interface*) a quelle basate sulle più recenti innovazioni offerte da ASP.NET, quali Web Form e servizi Web XML, Web API e tanto altro.

---

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
...

```

---

Algoritmi 3.1: Esempio di utilizzo di librerie in C#

### 3.3.2 MonoBehaviour

*MonoBehaviour*[12] è la classe di base che tutti gli script in *Unity* devono avere.

La classe *MonoBehaviour* espone una serie di funzioni evento che vengono chiamate da *Unity* al verificarsi di determinate condizioni.

Ad esempio, un evento come *MonoBehaviour.OnCollisionEnter()* si verificherà ogni volta che un oggetto entra in collisione con un altro oggetto.

Quando questo accade, *Unity* provvederà a richiamare automaticamente il metodo *OnCollisionEnter()*(la lista degli eventi è consultabile qui[13]).

Le funzioni evento vanno richiamate all'interno di metodi specifici della classe *MonoBehaviour*, i 3 principali sono:

- *Awake()*;
- *Start()*;
- *Update()*.

I Metodi *Awake* e *Start* vengono chiamati una sola volta nel ciclo di vita di un oggetto.

La differenza fra i due sta nel fatto che *Awake* viene chiamato molto presto, durante la preparazione della scena.

In quel momento gli oggetti in scena non sono completamente valorizzati, quindi in *Awake* è sconsigliato svolgere operazioni che richiedono i valori di altri oggetti, come ad esempio leggere il *tag* di un altro oggetto in scena.

*Start* è utile per eseguire operazioni di preparazione al gioco, come la ricerca di elementi nella scena di cui vogliamo salvare un riferimento per poi lavorarci più avanti durante l'esecuzione.

Molto interessante è il tempismo di chiamata dei due metodi a seconda dello stato, attivo o disattivo, dello script in cui compaiono *Awake* e *Start* e il *GameObject* su cui risiede lo script, che viene illustrato nella seguente immagine:

Stato GameObject	Stato script	Conseguenza
attivo	abilitato	viene chiamato prima <i>Awake</i> , poi <i>Start</i>
attivo	disabilitato	viene chiamato solo <i>Awake</i> . Se successivamente l'oggetto viene attivato, verrebbe chiamato anche <i>Start</i> (ma non di nuovo <i>Awake</i> )
disattivo	abilitato o disabilitato	né <i>Awake</i> né <i>Start</i> vengono chiamati. Se il <i>gameObject</i> viene attivato, passiamo nelle condizioni precedenti.

Figura 3.10: Tempismo di chiamata dei metodi *Awake* e *Start*

Il metodo *Update*, invece, viene chiamato ad ogni frame, il che lo rende molto utile, per esempio, per eseguire azioni come aggiornare o modificare la posizione di oggetti nella scena, per tenere traccia dell'input del giocatore e attribuire comportamenti periodici agli oggetti. ,

---

```
using UnityEngine;
using System.Collections;

public class GameObject: MonoBehaviour {

void Start(){ . . . }

void Update(){ . . . }
}
```

---

Algoritmi 3.2: Esempio di utilizzo dei metodi Start e Update

### 3.3.3 Librerie Utilizzate

L'utilizzo di librerie è essenziale per la progettazione e sviluppo di software, sono estremamente utili perché forniscono una collezione di entità di base pronte per l'uso, evitando al programmatore di dover riscrivere ogni volta le stesse funzioni o strutture dati e facilitando così le operazioni di sviluppo e manutenzione.

Per la realizzazione di questo prototipo si distinguono in due tipologie di librerie:

- Le librerie interne presenti nello *Unity Registry*;
- Le librerie *Photon* importate dall'*Asset Store*.

#### 3.3.3.1 Librerie Interne

Le principali librerie Interne utilizzate sono:

- *UnityEngine*, raccoglie le classi fondamentali e di riferimento per ogni progetto *Unity*:
  - *UnityEngine.Events*[14];
  - *UnityEngine.InputSystem*[15];
  - *UnityEngine.XR.Interaction.Toolkit*[16];
  - *UnityEngine.UI*[17].
- *System*, contiene le classi fondamentali e di base che definiscono eventi, gestori di eventi, interfacce, attributi, eccezioni di elaborazione e tipi di dati di riferimento e valore usati comunemente:
  - *System.Collections*[18];
  - *System.Collections.Generic*[19].
- *TMP*[20], offre soluzioni per il testo relativo all'interfaccia utente dell'applicazione finale.

### 3.3.3.2 Librerie Photon

Le librerie *Photon* importate sono:

- Photon.PUN[21];
- Photon.Chat[22];
- Photon.Voice[23].

## 3.4 Photon Engine



Figura 3.11: Struttura di Photon Engine

*Photon Engine*[24] è un motore grafico specializzato nello sviluppo di applicazioni multi-utente. Nella **Figura 3.11** si può vedere che il motore grafico è composto da due parti principali:

- *Photon Cloud*;
- *Photon Realtime*.

Il *Photon Cloud* è un soluzione **SaaS** (*Software As A Service*), questo significa che il *Runtime System* locale sul dispositivo client ha già riferimenti al sistema server, che è rilasciato e gestito in un ambiente cloud esterno.

Il sistema, perciò, non è completamente sotto controllo dell'utente, ma ha dipendenze da componenti remoti di *Photon*.

Questo è il motivo principale per cui i servizi offerti da *Photon* sono stati scelti per sviluppare le funzionalità di rete.

Il *Photon Realtime* è la parte del motore grafico composta dai *packages* e dalle librerie, quindi sono gli strumenti che l'utente deve utilizzare per sviluppare la propria applicazione multi-utente. In particolare per lo sviluppo del prototipo sono state utilizzate 3 applicazioni del *Photon Realtime*:

- PUN (*Photon Unity Networking*), utilizzata per la connessione tra utenti;
- *Photon Voice*, utilizzata per la comunicazione vocale tra utenti;
- *Photon Chat*, utilizzata per la comunicazione via chat tra utenti.

### 3.4.1 Implementazione di PUN in Unity

In questa sezione saranno illustrati i passaggi fondamentali per la preparazione di un progetto *Unity* alle funzionalità multi-utente di *Photon*.

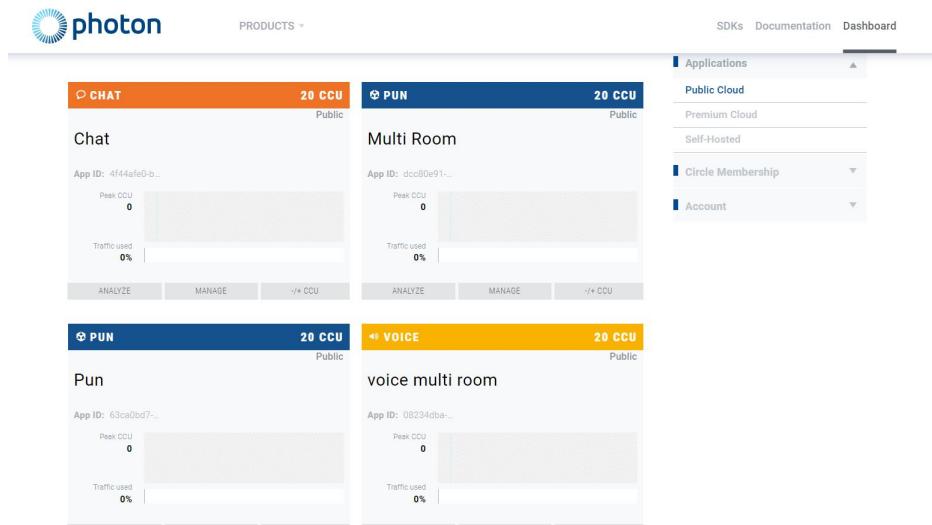


Figura 3.12: Dashboard delle applicazioni Photon Realtime

La **Figura 3.12** mostra la dashboard delle applicazioni *Photon Realtime* descritte in precedenza, l'accesso alla dashboard è consentito dopo aver creato un account *Photon* gratuitamente.

Ogni applicazione, Chat, PUN o Voice, è caratterizzata da un App ID che, dopo aver installato il package PUN dall'Asset Store di *Unity*, va inserito nell'apposito campo nella sezione dell'editor 'Window/Photon Unity Networking/Highlight Server Settings'.

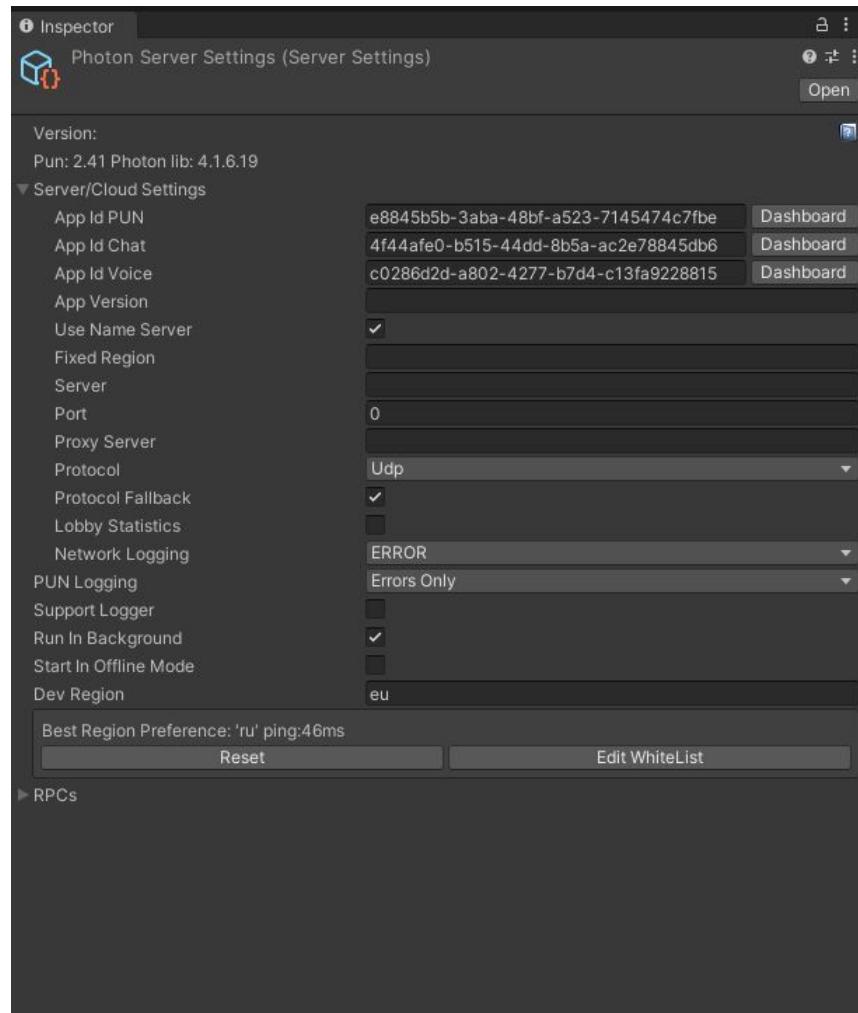


Figura 3.13: Photon Server Settings

A questo punto la fase di preparazione del progetto alle funzionalità network è terminata e si può procedere con la progettazione delle funzioni per la connettività tra utenti.

# Capitolo 4

## Funzionalità dell’utente

In questo capitolo verranno descritte più in dettaglio le funzionalità relative alle azioni dell’utente nella scena finale, in modo che si possa comprendere meglio il lato progettuale dietro il loro effettivo funzionamento nella scena.

È utile, innanzitutto, ricordare che sono state ideate due tipologie di utenti:

- Docente (*Master Client*): gode di particolari privilegi ed è abilitato all’uso di tutte le funzionalità, tranne la *Richiesta di Parlare* (2.2.6);
- Studente (*Generic Client*): non gode di alcun privilegio ed è abilitato all’uso di alcune funzionalità.

### 4.1 XR Origin

Prima di procedere con la descrizione delle funzionalità vere e proprie, è necessario fare una descrizione dell’oggetto che in ogni progetto in Realtà Virtuale non può mancare: **XR Origin**.



Figura 4.1: XR Origin nella Hierarchy del progetto

La presenza di questo oggetto all’interno del progetto è vitale, perché senza esso gli utenti non possono in alcun modo muoversi nell’ambiente, compiere qualsiasi tipo di azione o visualizzare la scena nel proprio visore.

Nella **Figura 4.1** è rappresentato XR Origin con le sue componenti:

- *Main Camera*: l’oggetto che permette la visualizzazione della scena all’utente;
- *LeftHand Controller*: l’oggetto che rappresenta il *controller* sinistro nella scena
- *RightHand Controller*: l’oggetto che rappresenta il *controller* destro nella scena

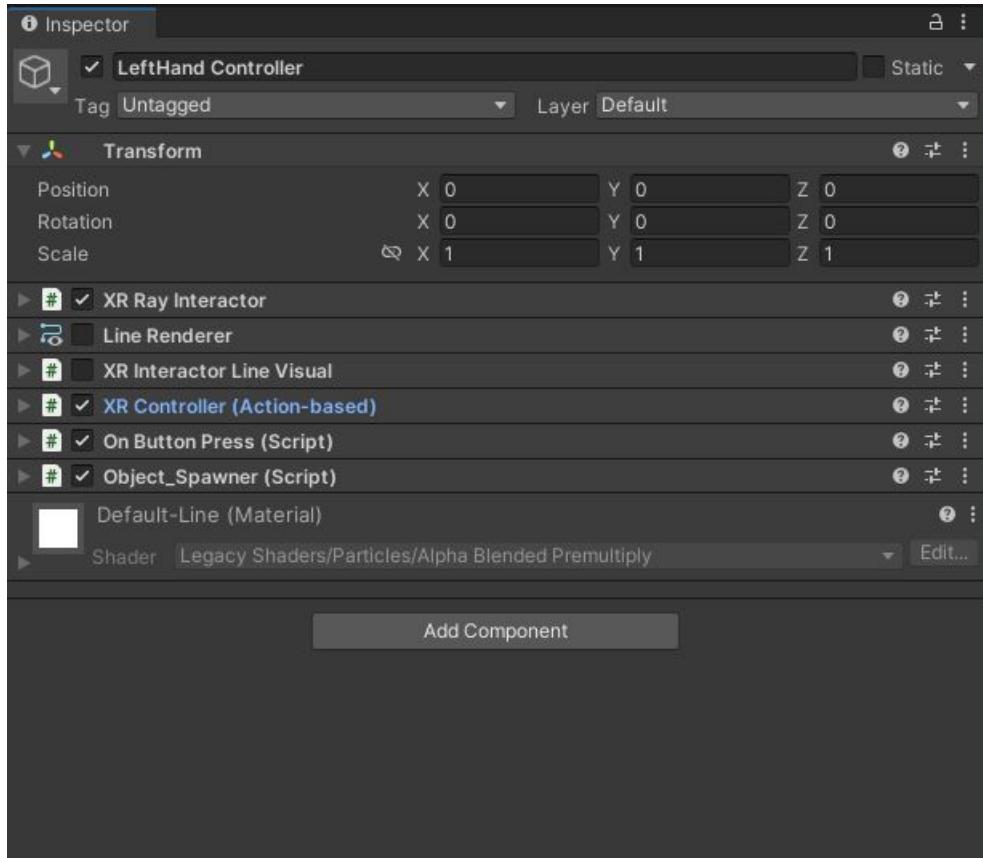


Figura 4.2: Componenti del LeftHand Controller

Nella **Figura 4.2** sono illustrate le componenti che abilitano il *controller* sinistro (*LeftHand Controller*) allo svolgimento di determinate azioni.

Per esempio, le componenti *XR Ray Interactor*, *Line Render*, *XR Interactor Line Visual* si occupano della renderizzazione grafica di un raggio e dell'interazione che ha quest'ultimo con gli oggetti in scena, mentre *XR Controller* è la componente che si occupa di tracciare i vari input che provengono dal *controller* sinistro, quali il movimento lungo i tre assi cardinali e la pressione di uno specifico tasto.

Ovviamente le componenti appena elencate, se applicate all'oggetto *RightHand Controller*, hanno l'effetto analogo sul *controller* destro.

## 4.2 Creazione e distruzione di oggetti

Questa funzionalità è stata progettata per poter permettere al docente di manipolare l'ambiente tramite la creazione e distruzione di oggetti (**bandierine**) in determinati punti della scena (**pannelli**). Abbiamo pensato che questa funzione possa essere utile per segnare luoghi particolari all'interno della scena e renderli, così, più visibili alla vista degli utenti.

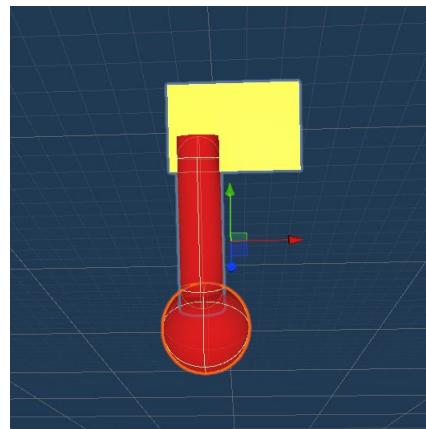


Figura 4.3: Modello della bandierina che viene creata all'interno della scena

Sulla bandierina e sui pannelli sono applicati gli script C# *Photon View* e *Photon Transform View* della libreria *Photon.Pun* che permettono la visualizzazione degli oggetti a tutti gli utenti in scena.

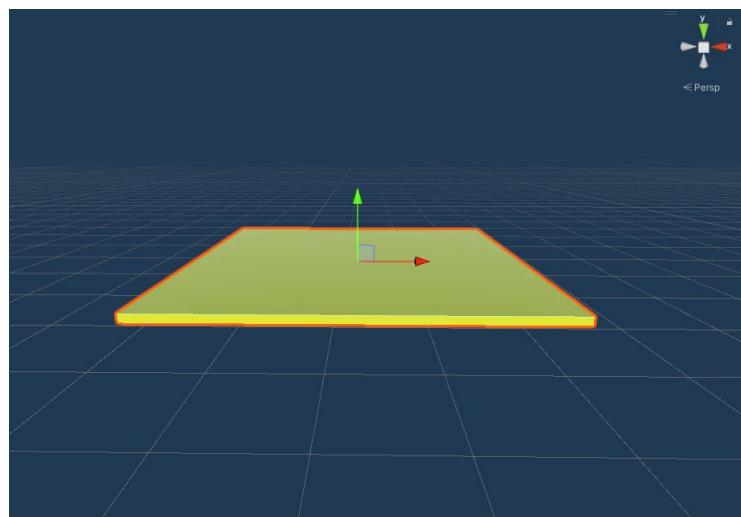


Figura 4.4: Modello del pannello su cui vengono create le bandierine

Inoltre, i pannelli possiedono una componente *Conta Bandiere* che tiene traccia del numero delle bandiere sul pannello.

#### 4.2.1 Object\_Spawner

*Object\_Spawner* è la classe C# che permette al docente di creare le bandierine in scena.

Questa componente è applicata all'oggetto *LeftHand Controller*, per cui le bandierine potranno essere create esclusivamente tramite l'utilizzo del *controller* sinistro.

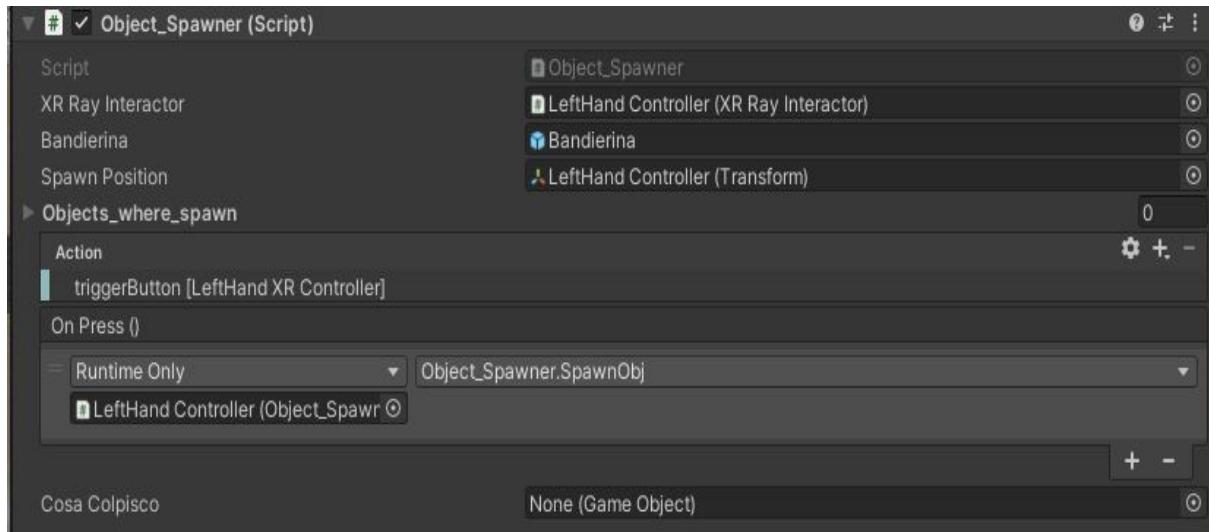


Figura 4.5: Object\_Spawner in dettaglio

Nella **Figura 4.5** sono illustrati i campi pubblici della classe *Object\_Spawner* nell'*Inspector Panel* di Unity.

Dall'alto verso il basso troviamo:

- **XR Ray Interactor:** il raggio del *controller* (nel nostro caso verrà impostato il raggio sinistro);
- **Bandierina:** il modello della bandierina che si vuole creare;
- **Spawn Posizion:** il punto di collisione tra il raggio e un oggetto in scena;
- **Objects\_where\_spawn:** lista di oggetti (pannelli) su cui le bandierine possono essere create;
- **Action:** il comando sul *controller* sinistro (in questo caso il tasto *triggerButton*) che permette di eseguire il metodo indicato nella sezione *On Press()* (il metodo *SpawnObj()*);
- **Cosa Colpisco:** l'oggetto con cui interseca il raggio del *controller* sinistro.

L'abilitazione dell'azione, cioè il tasto sul *controller*, è gestita nel metodo *Update()* della classe *Object\_Spawner*, in seguito verrà mostrato il diagramma di flusso che mostra le condizioni in cui l'azione è abilitata o no.

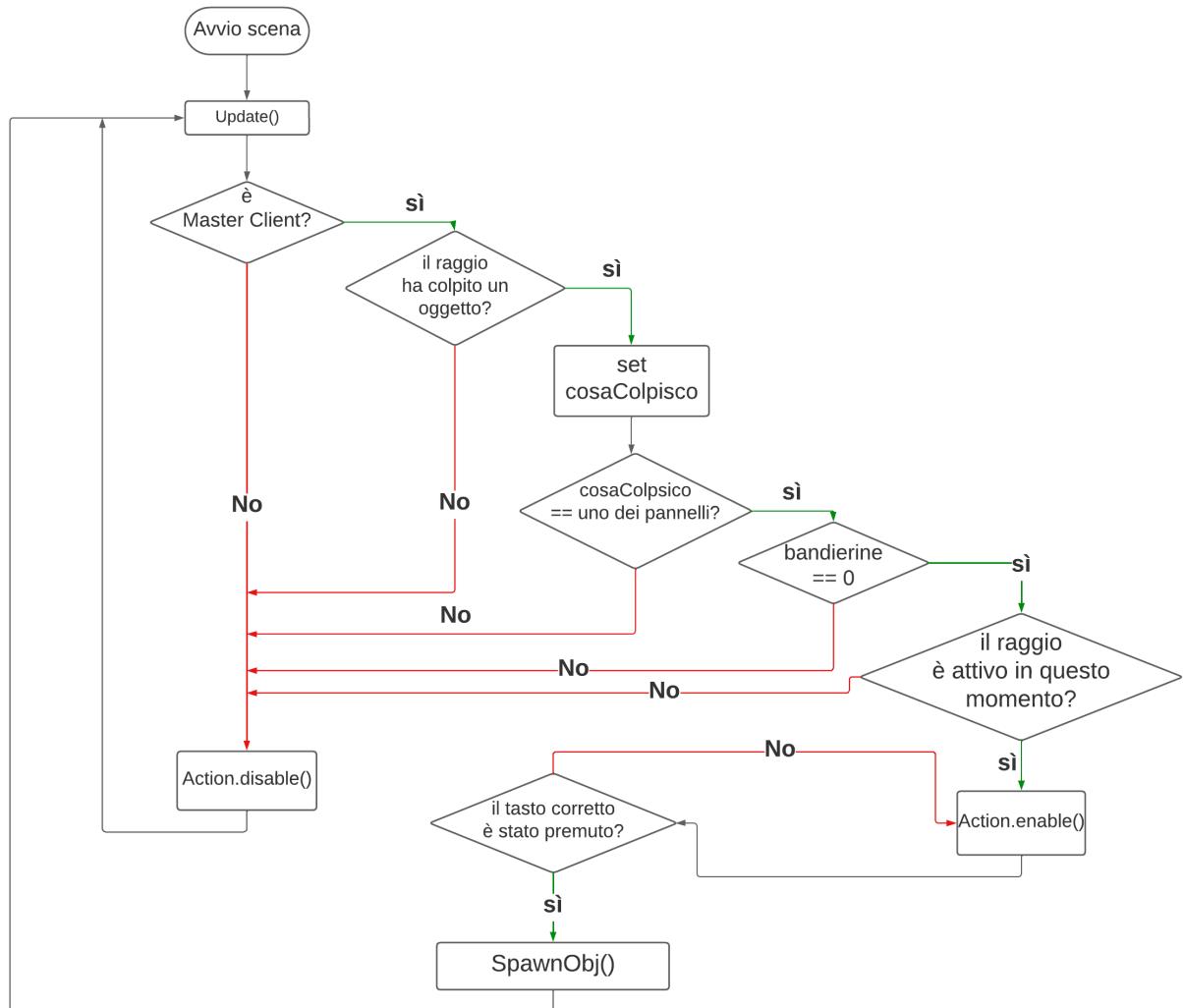


Figura 4.6: Diagramma di flusso del metodo *Update()* della classe *Object\_Spawner*

Il metodo *Update()* viene eseguito costantemente durante l'esecuzione dell'applicazione, perciò le condizioni mostrate nella **Figura 4.6** vengono controllate di continuo.

#### 4.2.2 Object\_Destroyer

*Object\_Destroyer* è la classe C# che permette al docente di distruggere le bandierine in scena. Questa componente è applicata all'oggetto *Bandierina* e le bandierine potranno essere distrutte esclusivamente tramite l'utilizzo del *controller* sinistro.

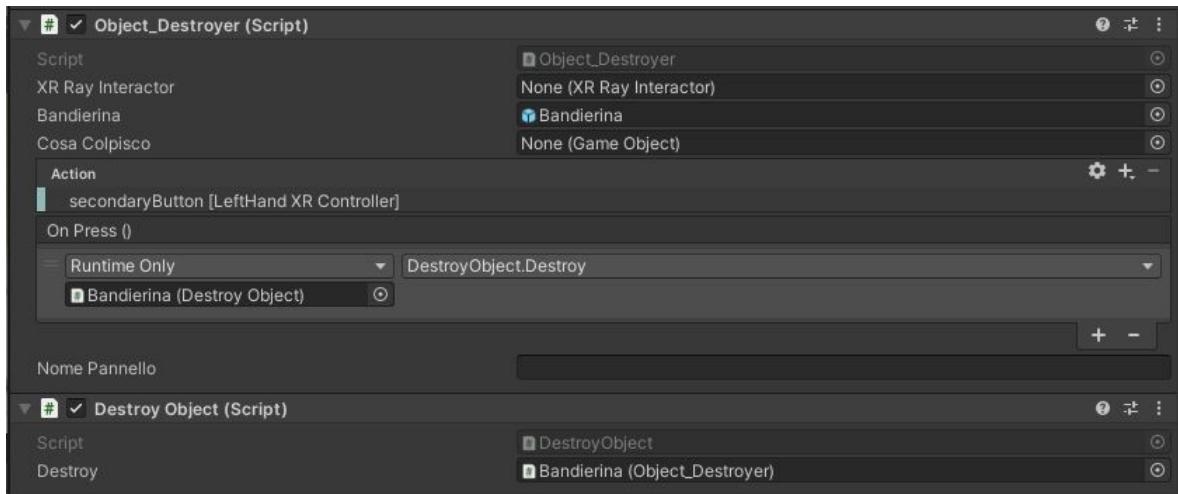


Figura 4.7: Object\_Destroyer e DestroyObject in dettaglio

Nella **Figura 4.7** sono illustrati i campi pubblici della classe *Object\_Destroyer* nell'*Inspector Panel* di Unity.

Dall'alto verso il basso troviamo:

- **XR Ray Interactor:** raggio del *controller* (nel nostro caso verrà impostato il raggio sinistro);
- **Bandierina:** il modello della bandierina che si vuole distruggere;
- **Cosa Colpisce:** l'oggetto con cui interseca il raggio del *controller* sinistro;
- **Action:** il comando sul *controller* sinistro (in questo caso il tasto *secondaryButton*) che permette di eseguire il metodo indicato nella sezione *On Press()* (il metodo *DestroyObject.Destroy*);
- **Nome Pannello:** il nome del pannello su cui la bandierina viene creata;

L'abilitazione dell'azione, cioè il tasto sul *controller*, è gestita nel metodo *Update()* della classe *Object\_Destroyer*, in seguito verrà mostrato il diagramma di flusso che mostra le condizioni in cui l'azione è abilitata o no.

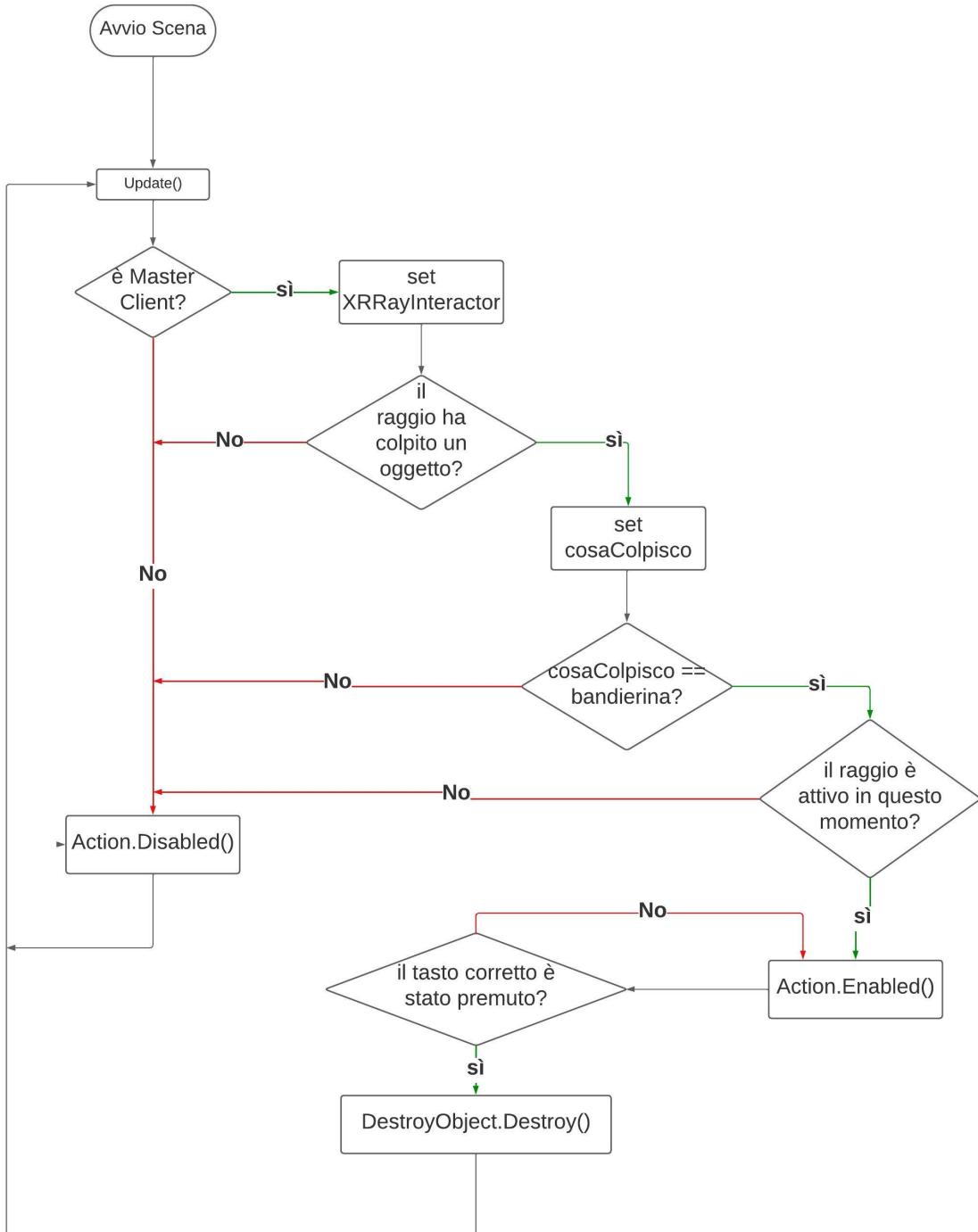


Figura 4.8: Diagramma di flusso del metodo *Update()* della classe *Object\_Destroyer*

Anche in questo caso, il metodo *Update()* viene eseguito costantemente durante l'esecuzione dell'applicazione, perciò le condizioni mostrate nella **Figura 4.8** vengono controllate di continuo.

La differenza rispetto al caso della creazione delle bandierine, è che il metodo abilitato appartiene ad una classe esterna.

Per concludere, si può notare che le funzionalità appena descritte sono delegate ai comandi del *con-*

*troller* sinistro, questa scelta è stata effettuata sia per una comodità d'uso, per un utente potrebbe risultare scomodo utilizzare una funzionalità sul *controller* sinistro e una sul *controller* destro, sia in ottica sviluppi futuri, applicare le stesse funzionalità ad entrambi i *controller* costringe lo sviluppatore a toglierle da un *controller* prima di poterne applicare altre.

### 4.3 Unity Canvas e Interfaccia Utente

Le funzionalità che verranno illustrate in seguito sono relative all'interfaccia utente, che viene realizzata con gli oggetti **Unity Canvas**.

Il Canvas è un oggetto di gioco che possiede una componente *Canvas* e tutti gli elementi dell'interfaccia utente devono essere figli di tale Canvas.

L'area del Canvas viene mostrata come un rettangolo nella scena, ciò permette una maggiore semplicità nel posizionamento degli elementi dell'interfaccia utente.

Gli elementi dell'interfaccia utente sono di vario tipo, ad esempio componenti visuali (come testo e immagini) e componenti interattive (come pulsanti, barre di testo in cui è possibile scrivere, menù a tendina, barre di scorrimento e altro ancora).

Le componenti interattive sono attivabili tramite l'utilizzo di dispositivi di input, come ad esempio mouse, tastiera, *controller* VR, microfono ecc...

### 4.4 Chat interattiva

Questa funzionalità è stata ideata per aggiungere un altro tipo di comunicazione tra utenti oltre a quella vocale.

L'interfaccia utente della chat è stata realizzata con gli oggetti **Unity Canvas**, in particolare è strutturata in 3 sezioni:

- Join Room: Canvas che contiene il bottone per connettersi alla chat;

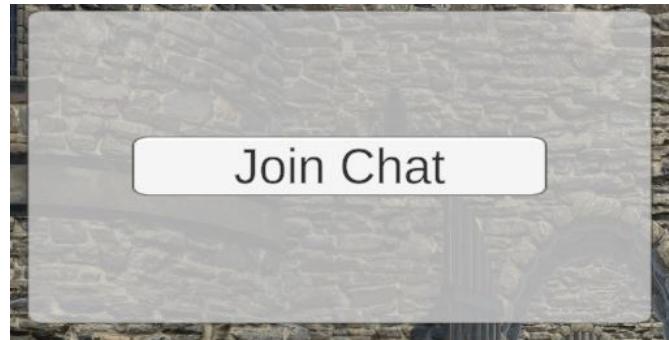


Figura 4.9: Join Room Canvas

- Chat Room: Canvas che rappresenta la chat vera e propria;

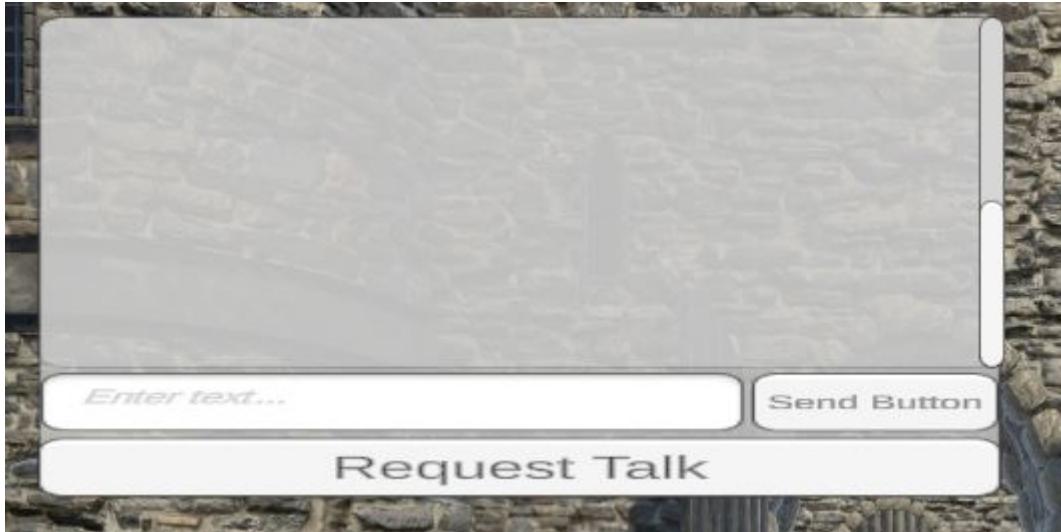


Figura 4.10: Chat Room Canvas

Per inviare i messaggi è necessario innanzitutto scrivere, tramite la tastiera, nel campo con il testo ‘Enter text’, e successivamente premere il bottone ‘Send Button’.

È presente anche un bottone (‘Request Talk’) per l’invio sulla chat di un messaggio prefabbricato, che lo studente può utilizzare qualora volesse richiedere l’attivazione del proprio microfono al docente.

- Open Chat: Canvas con un apposito bottone che permette di aprire *Chat Room*, è stato impostato un timer di 30 secondi che chiude *Chat Room* se in quel lasso di tempo non sono arrivati messaggi sulla chat;

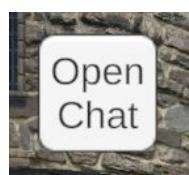


Figura 4.11: Open Chat Canvas

Per comprendere meglio il funzionamento della chat, verranno ora illustrati i diagrammi di sequenza relativi alle 3 sezioni della chat mostrate in precedenza.

I diagrammi di sequenza illustrano come le diverse parti di un sistema interagiscono tra loro per svolgere una funzione, e l’ordine in cui le interazioni avvengono quando un particolare caso d’uso viene eseguito.

In parole più semplici, un diagramma di sequenza mostra diverse parti del lavoro di un sistema in una ‘sequenza’ per ottenere qualcosa.

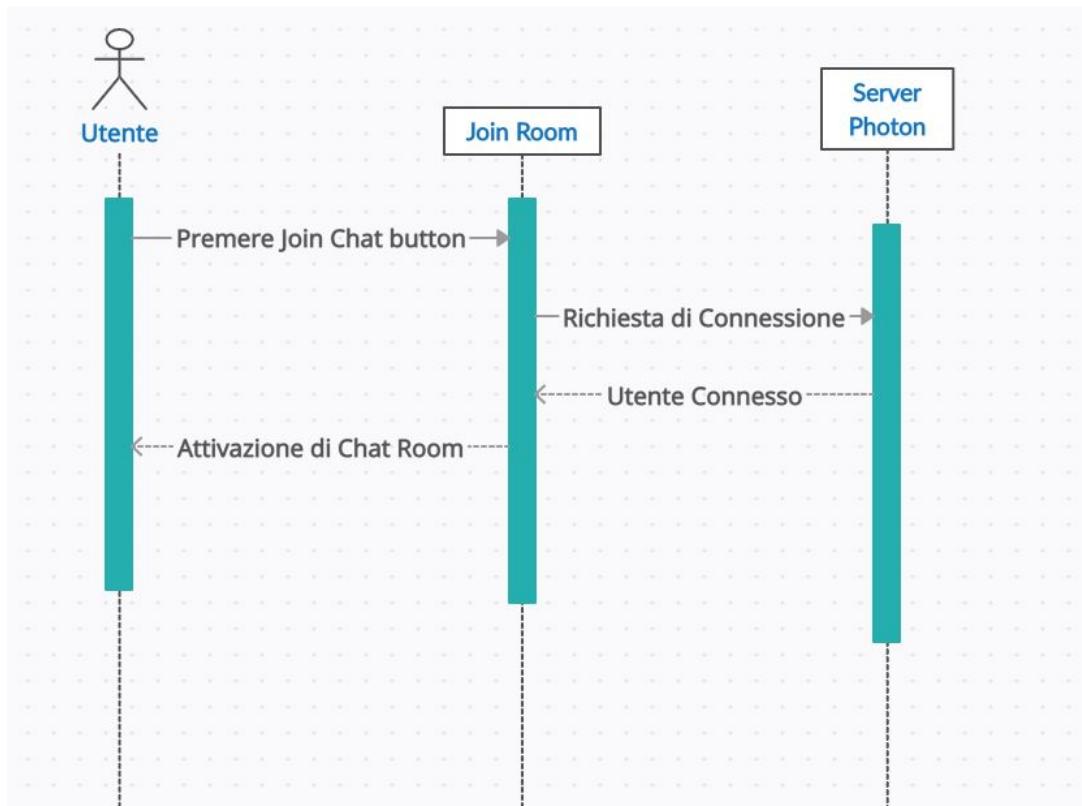


Figura 4.12: Diagramma di sequenza per Join Room

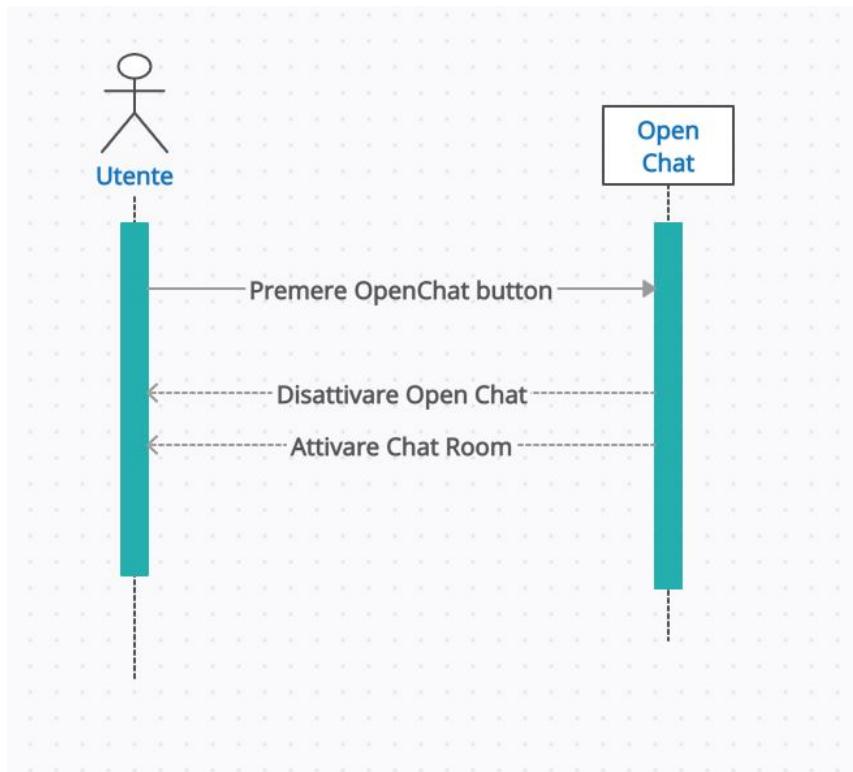


Figura 4.13: Diagramma di sequenza per Open Chat

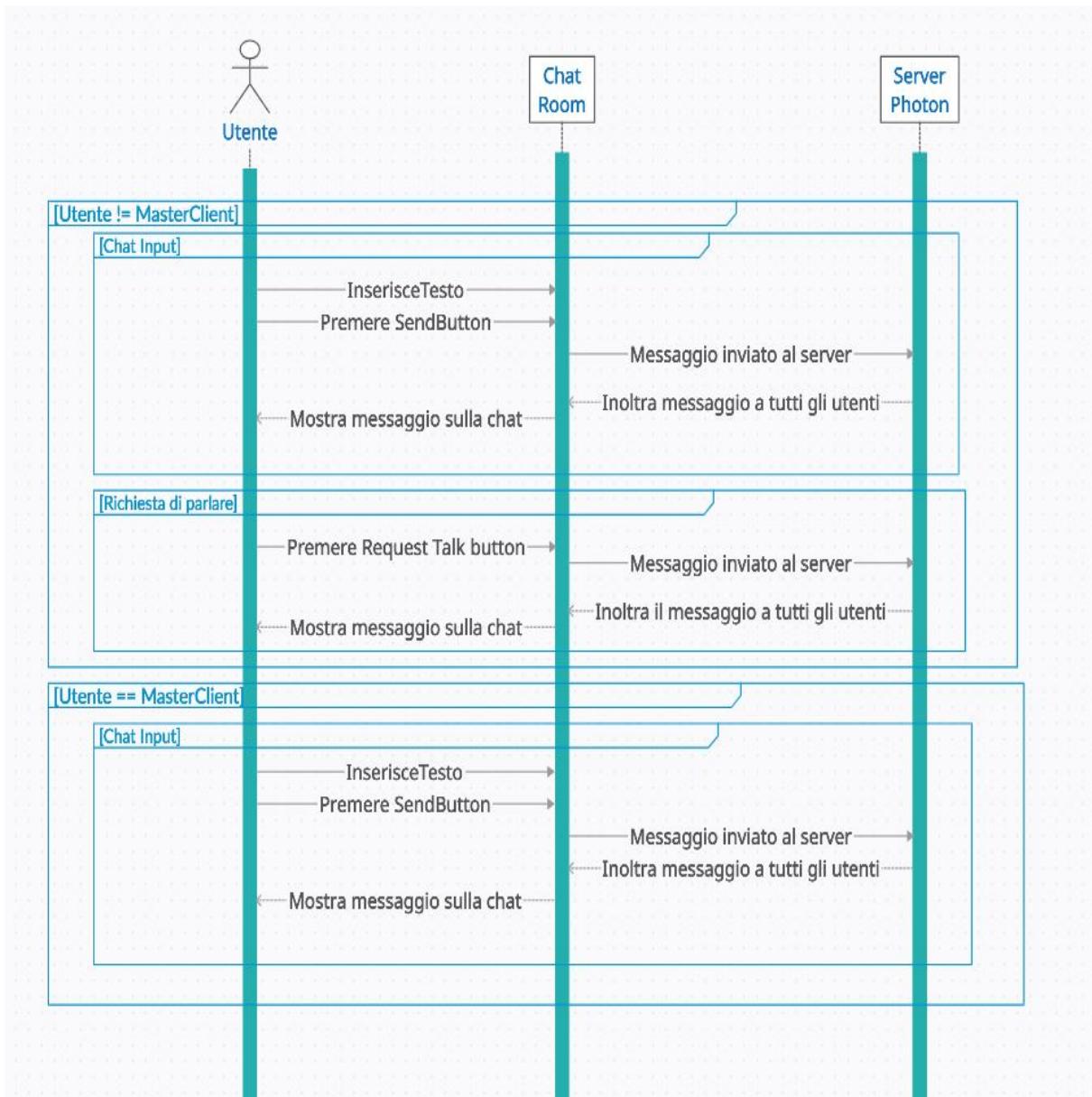


Figura 4.14: Diagramma di sequenza per Chat Room

## 4.5 Abilitazione e disabilitazione audio utenti

Questa funzionalità permette al docente di attivare e disattivare il microfono di tutti gli utenti presenti nella scena.

Anche in questo caso è presente un'interfaccia utente, realizzata con gli oggetti **Unity Canvas**, strutturata in due parti:

- Apri Impostazioni: Canvas con un apposito bottone dedicato all'apertura della Schermata Utenti;

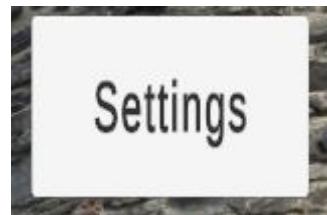


Figura 4.15: Apri Impostazioni Canvas

- Schermata Utenti: Canvas con la lista degli utenti connessi alla stanza con un bottone apposito per attivare o disattivare il microfono del giocatore desiderato.

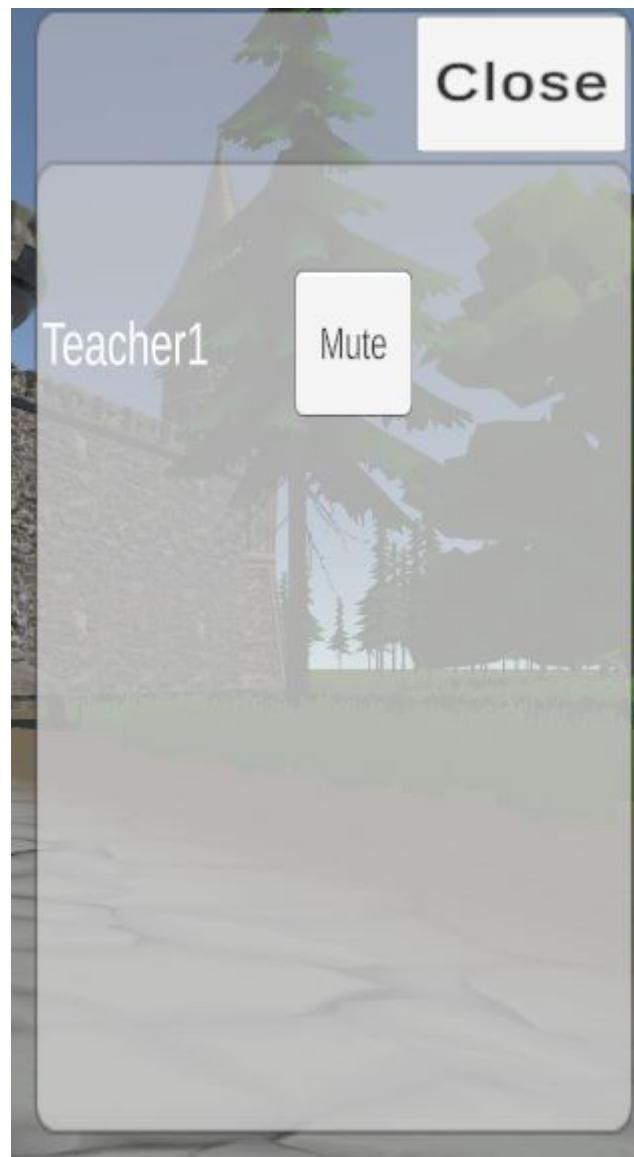


Figura 4.16: Schermata Utenti Canvas

Anche in questo caso verrà illustrato il diagramma di sequenza per comprendere meglio il meccanismo

di attivazione e disattivazione del microfono degli utenti.

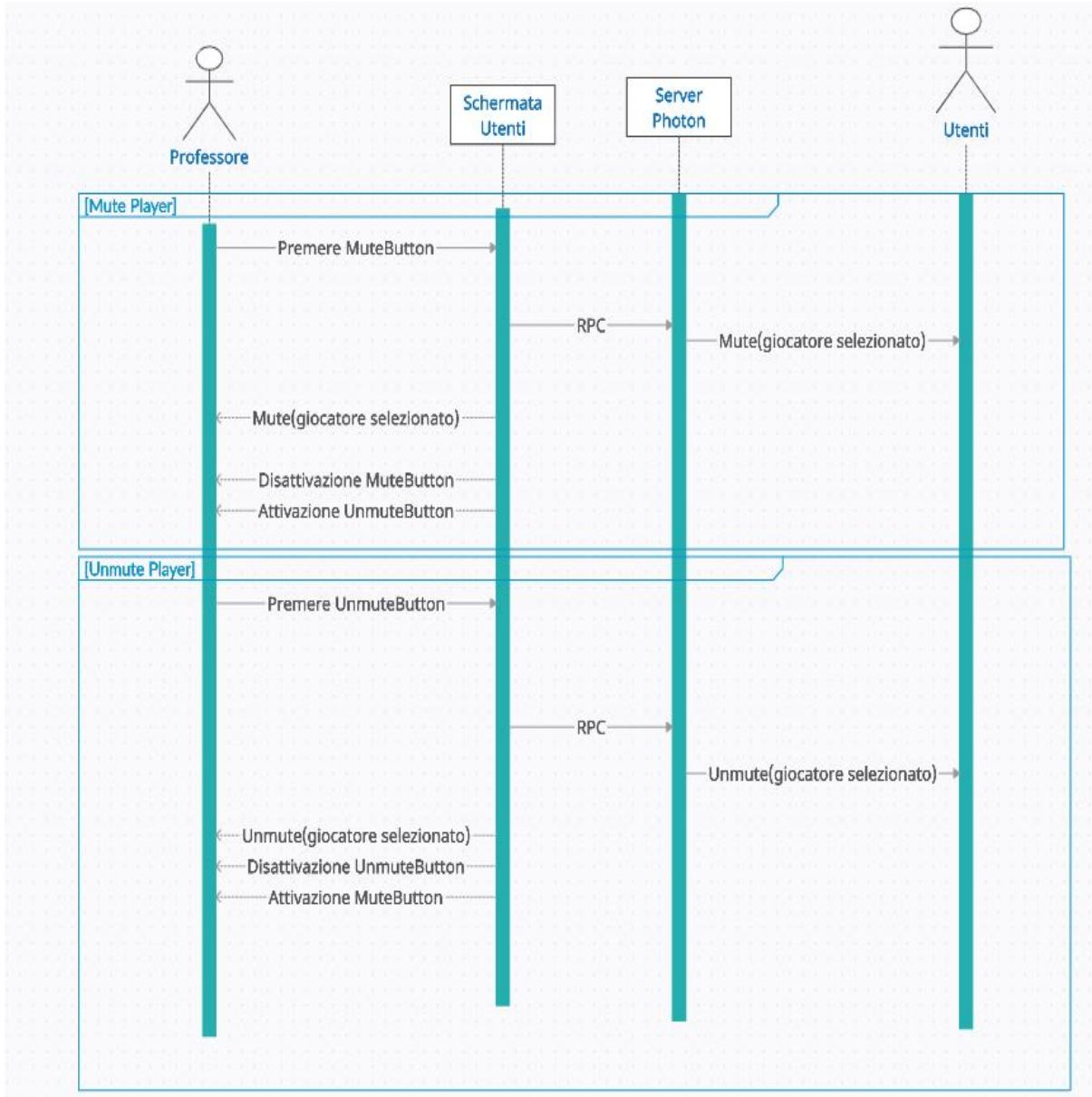


Figura 4.17: Diagramma di sequenza per l'attivazione e disattivazione del microfono

Per sviluppare questa funzionalità è stato necessario ricorrere ad una particolare proprietà della libreria *Photon.Pun*: le **RPC**.

Le chiamate di procedura remota (*RPC*) sono il modo in cui *Photon* esegue eventi o notifiche che si basano sul networking.

Impostando l'attributo *RPC* su un metodo, esso viene richiamato anche per gli altri utenti connessi alla rete *Photon* o che si connetteranno in seguito.

In questo caso specifico, la chiamata di procedura remota è applicata al metodo che attiva o disattiva la componente microfono di un giocatore.

# Capitolo 5

## Dimostrazioni d'uso

In questo capitolo verranno mostrate alcune immagini, dell'applicazione per desktop, che hanno lo scopo di mostrare il funzionamento delle specifiche software che sono state ideate per la realizzazione del prototipo.

### 5.1 Identificazione

La **Figura 5.1** e **Figura 5.2** mostrano come il sistema faccia apparire, così come indicato nelle specifiche, un messaggio di errore all'utente quando inserisce delle credenziali errate nei pannelli *username* e *password*.

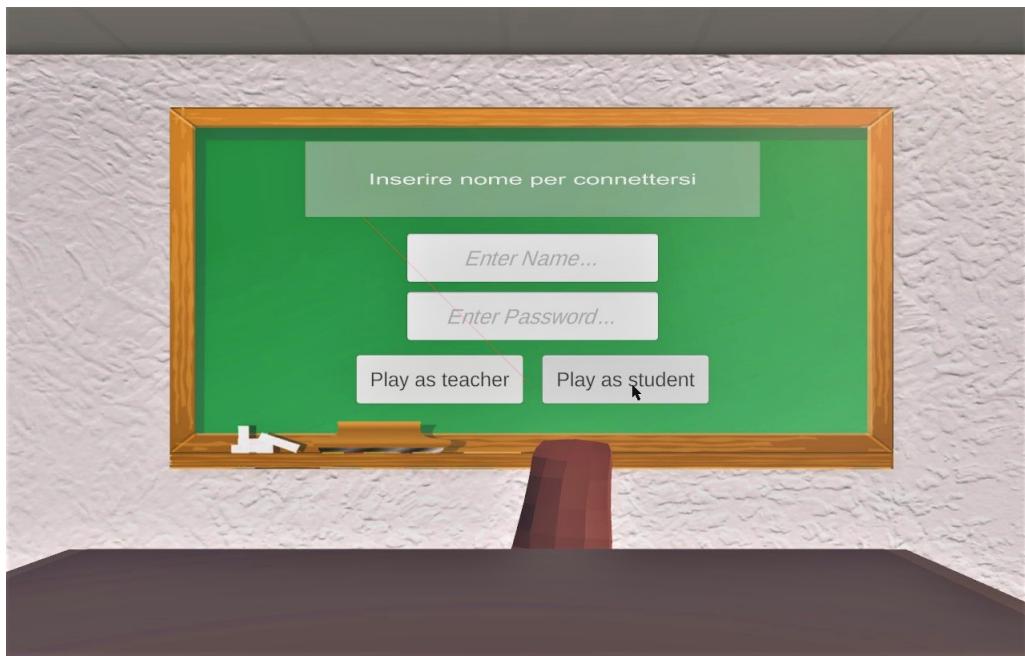


Figura 5.1: Schermata di errore per lo studente



Figura 5.2: Schermata di errore per il docente

La versione per desktop presenta un problema quando viene eseguita sul visore di riferimento, ovvero l’Oculus Quest 2.

Per delle limitazioni presenti nell’XR Origin, non è possibile utilizzare la tastiera di sistema.

Per ovviare a questo problema, è stato deciso di inserire, nella versione per Oculus Quest 2, un nome di default per gli studenti in modo da poter permettere il login anche tramite visore quando l’utente prova ad accedere senza inserire il nome.

## 5.2 Lobby



Figura 5.3: Lobby docente

La **Figura 5.3** rappresenta la scena che verrà caricata dall'applicazione dell'utente se è stato effettuato l'accesso come docente.

Si può notare la presenza del pulsante per creare la stanza (pulsante *Create Room*) e dell'elenco delle stanze disponibili per l'accesso.

In particolare, nella figura è presente un messaggio di errore che compare quando il docente cerca di creare una stanza senza nome.

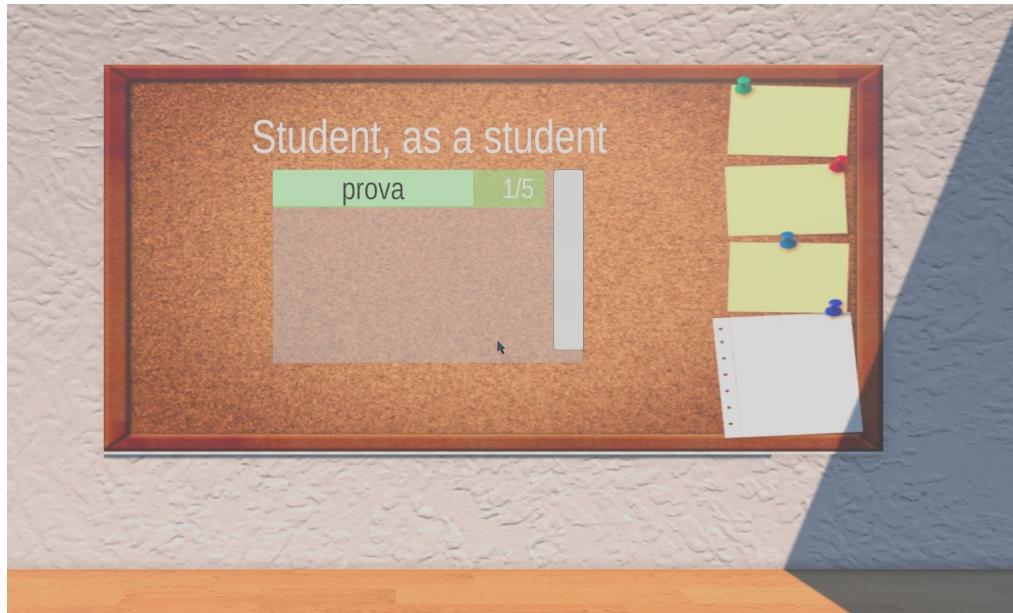


Figura 5.4: Lobby studente

La scena rappresentata nella **Figura 5.4** è quella caricata dall'applicazione dell'utente se è stato effettuato l'accesso come studente.

Lo studente, come già detto nelle specifiche e nei casi d'uso, visualizzerà solo l'elenco delle stanze a cui può unirsi.

### 5.3 Generic Room

La **Generic Room** rappresenta la scena del prototipo adibita all'interazione fra gli utenti, ovvero la parte del progetto dove sono implementate le funzionalità descritte nel **Cap.4**.



Figura 5.5: Visualizzazione fra utenti

Uno dei requisiti fondamentali illustrato nelle specifiche del prototipo è quello della percezione fra utenti.

La **Figura 5.5** rappresenta la visuale di un utente (*Teacher*), nella quale è ben visibile un secondo utente (*Student*).

### 5.3.1 Chat di testo e Impostazioni

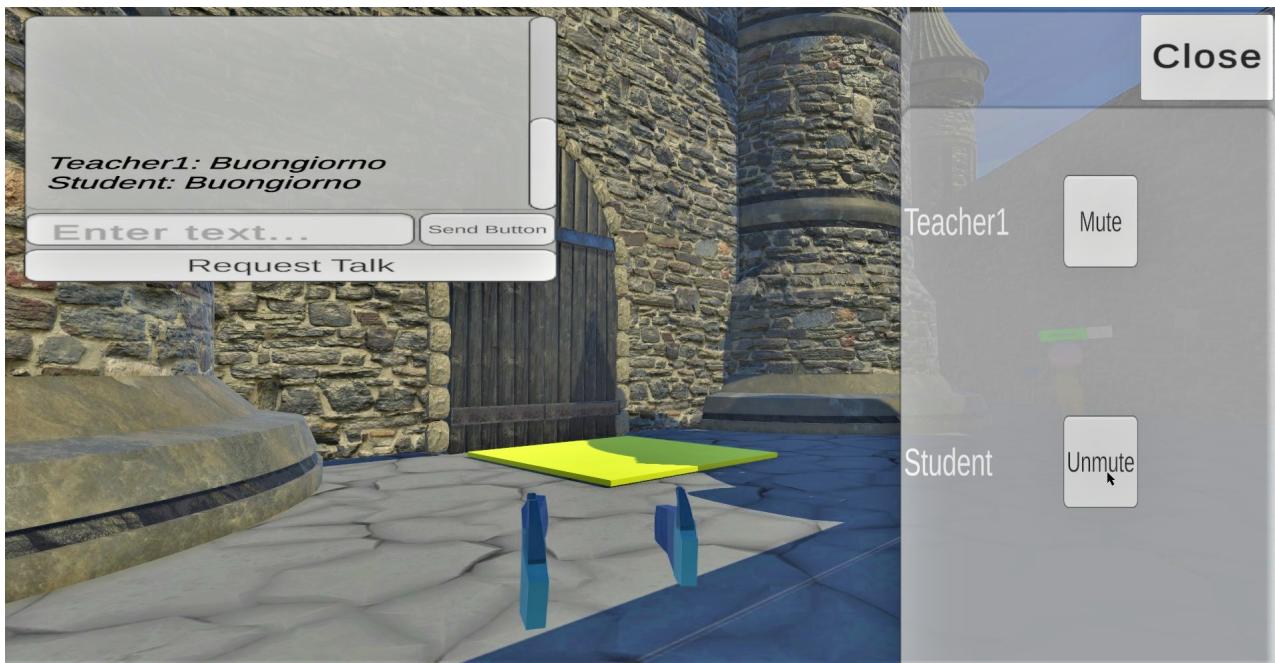


Figura 5.6: Funzionamento della Chat e delle Impostazioni

Nella **Figura 5.6** si può notare la presenza della chat sul lato sinistro in cui sono presenti due messaggi, uno inviato dal docente (*Teacher 1*) e uno inviato dallo studente (*Student*).

Le impostazioni, che sono accessibili solamente dal docente, si trovano sul lato destro della schermata e, come previsto dai requisti, presentano una lista con i giocatori connessi alla stanza e un bottone per ogni giocatore per l'attivazione o disattivazione del microfono.

In particolare, nella figura si nota che il docente ha disattivato il microfono dello studente.

La scritta sul bottone accanto al nome dello studente è passata da ‘Mute’ (disattiva il microfono) a ‘Unmute’ (attiva il microfono), se il bottone verrà premuto dal docente, il microfono verrà di nuovo attivato.

### 5.3.2 Creazione/Eliminazione di bandierine

Le immagini che seguiranno hanno scopo di mostrare sequenzialmente il funzionamento della creazione e distruzione delle bandierine da parte del docente.

Saranno presenti immagini sia dal punto di vista del docente sia da quello dello studente, per mostrare in modo ottimale lo svolgimento delle funzionalità.

La presenza della chat, nelle suddette immagini, risulta utile per percepire la contemporaneità degli eventi mostrati, infatti sia dal punto di vista dello studente che da quello del docente sono presenti gli stessi messaggi.



Figura 5.7: Scena prima del piazzamento della bandierina

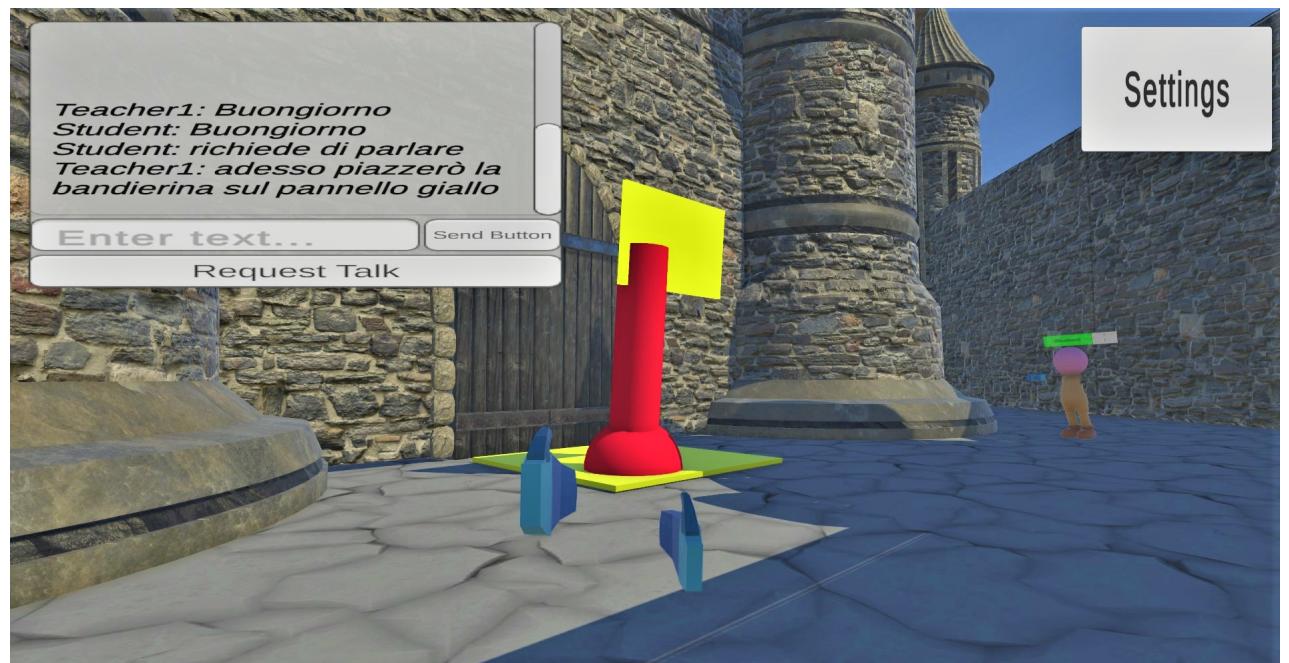


Figura 5.8: Scena dopo il piazzamento della bandierina lato docente

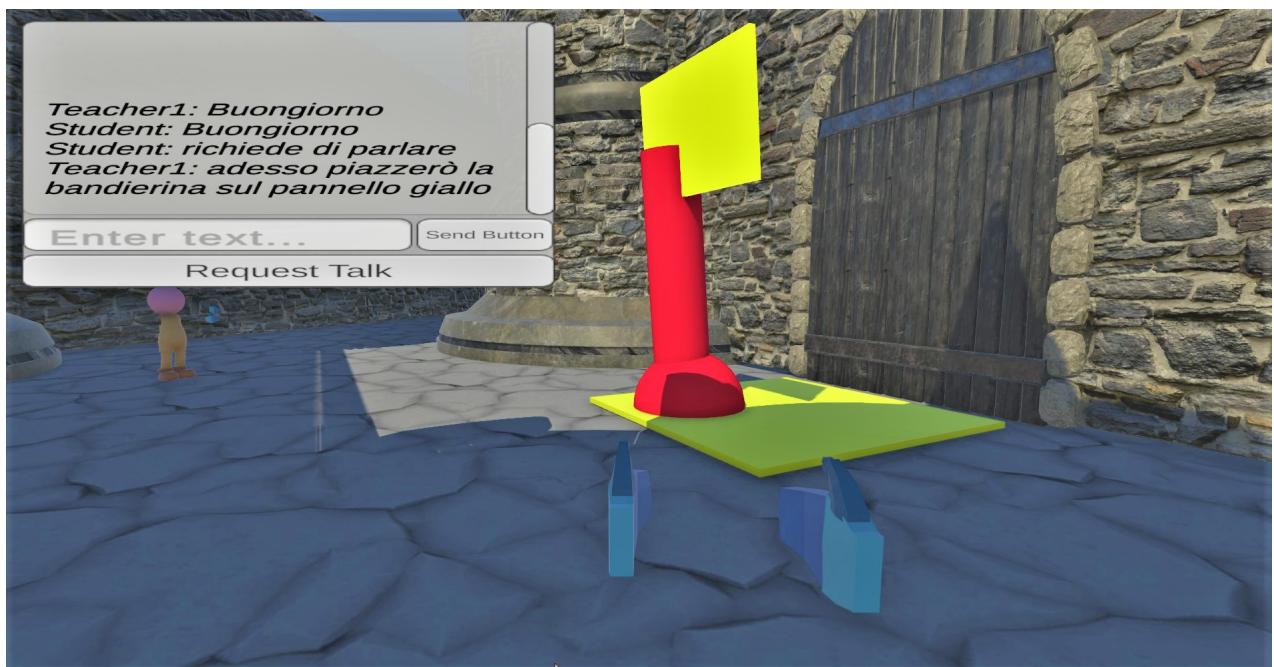


Figura 5.9: Scena dopo il piazzamento della bandierina lato studente

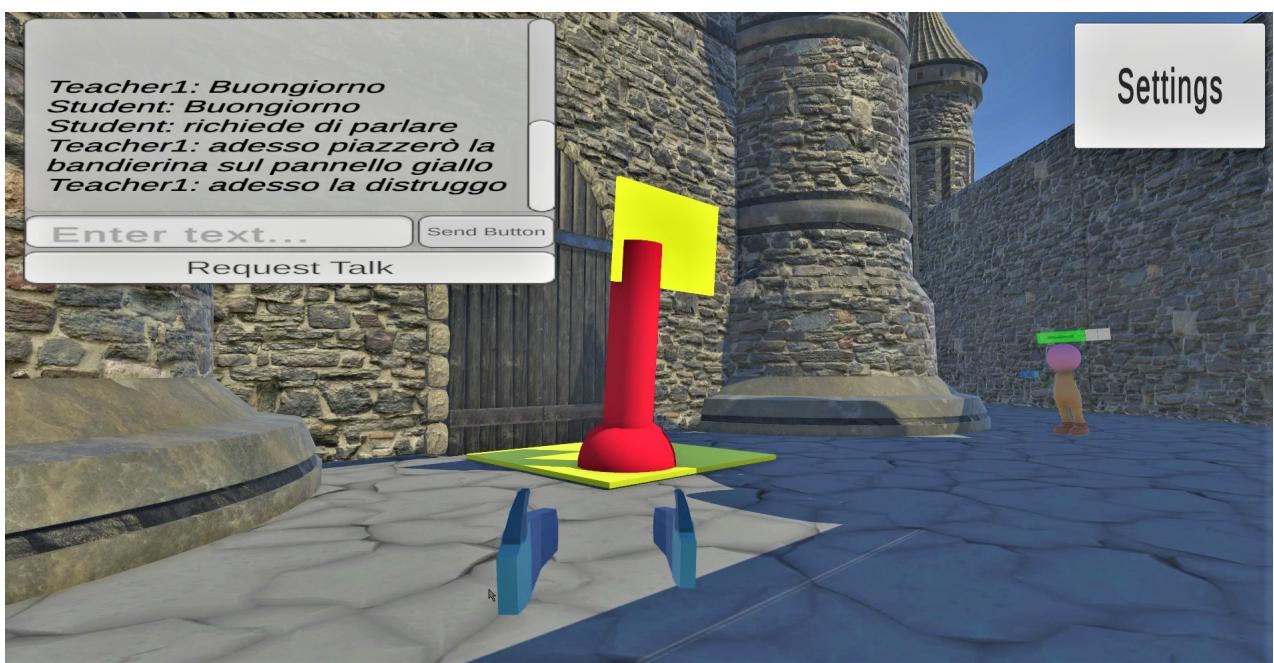


Figura 5.10: Scena prima della distruzione della bandierina



Figura 5.11: Scena dopo la distruzione della bandierina lato docente

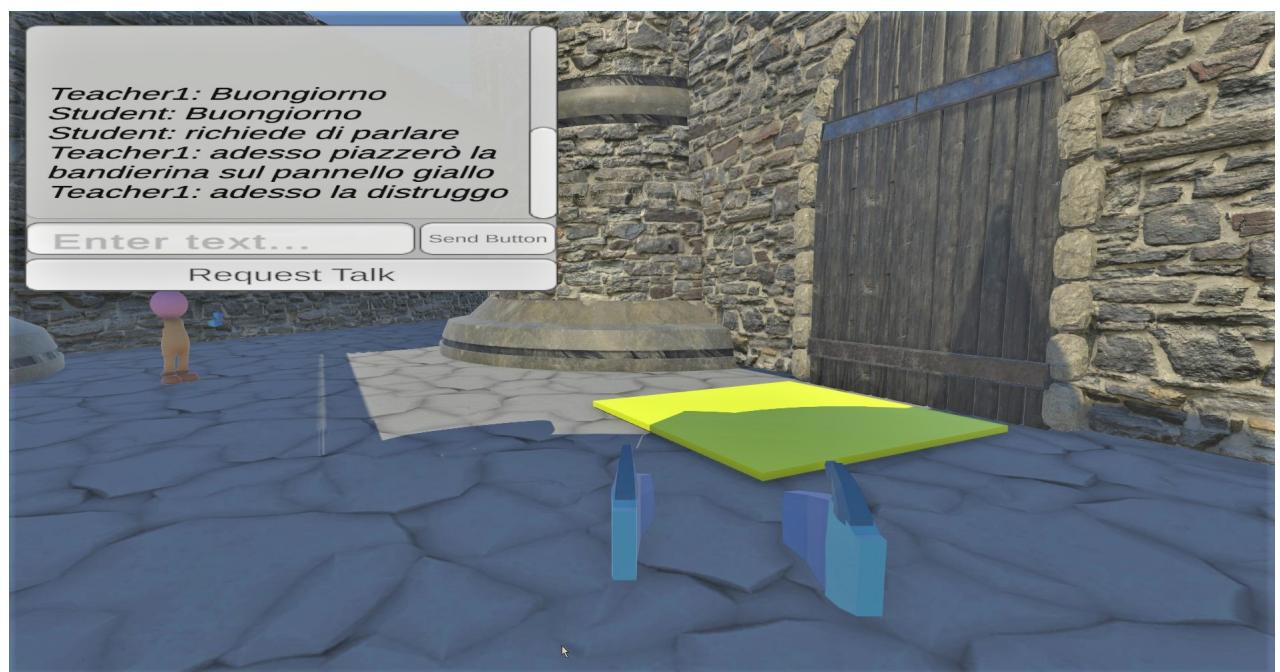


Figura 5.12: Scena dopo la distruzione della bandierina lato studente

# Capitolo 6

## Sviluppi futuri

Nei capitoli precedenti sono state illustrate le fasi del progetto, partendo dai requisiti che sono stati fissati fino allo sviluppo delle funzionalità vere e proprie del *software*.

Il progetto descritto in questa relazione è, appunto, un prototipo, quindi ha molti aspetti che, in futuro, potranno essere migliorati e altri elementi che potranno essere aggiunti al fine di renderlo un prodotto effettivamente utilizzabile a scopi didattici.

### 6.1 Sviluppi Futuri

Saranno ora illustrati alcuni aspetti che sono stati pensati per arricchire e migliorare il prototipo.

Le funzionalità che verranno descritte rappresentano una piccola parte rispetto a tutte quelle che si possono ideare, perché, come è ben noto, il mondo della progettazione e sviluppo *software* è ricco di possibilità e alternative.

#### 6.1.1 Visibilità dei Raggi

Si prenda, per esempio, uno studente che voglia chiedere informazioni su un punto preciso della scena, tramite la chat vocale o scritta potrebbe risultare complesso dare indicazioni precise sul punto che vuole mettere in risalto, visto che con quegli strumenti non può mostrare agli utenti quello che lui sta visualizzando nel proprio visore.

Per facilitare questa operazione, potrà essere estremamente utile rendere i raggi di ogni giocatore visibili anche agli altri utenti nella scena.

In questo modo, se uno studente volesse indicare con uno dei due raggi un punto preciso, anche gli altri utenti vedranno il raggio dello studente che sta indicando quel punto.

#### 6.1.2 Avatar realistici

In questo momento, per rappresentare gli utenti in scena, è presente un stilizzato di un essere umano. In futuro si potrebbero implementare i seguenti miglioramenti:

- Un *Avatar* che sia simile, il più possibile, ad un essere umano;
- Movimenti delle braccia e delle gambe dell'*Avatar* quando l'utente muove i *controller* o si muove nella scena;

- Più modelli di *Avatar*, in modo che un utente possa scegliere quello che più desidera.

### 6.1.3 Ambientazione esistente

Lo scenario finale, in cui tutti i giocatori si muovono e interagiscono, rappresenta un castello medioevale, con due piccoli villaggi ai lati, immerso nella natura.

Nell'introduzione è stato spiegato questo scenario è fittizio, siccome lo scopo del progetto è quello di immergere gli utenti in uno scenario reale, in futuro sarà sicuramente necessario importare la rappresentazione digitale di un'ambientazione esistente.

### 6.1.4 Sistema di Registrazione e Autenticazione

Un'altra meccanica sicuramente migliorabile è quella della registrazione e dell'autenticazione dell'utente.

Per quanto riguarda il docente, è presente un file con alcune coppie utente/password per poter passare dalla scena iniziale alla lobby docente.

Per lo studente, invece, non è presente alcun tipo di autenticazione vera e propria, deve solo inserire una qualsiasi stringa nel campo apposito per poter passare dalla scena iniziale alla lobby studente.

Inoltre, non è presente un sistema per la registrazione in un database dello *username* degli utenti, questo problema comporta la possibile presenza di due o più utenti con lo stesso *username*.

Esistono alcune aziende che offrono una soluzione per l'aggiunta un sistema di registrazione e autenticazione degli utenti per le applicazioni, una di queste è **Auth0**<sup>1</sup>.

---

<sup>1</sup>(<https://auth0.com/>) sito web di Auth0

# Bibliografia e Sitografia

- [1] Jason Jerald. *The VR Book: Human-Centered Design for Virtual Reality*. Association for Computing Machinery e Morgan & Claypool, October 2015.
- [2] Wikipedia. *Virtual Reality Headset*. Gen. 2023.  
URL: [https://en.wikipedia.org/wiki/Virtual\\_reality\\_headset](https://en.wikipedia.org/wiki/Virtual_reality_headset).
- [3] Wikipedia. *Unity Game Engine*. Gen. 2023.  
URL: [https://en.wikipedia.org/wiki/Unity\\_\(game\\_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine)).
- [4] Unity Technologies. *The Main Windows*. Gen. 2023.  
URL: <https://docs.unity3d.com/560/Documentation/Manual/UsingTheEditor.html>.
- [5] Unity Technologies. *The Hierarchy Window*. Gen. 2023.  
URL: <https://docs.unity3d.com/560/Documentation/Manual/Hierarchy.html>.
- [6] Unity Technologies. *The Scene View*. Gen. 2023.  
URL: <https://docs.unity3d.com/560/Documentation/Manual/UsingTheSceneView.html>.
- [7] Unity Technologies. *The Game View*. Gen. 2023.  
URL: <https://docs.unity3d.com/560/Documentation/Manual/GameView.html>.
- [8] Unity Technologies. *The Inspector Window*. Gen. 2023.  
URL: <https://docs.unity3d.com/560/Documentation/Manual/UsingTheInspector.html>.
- [9] Unity Technologies. *The Project Window*. Gen. 2023.  
URL: <https://docs.unity3d.com/560/Documentation/Manual/ProjectView.html>.
- [10] Unity Technologies. *Unity Console*. Gen. 2023.  
URL: <https://docs.unity3d.com/560/Documentation/Manual/Console.html>.
- [11] InfodocScuola. *Introduzione al linguaggio C# e a .NET Framework*. Gen. 2023.  
URL: <http://infodoc.altervista.org/guida-csharp/introduzione-al-linguaggio-csharp-e-a-net-framework/>.
- [12] Ciro Continisio. *MonoBehaviour, gli eventi di Unity*. Gen. 2023.  
URL: <https://www.html.it/pag/45529/monobehaviour-gli-eventi-di-unity/#:~:text=Monobehaviour%20%C3%A8%20la%20classe%20dalla,al%20verificarsi%20di%20certe%20condizioni..>
- [13] Unity. *MonoBehaviour*. Gen. 2023.  
URL: <https://docs.unity3d.com/ScriptReference/MonoBehaviour.html>.
- [14] Unity. *UnityEvents*. Gen. 2023.  
URL: <https://docs.unity3d.com/Manual/UnityEvents.html>.

- [15] Unity. *Namespace UnityEngine.InputSystem*. Gen. 2023. URL: <https://docs.unity3d.com/Packages/com.unity.inputsystem@0.2/api/UnityEngine.InputSystem.html>.
- [16] Unity. *XR Interaction Toolkit*. Gen. 2023. URL: <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.2/manual/index.html>.
- [17] Unity. *Namespace UnityEngine.UI*. Gen. 2023. URL: <https://docs.unity3d.com/Packages/com.unity.ugui@1.0/api/UnityEngine.UI.html>.
- [18] Microsoft. *System.Collections Spazio dei nomi*. Gen. 2023. URL: <https://learn.microsoft.com/it-it/dotnet/api/system.collections?view=net-7.0>.
- [19] Microsoft. *System.Collections.Generic Spazio dei nomi*. Gen. 2023. URL: <https://learn.microsoft.com/it-it/dotnet/api/system.collections.generic?view=net-7.0>.
- [20] Unity. *TextMeshPro*. Gen. 2023. URL: <https://docs.unity3d.com/Manual/com.unity.textmeshpro.html>.
- [21] Photon. *Photon PUN*. Feb. 2023. URL: <https://www.photonengine.com/pun>.
- [22] Photon. *Photon Chat Intro*. Feb. 2023. URL: <https://doc.photonengine.com/chat/current/getting-started/chat-intro>.
- [23] Photon. *Photon Voice Intro*. Feb. 2023. URL: <https://doc.photonengine.com/voice/current/getting-started/voice-intro>.
- [24] Photon Engine. *Photon Engine*. Gen. 2023. URL: <https://www.photonengine.com/>.