UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea triennale in

Ingegneria dell'informazione per Videogame e Realtà Virtuale

Studio e sviluppo di un sistema di audio immersivo in Unity

Study and development of an immersive audio system in Unity

Relatore: Chiar.ma

Prof.ssa **Stefania Cecchi**

Tesi di Laurea di:

Gianmarco Diomedi

Correlatore

Dr. **Alessandro Terenzi**

A.A. 2025/2026

“Se ho visto più lontano è perché sono salito sulle spalle di giganti”

Ai miei giganti.

Abstract

I moderni ambienti di sviluppo di realtà virtuale offrono molti strumenti per creare scenari ricchi e immersivi sotto il punto di vista visivo, ma non ci sono ancora altrettanti strumenti utili a ricreare suoni tridimensionali coerenti con la geometria presente nella scena. Questo lavoro di tesi ha come obiettivo quello di studiare e sviluppare un sistema di audio immersivo in ambiente virtuale per visori 3D utilizzando il motore di gioco Unity 3D. Il metodo scelto per lo sviluppo del sistema è l'Image Source Method (ISM), che tiene conto della geometria della scena virtuale per calcolare le riflessioni del suono nell'ambiente. Il sistema sviluppato è stato poi testato e i risultati ottenuti sono stati comparati con misurazioni reali fatte utilizzando il plugin di audio immersivo del visore Meta Quest.

SOMMARIO

ELENCO DELLE FIGURE

1. INTRODUZIONE

L'utilizzo dei motori di gioco moderni non si limita al mondo dell'intrattenimento, ma vengono largamente impiegati anche in ambiti di simulazione industriale, addestramento in scenari critici, telemedicina e modellazione architettonica interattiva [1, 2, 3].

Nell'ambito dello sviluppo di applicativi in realtà virtuale (VR), l'obiettivo principale è quello di aumentare il livello di immersività dell'utente all'interno dell'ambiente virtuale. Per perseguire questo obiettivo, il miglioramento delle tecniche di audio immersivo è fondamentale e nel corso dell'evoluzione tecnologica sono stati sviluppati diverse metodologie che potessero simulare l'effetto acustico di un ambiente reale [1].

Le tecniche di audio immersivo principalmente utilizzate si basano su algoritmi di Digital Signal Processing (DSP) che simulano l'effetto acustico all'interno di zone delineate [1, 4]. Un altro approccio che si può perseguire è quello basato sulla fisica e sulla geometria dello spazio. Questi metodi si basano proprio sul comportamento reale che l'onda sonora avrebbe nella realtà [1, 2]. Entrambi gli approcci offrono soluzioni efficaci per ricreare ambienti immersivi dal punto di vista acustico, ma hanno anche due comportamenti completamente differenti che verranno approfonditi nei capitoli successivi.

Il seguente elaborato vuole studiare ed applicare la tecnica dell'Image Source Method (ISM) la quale permette la simulazione acustica di un ambiente tenendo conto della sua geometria [1, 5]. L'obiettivo è quello di sviluppare un sistema che permetta l'applicazione del metodo all'interno del motore di gioco Unity 3D. L'applicativo avrà la capacità di estrarre i dati geometrici della scena 3D e sfruttarli per simulare le riflessioni sonore.

Il presente lavoro di tesi si articola in diverse sezioni. Nel Capitolo 1 verrà trattato lo stato dell'arte, presentando le principali tecniche per l'audio immersivo studiate in letteratura, con un'analisi accurata del loro funzionamento e delle relative criticità. Successivamente, nel Capitolo 2 verrà spiegato nel dettaglio l'*Image Source Method* (ISM), descrivendone la logica teorica e le soluzioni tecniche adottate per la sua implementazione all'interno del motore di gioco Unity 3D. Nel Capitolo 3 verrà invece illustrata l'architettura del sistema sviluppato, approfondendo le funzioni e i componenti principali che ne regolano il corretto funzionamento. Infine, il Capitolo 4 verterà sull'interpretazione e il confronto dei test effettuati durante lo studio; in questa fase, le Risposte Impulsive (IR) dell'applicativo sviluppato verranno comparate sia con quelle ottenute tramite il plugin per l'audio immersivo del Meta Quest SDK, sia con quelle misurate nell'ambiente reale che è stato modellato in 3D.

2. AUDIO IMMERSIVO: STATO DELL'ARTE

Il concetto di audio immersivo identifica un modello di riproduzione sonora che ha come obiettivo quello di ricostruire un campo acustico che simuli la provenienza del suono nello spazio tridimensionale.

I fattori fondamentali per riuscire ad ottenere un campo acustico simulato correttamente sono diversi. La simulazione del comportamento dell'apparato uditivo umano, il decadimento del livello sonoro dovuto allo spazio percorso dal suono, l'assorbimento da parte delle superfici riflettenti di una parte dell'onda sonora ed anche il ritardo temporale dovuto al percorso compiuto dal suono [4].

Esistono diverse tecniche per riuscire ad ottenere un audio immersivo in un ambiente virtuale. Nel prossimo paragrafo si andranno ad illustrare i principali metodi utilizzati e studiati in letteratura.

2.1. PRINCIPALI TECNICHE E METODOLOGIE PER L'AUDIO IMMERSIVO

Nel campo dell'audio immersivo le tecniche computazionali per simulare la IR di un ambiente 3D si dividono in tre grandi categorie principali che seguono approcci al problema completamente differenti. I metodi basati su filtri di riverberazione, i metodi basati sull'equazione d'onda e i metodi geometrici. Tutti e tre partono da presupposti differenti per risolvere il problema della propagazione del suono.

Il primo utilizza diversi tipi di filtri acustici per simulare l'effetto riverberante di un ambiente. Un esempio è il riverbero di Schroeder [6]. Questo tipo di riverbero utilizza dei filtri Comb e dei filtri All-Pass per riprodurre il riverbero di un ambiente. Questo, come molti altri tipi di metodi basati sui filtri sono capaci di riprodurre molto fedelmente la IR di un ambiente reale, ma non tengono conto della geometria dell'ambiente e di come esso possa cambiare nel tempo, si basano infatti su zone di riverbero (Reverb Zones) che caratterizzano una determinata area dell'ambiente che presenta determinati parametri impostati che determinano il tipo di ambiente simulato [6]. Questa limitazione è la principale differenza tra questo primo tipo ed altri approcci discussi nel seguito.

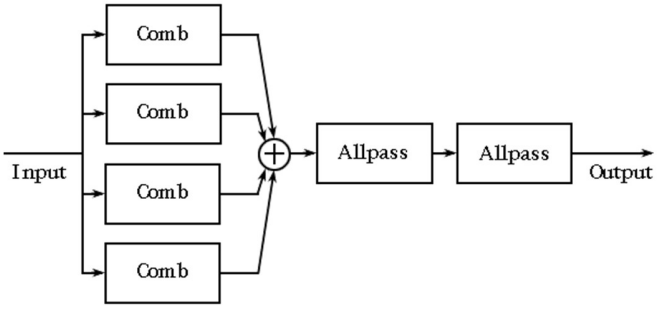

Figura 1 Schema del riverbero di Schroeder

Il secondo approccio si basa sull'analisi del fronte d'onda sonoro e sul calcolo del comportamento di quest'ultimo in relazione all'ambiente circostante. Un esempio è la tecnica del Digital Waveguide Mesh (DWM) [7, 8] che semplifica lo spazio tridimensionale da simulare in un reticolo regolare di nodi interconnessi tra loro. La propagazione sonora avviene proprio tramite questi nodi che simulano lo spostamento del fronte d'onda nello spazio e calcola di volta in volta le variazioni del livello di intensità sonora. In questo caso la simulazione tiene conto dell'ambiente virtuale e simula al meglio l'ambiente 3D anche in caso di eventuali variazioni dell'ambiente. Le limitazioni si trovano nel costo computazionale del sistema. Questo processo è infatti molto oneroso a livello computazionale e rischia di saturare rapidamente la CPU e la RAM anche per stanza di medie dimensioni [7, 8].

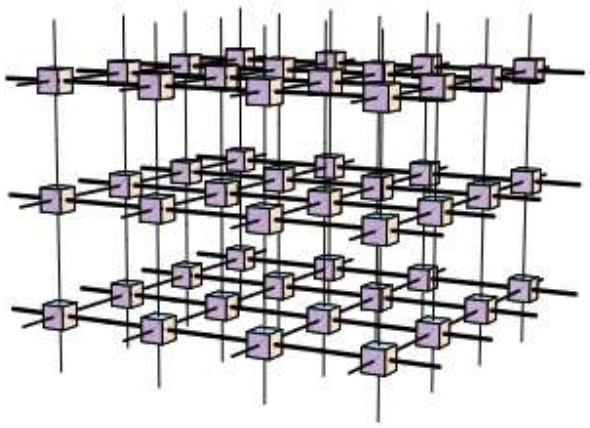

Figura 2 Schema di una DWM

I metodi geometrici, invece, cercano di approssimare il suono ad una serie di raggi o fasci che viaggiano nell'ambiente virtuale. Tra le principali tecniche troviamo il Ray Tracing Acustico, una tecnica che si basa sulla proiezione di numerosi raggi privi di spessore all'interno della scena. Questi raggi possono riflettersi sulle superfici e per ogni riflessione viene calcolata la dispersione di energia dovuta a quest'ultima. Queste riflessioni avvengono finchè non avviene una collisione con l'ascoltatore. La principale limitazione di questo sistema è dovuta al possibile sotto campionamento dello spazio. I raggi monodimensionali potrebbero non intersecare l'ascoltatore, soprattutto se presenta dimensioni ridotte. Questo comporta quindi un aumento significativo del numero di raggi necessari per evitare il problema [1, 2, 5].

Un altro metodo che risolve in parte i problemi del precedente è il Beam Tracing. Questa tecnica sostituisce i raggi monodimensionali con dei volumi geometrici che viaggiano nell'ambiente 3D. In questo caso il problema del sottocampionamento diventa molto meno importante e sposta invece le criticità sotto il punto di vista computazionale. Gestire un elevato numero di volumi geometrici in un ambiente 3D ricco di geometrie potrebbe portare ad un onere computazionale troppo elevato per applicazioni in realtime [2, 9, 10, 11].

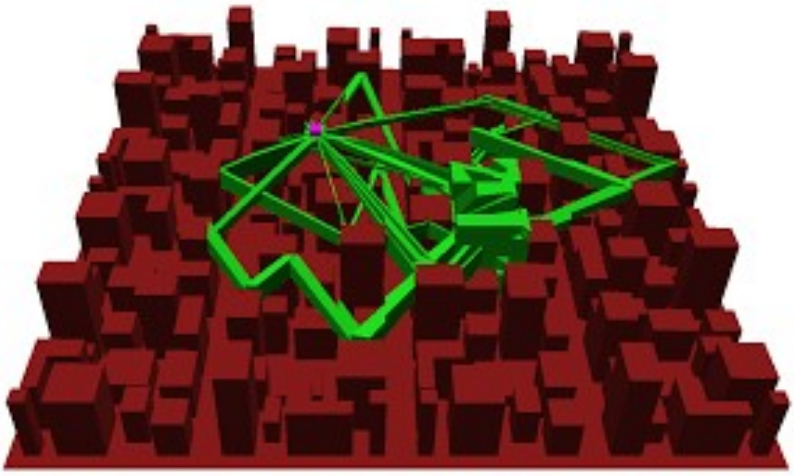

Figura 3 Schema di riflessioni di un sistema Beam Tracing

Un esempio di questo tipo di tecnica è proprio l'ISM, una tecnica che sfrutta una serie di raggi proiettati dalla sorgente sonora, per calcolare la traiettoria e le riflessioni del suono nell'ambiente. Anche questi metodi tengono conto dello spazio virtuale ed il loro costo computazionale è notevolmente ridotto rispetto ai metodi basati sull'analisi del fronte d'onda. Dall'altro canto però i metodi geometrici possono riscontrare dei problemi di ipersemplificazione del sistema o di sotto campionamento dell'ambiente. La descrizione del funzionamento dell'ISM ed i principali limiti tecnici di questo metodo verranno approfonditi nel capitolo successivo [1, 5].

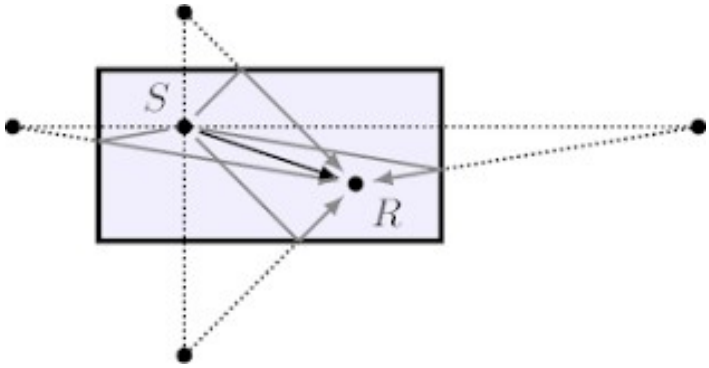

Figura 4 Schema di riflessione dell'ISM

Esistono anche esempi di tecniche ibride che mescolano la semplicità computazionale dei filtri acustici, alla resa dei metodi geometrici. Un esempio è il sistema sviluppato da Motive Studio sul Frostbite Engine per il remake del primo capitolo di Dead Space la famosa serie di giochi survival horror di Electronic Arts [12]. Il sistema sfrutta una serie di raggi proiettati dalla sorgente sonora che devono raggiungere il giocatore senza poter superare gli ostacoli fisici presenti nella scena. Tramite una serie di "gate" invisibili il suono può raggiungere il giocatore mantenendo una direzione di arrivo coerente con l'ambiente 3D e l'effetto riverberante viene poi gestito tramite filtri di riverbero [12].

2.2. OTTIMIZZAZIONE DELLA FEDELTÀ GEOMETRICA

Gli studi sull'applicazione dell'approccio geometrico per l'audio immersivo nella Realtà Virtuale evidenziano come la complessità computazionale della simulazione possa essere ottimizzata agendo sul bilanciamento tra fedeltà fisica e accuratezza percettiva. La modellazione di ogni poligono di un ambiente e la discretizzazione dei coefficienti di assorbimento in bande di frequenza ad altissima risoluzione richiedono risorse computazionali elevate, spesso non necessarie per l'orecchio umano. Ricerche dedicate alla semplificazione dei modelli acustici dimostrano che una decimazione controllata della geometria ambientale riduce linearmente il numero di intersezioni da calcolare, senza alterare in modo significativo i parametri acustici globali della stanza o la percezione spaziale del riverbero da parte dell'utente. Allo stesso modo, la riduzione della risoluzione in frequenza dei coefficienti di assorbimento dei materiali consente di snellire i calcoli necessari per ogni riflessione mantenendo un elevato realismo del riverbero e della spazializzazione [2].

3. DESCRIZIONE ED ANALISI DELL'APPROCCIO PROPOSTO

La scelta dell'approccio è stata compiuta andando a considerare la difficoltà di applicazione ed- il costo computazionale del sistema. L'ISM ha soddisfatto entrambe le esigenze di progetto: il sistema ha bisogno di una gestione dei raggi e delle riflessioni che nell'ambiente Unity 3D è già implementata e ottimizzata, mentre il costo computazionale necessario può essere facilmente controllato andando ad impostare dei valori limite per le riflessioni [1].

3.1. ASTRAZIONE GEOMETRICA DEL SISTEMA

L'Image Source Method basa il suo funzionamento sulla sorgente originale che dà origine ad una serie di raggi che, viaggiando nell'ambiente, hanno la capacità di simulare le riflessioni sonore sulle superfici [1].

La sorgente è quindi il fulcro del sistema. I raggi che questa propaga non sono altro che dei vettori geometrici con la capacità di entrare in collisione con le superfici ambientali. Ogni volta che un raggio entra in collisione con una superficie questo si riflette secondo la normale della superficie e continua il suo cammino. L'insieme dei raggi proiettati dalla sorgente e le loro riflessioni sulle superfici dell'ambiente permettono di ricostruire il percorso che il suono ha intrapreso nella scena seguendo la geometria presente [1].

3.2. MODELLO MATEMATICO E FORMULE PER IL CALCOLO ACUSTICO

Ogni volta che i raggi della sorgente originale o le loro riflessioni entrano in contatto con una superficie il sistema esegue una serie di calcoli per andare a generare una sorgente immagine della sorgente originale che andrà a simulare la riflessione sonora. La riflessione di un suono su una superficie può essere considerata come una nuova sorgente posta in posizione simmetrica rispetto alla sorgente originale, dove la superficie colpita può essere considerata l'asse di simmetria [1]. Per andare a calcolare la posizione S' della sorgente immagine viene impiegata la seguente formula:

$$S' = S + 2d\vec{N},$$

dove S è il vettore posizione della sorgente che ha originato la riflessione (la sorgente originale per le prime riflessioni e le sorgenti immagine per le successive), d è la distanza tra la sorgente che ha originato la riflessione e il punto di collisione con la superfice riflettente e $\vec{N}$ il vettore normale della superfice colpita [1].
La sorgente immagine deve riprodurre lo stesso suono della sorgente originale, ma con delle riduzioni al livello di intensità sonora. La prima riduzione è dovuta al decadimento acustico in aria. La decrescita dell'intensità

sonora segue una scala logaritmica rispetto alla distanza totale percorsa dal raggio. La formula utilizzata nel sistema sviluppato per simulare il decadimento acustico in aria è la seguente:

$$A(d) = \left(\frac{1}{2}\right)^{\gamma * d_{tot}},$$

dove d_{tot} è la distanza totale percorsa dal raggio e γ è il coefficiente di decrescita che permette il controllo dell'intensità del decadimento [1].

Un'altra modifica che deve avvenire al livello di intensità sonora della sorgente immagine è legata all'assorbimento acustico della superfice riflettente. Ogni superfice nell'ambiente ha un determinato coefficiente di assorbimento α che determina quanta energia sonora verrà assorbita al momento della riflessione [1]. Il coefficiente è un valore adimensionale che può assumere valori tra uno e zero compresi. Noto α, la formula per calcolare il coefficiente di riflessione β è la seguente:

$$\beta = \sqrt[2]{1 - \alpha}.$$

Il coefficiente β viene poi moltiplicato progressivamente al livello di intensità sonora per ogni riflessione [1].
Oltre al livello di intensità sonora, il sistema deve calcolare il ritardo temporale dovuto al tempo impiegato dal suono per percorrere la traiettoria del raggio, anche chiamato time of flight. Questo ritardo viene calcolato andando ad utilizzare la velocità di propagazione del suono in aria $v_s = 343\frac{m}{s}$ e la distanza totale percorsa dal raggio al momento della riflessione d_{tot} seguendo la relazione lineare:

$$\Delta t = \frac{d_{tot}}{v_s}$$

3.3. VINCOLI ALGORITMICI

Per riuscire a limitare il consumo di risorse computazionali richieste dall'ISM, devono essere imposti dei vincoli all'algoritmo di propagazione sonora. Il vincolo principale, presentato in letteratura, è posto sull'ordine massimo delle riflessioni. L'ordine di una riflessione indica il numero di volte che il raggio generatore ha già riflettuto su una superfice: le prime riflessioni scaturite dalla sorgente originale sono considerate di ordine zero, mentre le riflessioni immediatamente successive avranno un ordine pari ad uno. Questa proprietà della riflessione può essere controllata e vincolata nell'algoritmo per evitare che venga generato un numero eccessivamente elevato di riflessioni. L'algoritmo presenta un vincolo che blocca la generazione di riflessioni una volta che queste hanno raggiunto un determinato ordine [1].

Un altro vincolo progettato per questo sistema è legato al numero massimo di riflessioni iniziali che una singola sorgente può generare. Anche questo vincolo è legato alla gestione delle necessità computazionali del sistema ed è determinante in ambienti con una geometria molto complessa.

4. PROGETTAZIONE E SVILUPPO DEL SISTEMA

4.1. SELEZIONE DELL'AMBIENTE DI SVILUPPO

L'ambiente di sviluppo per il sistema proposto è stato scelto secondo le necessità progettuali delineate. La prima è quella di sviluppare una serie di codici che permettano l'audio immersivo in un ambiente virtuale modellato in 3D. I due principali ambienti di sviluppo che permettono di raggiungere questo scopo sono: Unity ed Unreal Engine [13, 14]. La seconda necessità progettuale è stata quella di avere un sistema che non avesse un elevato costo computazionale, Unity sotto questo aspetto si è rivelato il migliore tra i due ambienti, in quanto offre un motore di gioco più leggero rispetto ad Unreal Engine.

4.2. GAME OBJECTS E SCENA

I Game Objects principali utilizzati da questo sistema sono tre: la sorgente sonora, l'ascoltatore e le superfici riflettenti. La sorgente sonora contiene uno degli script principali sviluppati e con il componente $Sound\ Object$ nativo di Unity. Questo game object è necessario per la riproduzione del suono e l'innesco delle riflessioni sonore. L'ascoltatore tramite il componente di Unity $Audio\ Listener$ può percepire i suoni riprodotti nella scena e, assieme ad un altro codice personalizzato, permette anche la registrazione dei segnali percepiti dal componente. Le superfici riflettenti permettono di ricreare la geometria dell'ambiente 3D tramite dei componenti $Collider$ e di calcolare le proprietà delle sorgenti immagine grazie ad uno script. La scena presenta anche un quarto Game Object che viene utilizzato per controllare la riproduzione della sorgente sonora e per far avviare la registrazione. Questo oggetto non è necessario per il corretto funzionamento del sistema, ma utile soltanto a scopo di test.

Per poter permettere la comparazione delle IR simulate a delle IR reali, è stato modellato in 3D un ambiente reale presente all'interno del Laboratorio Audio DSP dell'Università Politecnica delle Marche. La stanza è stata ricreata completamente utilizzando il software di modellazione 3D CAD Rhinoceros [15] andando a misurare le dimensioni della stanza e degli arredi principali tramite un metro a nastro ed un metro laser.

Figura 5 Stanza usata per gli esperimenti e modellata

La stanza non è stata modellata nella sua interezza, ma sono stati scelti gli arredi principali con superfici riflettenti abbastanza grandi da impattare sulla resa acustica dell'ambiente, come i tavoli e la lavagna Lim.

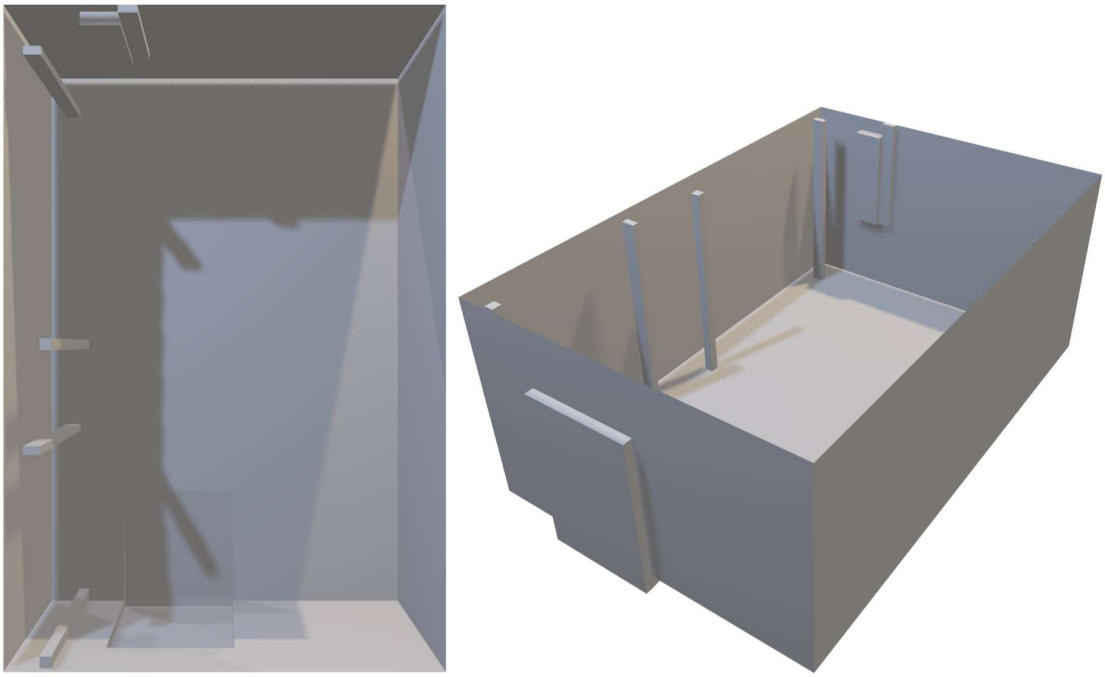

Figura 6 Modello 3D della stanza

Una volta importato il modello all'interno della scena in Unity le diverse superfici sono state colorate e rese fisiche tramite il componente collider.

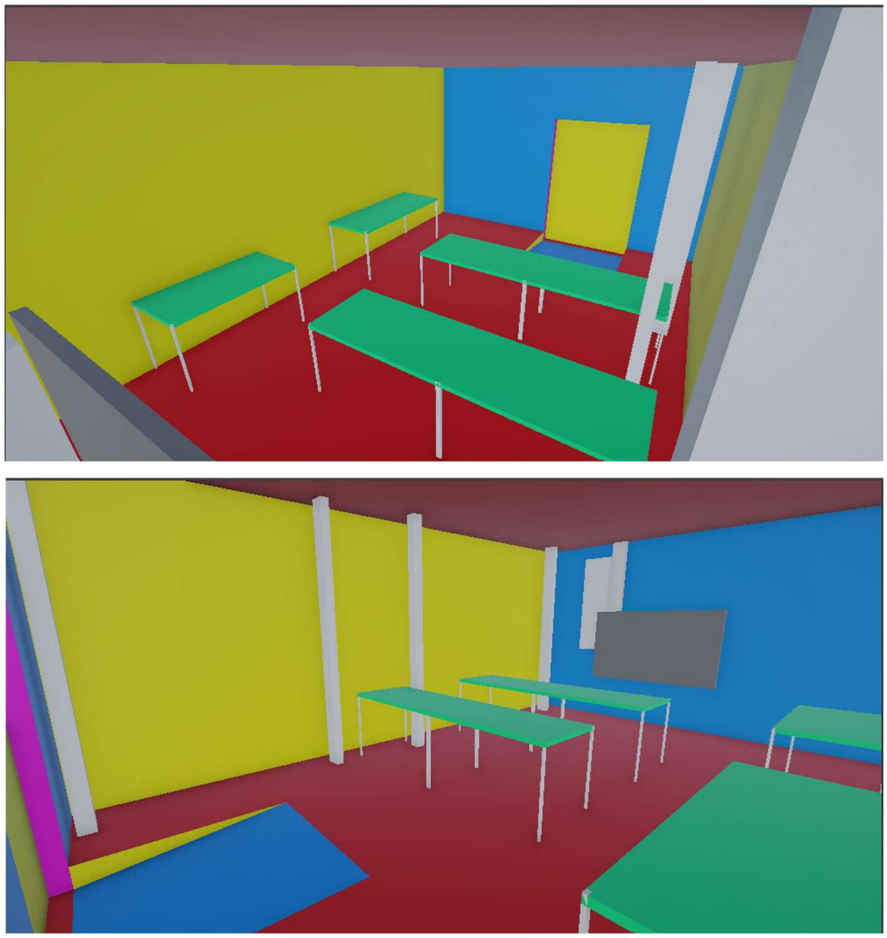

Figura 7 Stanza importata in Unity

4.3. LOGICA E FUNZIONAMENTO DEL SISTEMA

Il sistema progettato presenta due script in linguaggio C# principali che guidano il funzionamento dell'ISM:

- $Sound\_Object$: dedicato alla proiezione dei raggi necessari per simulare le riflessioni
- $Reverb\_Object$: dedicato al calcolo dei parametri necessari alla generazione delle sorgenti immagine

Nel progetto sono presenti anche altri due script di supporto che gestiscono la riproduzione ripetuta del suono ($Start\_sound\_manager$) e la registrazione di ciò che viene percepito dall'ascoltatore ($Registratore$).

4.3.1. SOUND_OBJECT

Il codice dedicato alla proiezione dei raggi ed alla schedulazione della riproduzione del suono da parte della sorgente sonora è composto da sette funzioni, di cui $ScanAmbient$, $ReflectSound$ e $RayReflect$ sono le principali.

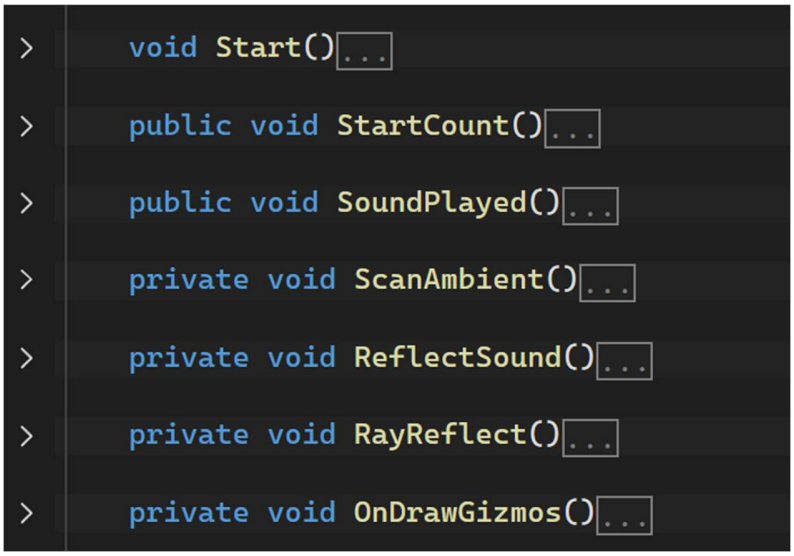

Figura 8 Principali funzioni dello script Sound_Object

Il metodo $ScanAmbient$ implementa l'algoritmo di generazione dei raggi di riflessione della sorgente sonora originale. La generazione dei raggi nel volume del cono scelto dall'utente avviene attraverso una struttura a due cicli for annidati, che lavorano in coordinate polari sferiche proiettate rispetto all'asse di orientamento della sorgente ($transform.forward$). Il ciclo esterno gestisce la progressione radiale dell'apertura del cono. Campiona lo spazio espandendosi dal centro del cono verso il suo limite esterno tramite un numero discreto di passaggi definiti dalla variabile $segments$. Il ciclo interno, per ciascun livello di apertura radiale impostato dal ciclo esterno, esegue una rotazione completa di 360° attorno all'asse principale. L'effetto di questa struttura annidata sui raggi generati è la proiezione di una serie di anelli concentrici di vettori. Ad ogni ripetizione del ciclo esterno che aumenta l'apertura, il ciclo interno traccia una circonferenza di raggi sempre più ampia.
L'equazione:

$$\text{currentAngle} = \left(\frac{angle}{segments}\right) * \text{i},$$

discretizza lo spazio angolare del cono dividendo l'angolo totale di apertura ($angle$) per il numero di segmenti totali ($segments$) ottenendo la distanza tra le circonferenze concentriche. Moltiplicando questo valore per l'indice corrente del ciclo i, la formula calcola l'angolo di deviazione laterale rispetto all'asse centrale per l'anello corrente. Visivamente, questa formula determina quanto il raggio debba "allargarsi" rispetto alla retta frontale della sorgente. Il calcolo della rotazione del singolo raggio che compone le circonferenze del cono avviene tramite la formula:

$$\text{coneRotation} = \text{Quaternion.AngleAxis}(\text{j}, \text{direction}) * \text{Quaternion.AngleAxis}(\text{currentAngle}, \text{transform.up}),$$

$Quaternion.AngleAxis(currentAngle, transform.up)$ applica la deviazione calcolata in precedenza, inclinando il raggio lateralmente rispetto all'asse verticale della sorgente. Moltiplicando questo risultato per $Quaternion.AngleAxis(j, direction)$, il vettore viene fatto ruotare attorno all'asse frontale della sorgente di un angolo pari a j gradi.

```
Vector3 origin = transform.position;
Vector3 direction = transform.forward;

for (int i = 1; i <= segments; i++)
{
    float currentAngle = (angle / segments) * i;

    for (int j = 0; j < 360; j++)
    {
        Quaternion coneRotation = Quaternion.AngleAxis(j, direction) * Quaternion.AngleAxis(currentAngle, transform.up);
        Vector3 rayDir = transform.rotation * coneRotation * direction;

        Ray ray = new Ray(origin, rayDir);
```

Figura 9 Calcolo del raggio di scansione nella funzione ScanAmbient

Ad ogni proiezione del raggio, la funzione esegue un controllo sul GameObject che è stato colpito dal raggio. Se questo ha assegnato il tag “Reflective” l’algoritmo esegue un controllo sulla presenza dell’oggetto nella lista $scanResults$, contenente tutte le collisioni avvenute dall’inizio della funzione, e se non è stata trovata alcuna referenza all’oggetto colpito la collisione viene salvata all’interno di $scanResults$ per permettere il calcolo della sorgente immagine.

```
if (Physics.Raycast(ray, out hit))
{
    if (hit.collider.gameObject.tag == "Reflective")
    {
        foreach (CollisionInfo res in scanResults)
        {
            if (res.obj == hit.collider.gameObject)
            {
                exist = true;
                break;
            }
        }

        if (!exist)
        {
            CollisionInfo newInfo = new CollisionInfo();
            newInfo.obj = hit.collider.gameObject;
            newInfo.normal = hit.normal;
            newInfo.hitPoint = hit.point;
            scanResults.Add(newInfo);

            Debug.DrawLine(transform.position, hit.point, Color.red, 100);
        }
        else
            exist = false;
    }
}
```

Figura 10 Codice di controllo della collisione del raggio

Terminata l’esecuzione della scannerizzazione, la funzione $ReflectSound$ si occupa di inviare alle pareti colpite dai raggi le informazioni utili alla generazione delle sorgenti immagine (Posizione della sorgente, normale del

punto di collisione, ordine della riflessione, clip audio da riflettere, raggio riflesso, punto di riflessione, distanza totale percorsa, volume della sorgente) che verranno poi elaborate nello script $Reverb\_Object$.

```
foreach(CollisionInfo res in scanResults)
{
    res.obj.GetComponent<Reverb_Object>().CreateReflectedSound(reflectPos, res.normal, order
}
```

Figura 11 Corpo della funzione ReflectSound che si occupa, per ogni elemento in scanResult, di richiamare la funzione di generazione della sorgente immagine

Nel caso in cui la sorgente che sta causando la riflessione sia una sorgente immagine, la funzione di scannerizzazione descritta in precedenza viene sostituita dalla funzione $RayReflect$ che gestisce la proiezione del singolo raggio riflesso all'interno della scena. La riflessione del raggio avviene tramite la funzione nativa di Unity $Vector3.Reflect$ rispetto alla normale della superfice colpita.

```
Ray ray = new Ray(reflectPos, reflectDir);
RaycastHit hit;
if (Physics.Raycast(ray, out hit))
{
    if (hit.collider.gameObject.tag == "Reflective")
    {
        CollisionInfo newInfo = new CollisionInfo();
        newInfo.obj = hit.collider.gameObject;
        newInfo.normal = hit.normal;
        newInfo.hitPoint = hit.point;
        scanResults.Add(newInfo);

        Debug.DrawLine(reflectPos, hit.point, Color.blue, 100);

        ReflectSound();
    }
}
```

Figura 12 Corpo della funzione RayReflect

Le altre funzioni presenti nello script permettono la temporizzazione della riproduzione del suono, tramite l'ausilio del buffer audio, la distruzione automatica della sorgente immagine dopo la riproduzione e la visualizzazione nella scena del cono sonoro della sorgente, per facilitare l'impostazione della sorgente sonora.

L'angolo di ampiezza massima e il numero di circonferenze concentriche possono essere personalizzati dall'utente tramite l'inspector di Unity. È esposta anche la variabile contenente la clip audio da riprodurre.

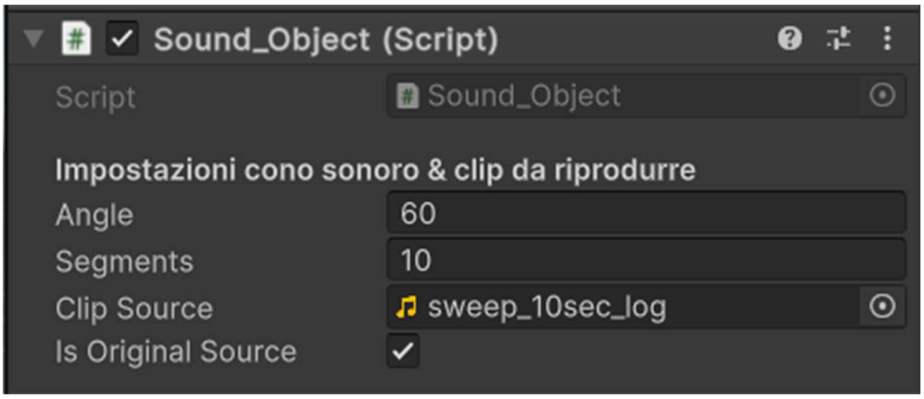

Figura 13 Visualizzazione dall'inspector dello script Sound_Object

4.3.2. REVERB_OBJECT

Il codice assegnato a tutte le superfici riflettenti permette il calcolo dei parametri necessari per la corretta generazione delle sorgenti immagini che sono stati illustrati nel capitolo precedente.

```
2 riferimenti | Gpt
float ReflectedVolume(float lastVolume, float distanzaR, float beta, float R0 = 0.5f)...

1 riferimento | Gpt
public void CreateReflectedSound(Vector3 soundPos, Vector3 normal, int order,
    AudioClip clip, Vector3 reflectDir, Vector3 hitPoint, float distanceTravel,
    float lastVolume)...
```

Figura 14 Principali funzioni dello script Reverb_Object

Il decadimento del livello di intensità sonora dovuto al viaggio e alla riflessione viene gestito dalla funzione $ReflectedVolume$ tramite la formula:

$$volume = lastVolume * beta * Mathf.Pow(0.5f, distanzaR * decrCoeff),$$

dove $lastVolume$ contiene l'intensità sonora prima del decremento, $beta$ è il coefficiente di riflessione della superfice, $distanzaR$ è la distanza percorsa dal suono e $decrCoeff$ è il coefficiente di decrescita del volume.

```
float volume = lastVolume * beta * Mathf.Pow(0.5f, distanzaR*StartSoundManager.istance.decrCoeff);

return Mathf.Clamp01(volume);
```

Figura 15 Corpo della funzione ReflectedVolume

All'interno del metodo principale questo calcolo viene eseguito due volte: prima per ricavare il volume residuo immediatamente dopo l'impatto sulla parete ($volumeR$), e successivamente per calcolare l'attenuazione subita dal suono nel viaggiare dal punto di impatto all'orecchio dell'utente ($newVolumeP$), utilizzando quest'ultimo valore per il componente *AudioSource* della sorgente immagine generata.

```
float dist = Vector3.Distance(soundPos, hitPoint);
distanceTravel += dist;

Vector3 refSoundPos = soundPos - 2 * Vector3.Distance(soundPos, transform.position) * normal;
float distToPlayer = Vector3.Distance(hitPoint, player.transform.position);

float volumeR = ReflectedVolume(lastVolume, dist, reflection_Coefficient_B);
float newVolumeP = ReflectedVolume(volumeR, distToPlayer, 1);
```

Figura 16 Sezione della funzione CreateReflectedSound dedicata al calcolo del volume della sorgente immagine

Dopo aver effettuato i dovuti calcoli e ottenuto tutti i parametri necessari alla generazione della sorgente immagine, la funzione istanzia un nuovo oggetto sorgente e gli assegna tutti i parametri calcolati. L'oggetto, che contiene lo script $Sound\_Object$, tramite la funzione nativa di Unity $Start$ andrà ad eseguire tutte le funzioni necessarie a far ricominciare il ciclo di riproduzione e riflessione.

I parametri personalizzabili principali ed esposti nell'inspector di Unity sono Absorption_Coefficient_A e Reflection_Coefficient_B che rappresentano rispettivamente i valori α e β descritti in precedenza. Questi coefficienti se impostati al valore -1 vengono considerati nulli dal sistema che cercherà di calcolarli seguendo la formula:

$$\beta = \sqrt[2]{1-\alpha}.$$

Nel caso in cui entrambi i valori vengano impostati a -1, il sistema interpreterà la superfice come completamente riflettente, andando ad impostare $\alpha = 0$ e $\beta = 1$.

Figura 17 Visualizzazione dall'inspector dello script Reverb_Object

5. IMPLEMENTAZIONE DEL PLUGIN META

Per avere un altro riferimento di IR simulata con un sistema già presente in commercio, è stato installato ed impiegato un plugin per l'audio immersivo del Meta SDK [16]. Questo plugin ha permesso di confrontare l'efficacia del sistema sviluppato rispetto ad un sistema che viene già largamente utilizzato in ambiente Unity.

Le informazioni note sul funzionamento del plugin si riducono all'utilizzo di raycast per il calcolo delle riflessioni geometriche, ma come visto nel secondo capitolo ci sono diverse tecniche che sfruttano questi oggetti virtuali per il calcolo.

5.1. FUNZIONAMENTO DEL PLUGIN

Il plugin presenta quattro script principali che regolano il comportamento della riproduzione sonora. Il primo che si occupa dell'impostazione della sorgente sonora come sorgente di audio immersivo, il secondo che permette di modificare le proprietà acustiche delle superfici riflettenti, il terzo che gestisce la generazione della geometria delle superfici riflettenti in scena ed il quarto che genera una acoustic map che verrà adoperata dal sistema per riprodurre l'effetto acustico desiderato.

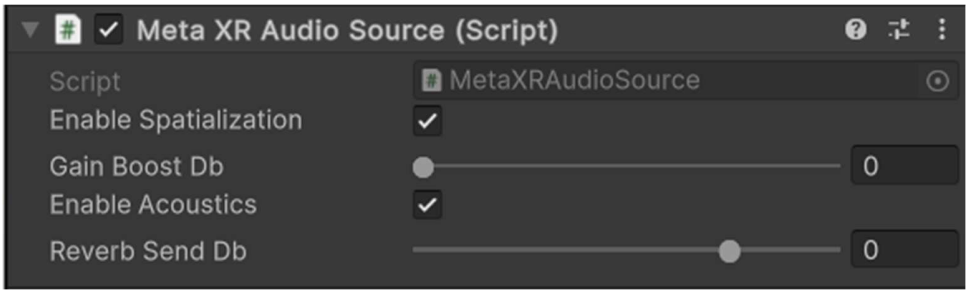

Figura 18 Visualizzazione dall'inspector dello script per la sorgente audio del plugin di Meta

5.2. IMPOSTAZIONE DEL PLUGIN

Per permettere il corretto funzionamento del plugin sono stati inseriti nella scena due nuovi Game Objects e gli sono stati assegnati rispettivamente lo script per la generazione della geometria ambientale e quello per la creazione della acoustic map.

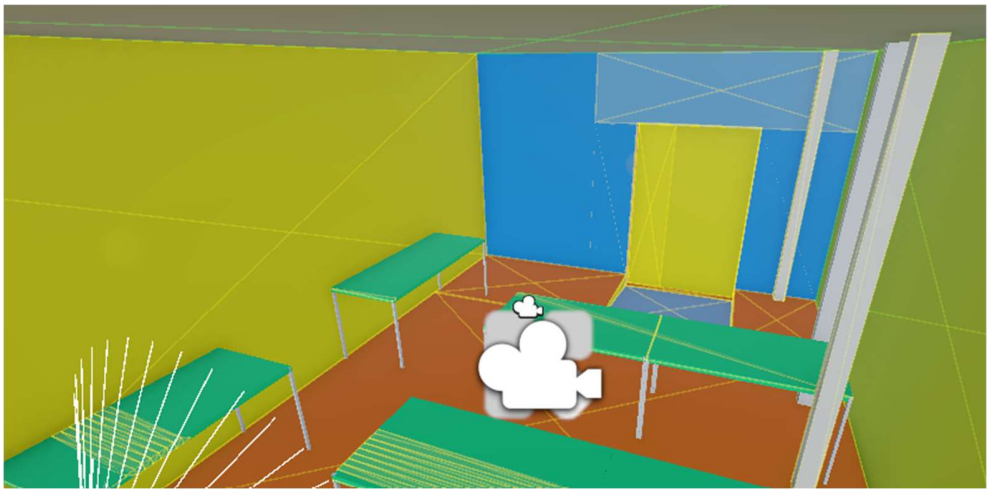

Figura 19 Visualizzazione delle geometrie mappate dal plugin

È stata anche inserita nella scena una sorgente sonora con lo script necessario per interagire con l'ambiente acustico e per avere più controllo sui parametri della sorgente è stato utilizzato lo script sperimentale: $MetaXRAudioSourceExperimentalFeatures$ che permette di modificare parametri importanti per la riverberazione del suono [12].

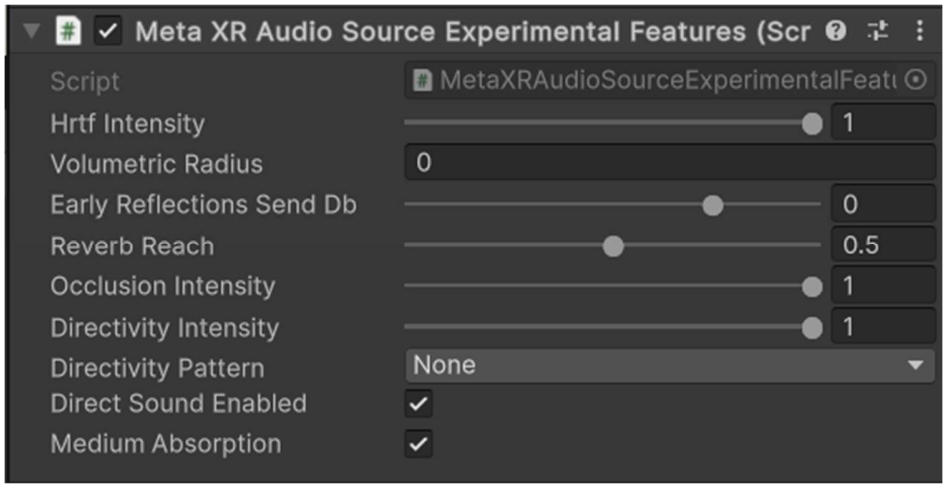

Figura 20 Visualizzazione dall'inspector dello script sperimentale per la sorgente audio del plugin di Meta

A tutte le superfici presenti in scena è stato anche assegnato lo script $Meta\ XR\ Acoustic\ Material$ per permettere la personalizzazione dei parametri di riflessione degli oggetti.

5.2.1. PARAMETRI DELLE SUPERFICI

Per modificare il coefficiente di riflessione delle superfici, il plugin utilizza un sistema basato sui materiali e sulle proprietà fonoassorbenti che questi possiedono. I materiali permettono di modificare tre parametri principali:

Absorption, Transmission e Scattering. Questi tre parametri sono personalizzabili dall'utente andando a inserire dei punti all'interno di un grafico Intensità/Frequenza e modificandone le coordinate [12].

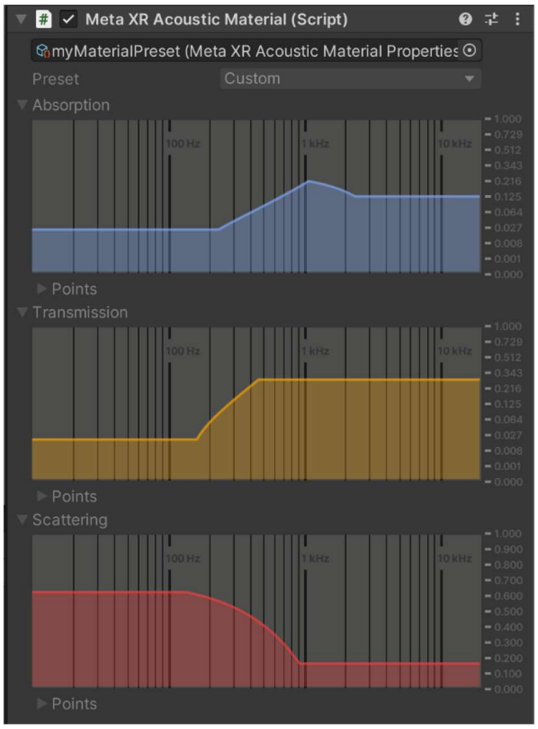

Figura 21 Visualizzazione dall'inspector dello script per la personalizzazione dei materiali delle superfici riflettenti

Questo tipo di personalizzazione è estremamente dettagliata e permette di ricreare molto fedelmente i materiali esistenti nella realtà. Per questa ricerca sono stati utilizzati alcuni dei materiali messi a disposizione dal plugin.

Figura 22 Tipologie di materiali utilizzati per i test

6. TEST SPERIMENTALI

I test effettuati sulle IR del sistema proposto sono stati confrontati con le IR reali e le IR del plugin di Meta andando a delineare undici posizioni per ricevitore e sorgente all'interno della stanza. Le posizioni sono state scelte cercando di ottenere la maggior parte delle impostazioni possibili all'interno della stanza (Sorgente e ricevitori vicine, lontane, disallineate, allineate) e le prime sei sono state fatte andando soltanto a cambiare la posizione del ricevitore, le successive tre sono state prese muovendo soltanto la sorgente sonora e le ultime due presentano sorgente e ricevitore nelle stesse posizioni, ma la sorgente sonora è stata ruotata rispettivamente di 180° e 90°.

1	X (m)	Y (m)
Mic	1.35	2.01
Sorg	1.37	1
2		
Mic	1.35	3.52
Sorg	1.37	1
3		
Mic	2.33	3.1
Sorg	1.37	1
4		
Mic	3.28	3.08
Sorg	1.37	1
5		
Mic	4.83	4.87
Sorg	1.37	1
6		
Mic	1.56	4.83
Sorg	1.37	1
7		
Mic	1.56	4.83
Sorg	2.79	1.55
8		
Mic	1.56	4.83
Sorg	2.31	3.2
9		
Mic	1.56	4.83
Sorg	1.49	3.37
10		
Mic	1.56	4.83
Sorg	1.22	1.43
11		
Mic	1.56	4.83
Sorg	1.22	1.43

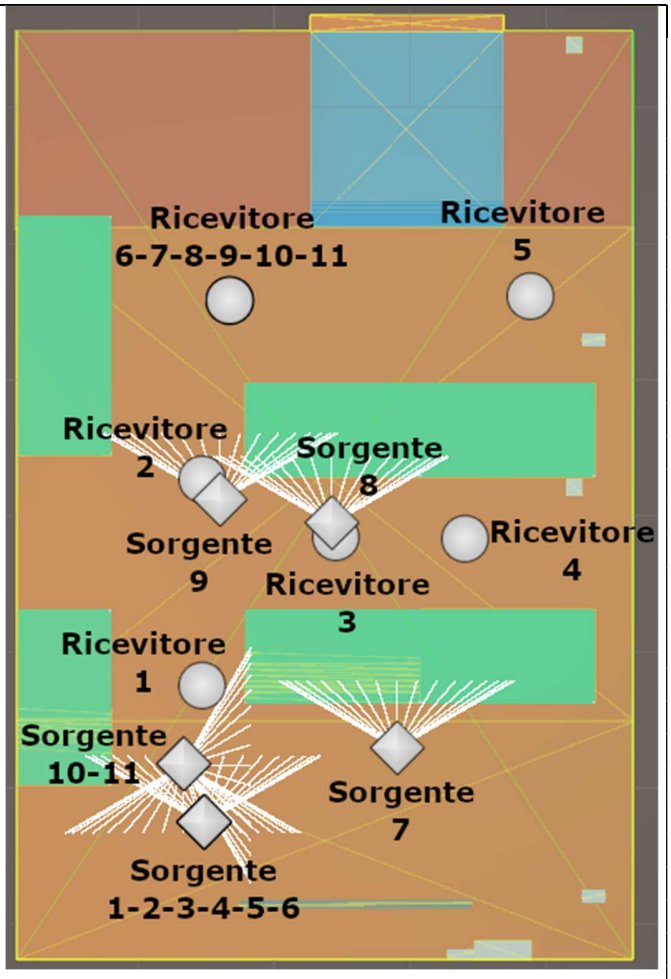

Figura 23 Mappatura delle undici posizioni di registrazione nella stanza

Di queste undici posizioni i test si sono concentrati principalmente sulla posizione 1 e per un confronto finale sulle posizioni 2 e 3. Questo perché i test preliminari effettuati su tutte e undici le configurazioni hanno mostrato una grande discrepanza delle IR simulate e reali. Per questo, concentrarsi sul miglioramento dell'IR della singola posizione, è stato ritenuto più utile ai fini dello studio.

Si sottolinea che, per evitare problemi di disallineamento delle IR, tutti i grafici mostrati sono:

- Normalizzati rispetto all'intensità
- Allineati rispetto al picco principale che è posto ad ascissa zero

6.1. PARAMETRI E IMPOSTAZIONI PER I TEST

I test sono stati gestiti tramite un Manager in grado di gestire la riproduzione dell'impulso sonoro della sorgente e di far avviare la registrazione della risposta. Lo script esegue un ciclo che ripete la riproduzione del suono con un tempo personalizzabile. Tramite l'inspector è data anche la possibilità di modificare l'ordine massimo delle riflessioni che può essere raggiunto dal sistema, il numero massimo di riflessioni che possono scaturire da una singola sorgente, il coefficiente di decrescita dell'intensità sonora e la velocità con cui si propaga il suono nella stanza.

Figura 24 Visualizzazione dall'inspector dello script StartSoundManager con i parametri utilizzati per i test

Lo script $Registratore$ fornisce la possibilità all'utente di scegliere se ottenere un segnale Mono o Stereo tramite una variabile esposta nell'inspector.

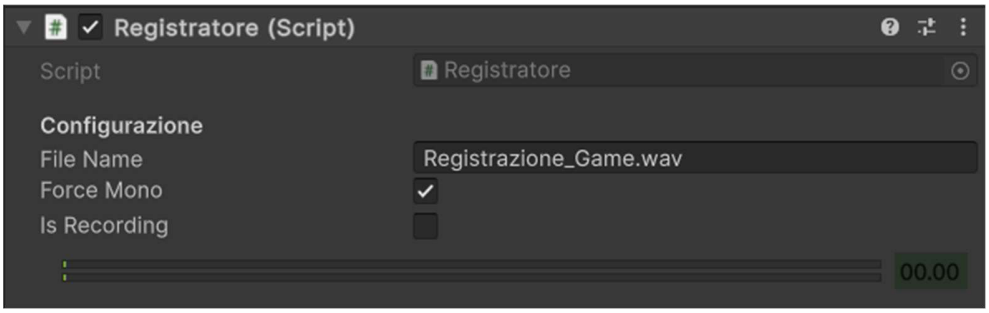

Figura 25 Visualizzazione dall'inspector dello script Registratore con il parametro Force Mono impostato come nei test

6.2. TEST PRELIMINARI SULLA PRIMA POSIZIONE

Il confronto delle IR simulate (del sistema proposto e del plugin Meta) con la IR misurata nell'ambiente reale è visibile in figura 26

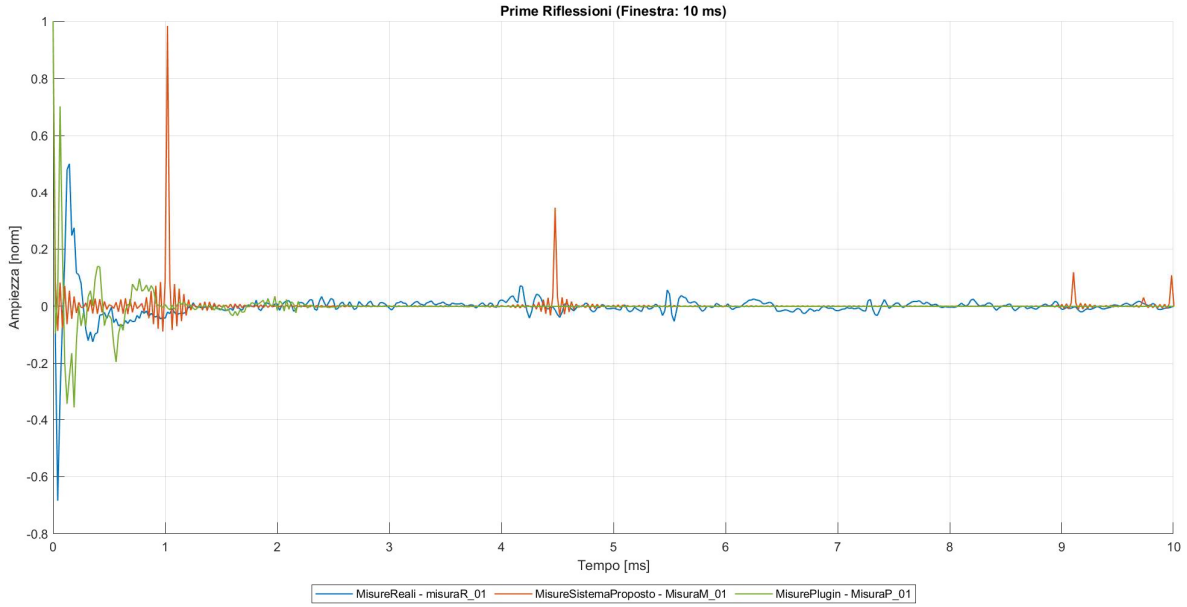

Figura 26 Confronto delle IR simulate con quella reale nella posizione di registrazione 1

Nel grafico di figura 26 si può notare che il plugin (verde chiaro) ed il sistema proposto (arancio) hanno due comportamenti molto differenti tra di loro. Il primo sembra concentrare i picchi nei primi istanti di tempo (0-2ms). Il sistema proposto invece, ha un comportamento che tenta di simulare non solo le prime riflessioni dell'ambiente, ma anche quelle più tardive (9-10ms). Questa differenza nel comportamento è sicuramente dovuta alla differenza di algoritmo adottato dal plugin rispetto al sistema proposto.

6.3. MIGLIORAMENTI E TEST SULLA PRIMA POSIZIONE

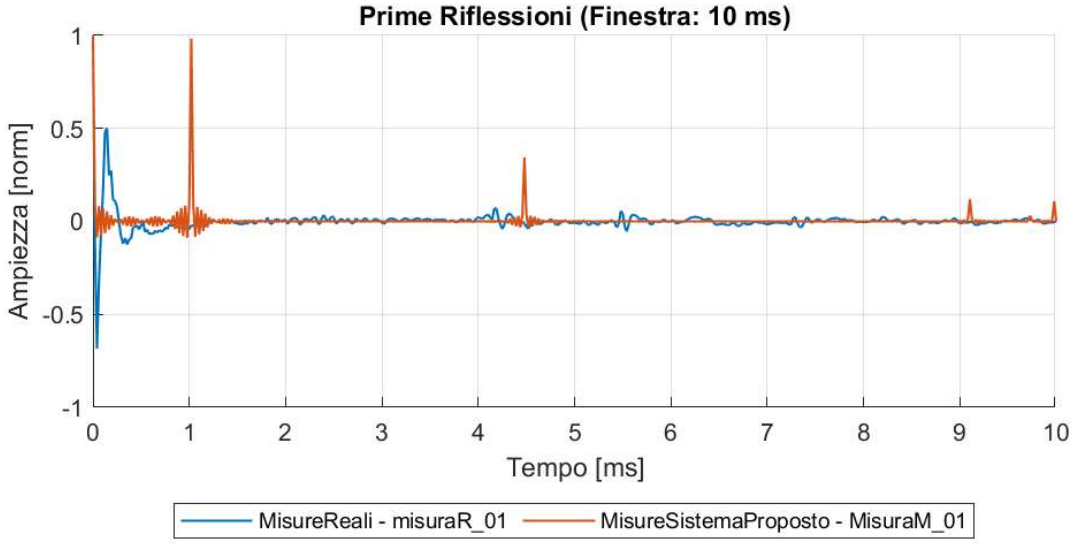

Figura 27 Confronto IR del sistema proposto e reale nella posizione di registrazione 1

Una volta analizzate le IR e trovate le discrepanze con la realtà, i test si sono concentrati sul rendere la IR del sistema proposto più vicina alla realtà. Per raggiungere questo obiettivo le modifiche al sistema si sono concentrate su due punti principali:

- Modifica dei coefficienti di calcolo
- Modifica della geometria in scena

6.3.1. MODIFICHE AI COEFFICENTI

I due coefficienti principali che nel corso dei test si sono rivelati utili all'aumento della similarità dei grafici simulati con quelli reali sono i Coefficienti di riflessione e assorbimento α e β e il Coefficiente di decrescita γ. Entrambi legati al calcolo del livello di intensità sonora delle sorgenti immagine. I coefficienti α e β propri delle superfici riflettenti sono stati scelti seguendo i valori tabellati dei coefficienti fonoassorbenti dei materiali edili.

Materiale	Coefficiente di assorbimento alla frequenza di Hz					
	125	250	500	1000	2000	4000
Elementi costruttivi						
Parete in muratura:						
– con intonaco di gesso	0,01	0,01	0,02	0,03	0,04	0,05
– con intonaco civile	0,03	0,03	0,04	0,04	0,04	0,04
Marmo lucidato o piastrelle	0,01	0,01	0,01	0,02	0,02	0,02
Pavimento in legno (parquet su sottofondo cementizio)	0,04	0,04	0,07	0,07	0,07	0,07
Pavimento in gomma	0,04	0,04	0,06	0,06	0,08	0,08
Vetrata (grosso spessore)	0,18	0,06	0,04	0,03	0,02	0,02
Finestra chiusa	0,35	0,25	0,18	0,12	0,07	0,04
Soffitto in cemento	0,01	0,01	0,02	0,02	0,02	0,02
Soffitto in pannelli di legno	0,28	0,22	0,17	0,09	0,10	0,11
Elementi di rivestimento e di arredo						
Moquette	0,05	0,10	0,25	0,40	0,60	0,70
Tendaggio leggero non drappeggiato	0,03	0,05	0,10	0,15	0,25	0,30
Tendaggio pesante drappeggiato	0,50	0,50	0,70	0,90	0,90	0,90
Lana di roccia, spessore 5 cm	0,38	0,54	0,65	0,76	0,78	0,85
Lana di vetro ricoperta di lamierino metallico forato per il 15% dell'area	0,50	0,75	0,75	0,85	0,75	0,70
Feltro soffice, spessore 5 cm	0,25	0,35	0,60	0,85	0,90	0,90
Pannello di fibre di legno, incollato	0,15	0,25	0,40	0,50	0,50	0,40
Pannello di fibre minerali, incollato	0,15	0,30	0,45	0,50	0,60	0,55

Figura 28 Tabella dei coefficienti di riflessione dei materiali edili e di arredo

Nella realtà le proprietà fonoassorbenti dei materiali variano in base alla frequenza dell'onda sonora. Questo comportamento nel sistema proposto non è presente e sono stati quindi utilizzati dei valori mediani che possano essere comparabili al comportamento reale.

Il coefficiente di decrescita γ è stato modificato andando ad osservare il grafico di decadimento dell'equazione utilizzata nell'algoritmo.

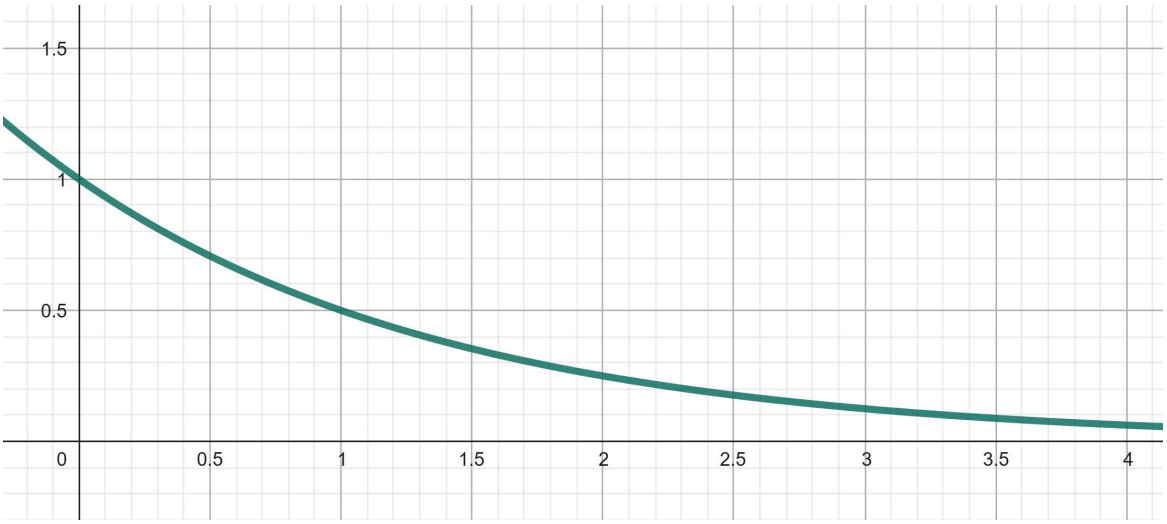

Figura 29 Grafico della curva di decrescita del livello di intensità sonora rispetto allo spazio percorso

Il valore del coefficiente è stato impostato a uno per ottenere un dimezzamento del livello di intensità al raddoppio della distanza percorsa.

Il grafico che risulta delle modifiche effettuate ai coefficienti α, β e γ è il seguente:

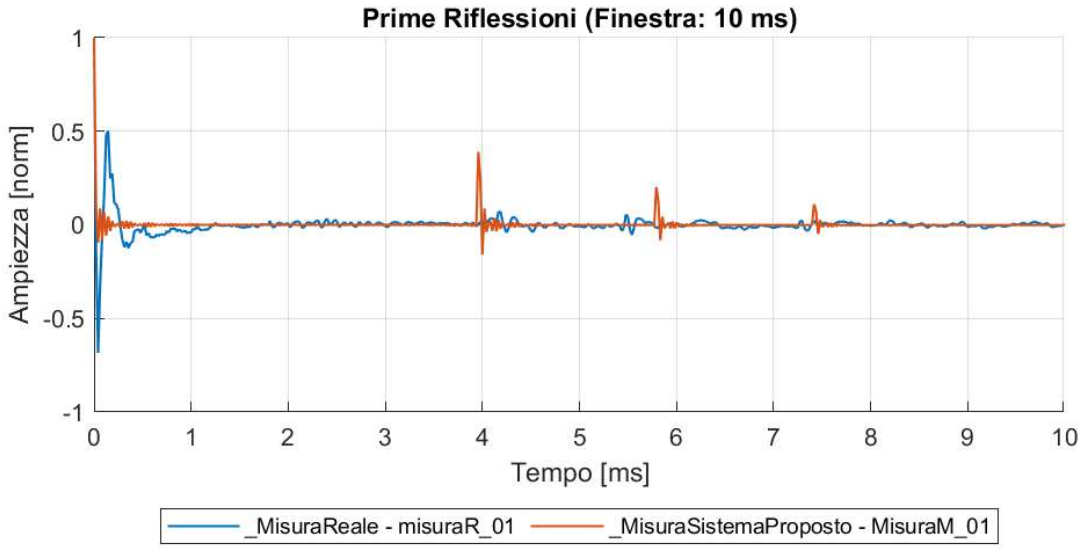

Figura 30 Confronto IR del sistema proposto e reale nella posizione di registrazione 1 con modifiche ai coefficienti di riflessione e di decrescita

In figura 30 si nota un netto miglioramento della IR simulata del sistema proposto. I picchi simulati sembrano avere un'ampiezza più simile a quella dei picchi reali. Questo miglioramento ha portato alla luce un secondo problema del sistema proposto. Il numero dei picchi simulati è minore rispetto al numero dei picchi reali. La causa è stata ricondotta alla principale limitazione del sistema sviluppato.

6.3.2. MODIFICHE ALLA GEOMETRIA

Come detto in precedenza quando la sorgente sonora innesca le riflessioni sulle superfici presenti in scena controlla se la superfice ha già innescato un'altra riflessione e in caso positivo non innesca la funzione di calcolo della sorgente immagine. Questo blocco della riflessione limita notevolmente il numero di riflessioni totali a parità di ordine e potrebbe non permettere alcune riflessioni importanti per la simulazione della IR. Questa limitazione è sì imputabile alla semplificazione del sistema rispetto al comportamento reale, ma può essere aggirata modificando la geometria e la composizione delle superfici riflettenti presenti in scena.

Osservando la posizione uno della sorgente sonora sono state individuate due superfici molto vicine alla sorgente che potrebbero generare riflessioni importanti per la IR simulata.

Osservando i percorsi di riflessione generati dal sistema si osserva che il punto di riflessione con le due superfici sopra citate è molto lontano rispetto alla minima distanza tra le due superfici e la sorgente sonora. Questa potrebbe essere la causa dell'assenza di alcuni picchi nella IR.

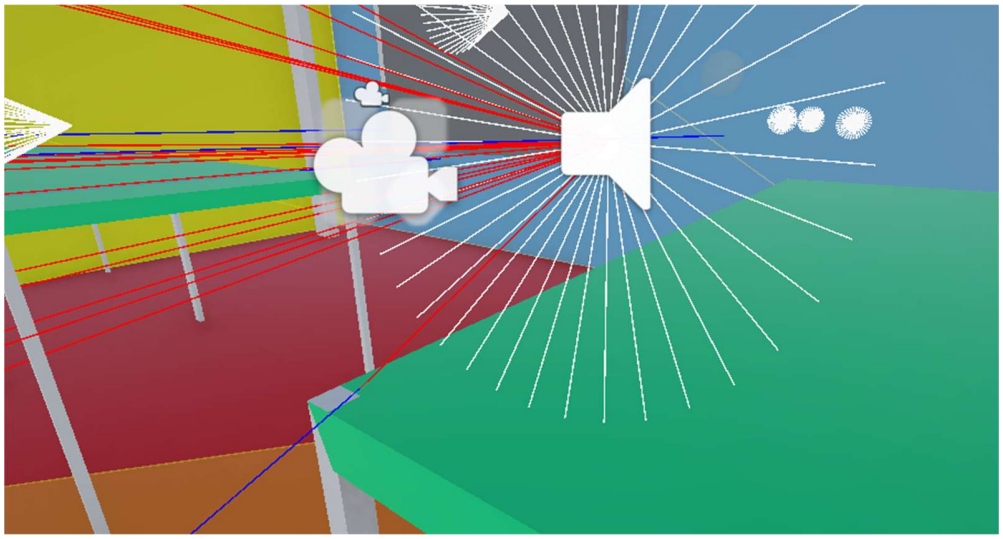

Figura 31 Particolare della riflesione che la sorgente sonora ha con uno dei tavoli nella scena

Nella figura 31 si nota come il raggio diretto proveniente dalla sorgente sonora (linea rossa) che collide con la superfice del tavolo in primo piano è molto distante rispetto al punto del tavolo più vicino alla sorgente. Per ovviare a questo problema è stata sfruttata una caratteristica intrinseca del sistema stesso. Essendo le superfici riflettenti legate solamente alla geometria dei collider che vengono applicati al Game Object con lo script per le superfici riflettenti, un aumento del numero di collider nella composizione del singolo modello geometrico permette anche un aumento delle possibili riflessioni che quell'oggetto può produrre.

Un primo test è stato effettuato modificando la grandezza totale del collider del tavolo, andando a mantenere soltanto il collider più vicino alla sorgente, così da forzare la riflessione più importante.

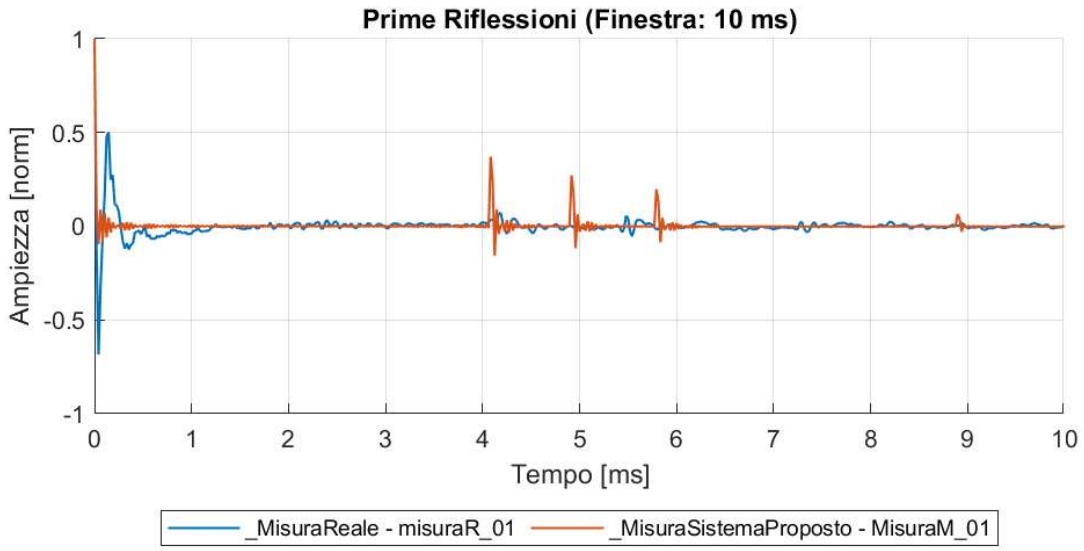

Figura 32 Confronto IR del sistema proposto e reale nella posizione di registrazione 1 con modifiche ai coefficienti e alla geometria del tavolo

Osservando il risultato del grafico in figura 32, si denota che la posizione dei picchi è variata. Questo conferma il fatto che, andando a modificare la geometria dei collider delle superfici si può agire sulla posizione dei picchi della IR simulata.

In un secondo test la superfice del tavolo è stata ricomposta andando ad utilizzare otto collider differenti, che hanno permesso al sistema di aumentare ad otto il numero di riflessioni per il singolo modello 3D.

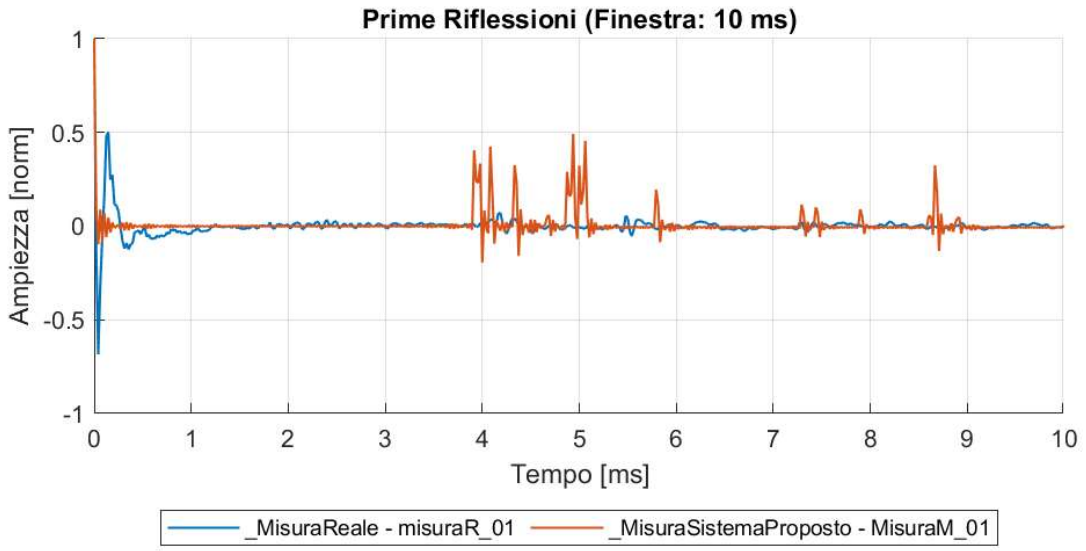

Figura 33 Confronto IR del sistema proposto e reale nella posizione di registrazione 1 con modifiche ai coefficienti e con aumento del numero di collider per la singola superfice del tavolo

Dal grafico in figura 33 si nota come il numero di picchi sia aumentato di molto rispetto al risultato precedente. Questo approccio è quindi fondamentale per aumentare il numero di riflessioni simulate a quelle reali, aumentando di conseguenza la fedeltà del sistema.

6.4. COMPARAZIONI CON LE POSIZIONI 2 E 3

Per confrontare i risultati ottenuti dai test con la prima posizione, sono state effettuate delle prove analoghe per le posizioni due e tre mostrate in precedenza. In entrambi i casi il ricevitore si trova molto più distante dalla sorgente rispetto al caso precedente. Questo permette l'osservazione del comportamento del sistema in una situazione molto diversa rispetto alla precedente.

Seguono i risultati dei test effettuati con le posizioni due e tre. I primi due grafici mostrano le IR senza modifiche alla geometria o ai coefficienti, poi seguono i grafici con solo modifiche ai coefficienti, dopo verranno mostrati i grafici con modifiche sia alla geometria che hai coefficienti, prima con soltanto la geometria dei tavoli ridotta per forzare le riflessioni e poi con la geometria dei tavoli composta da otto collider.

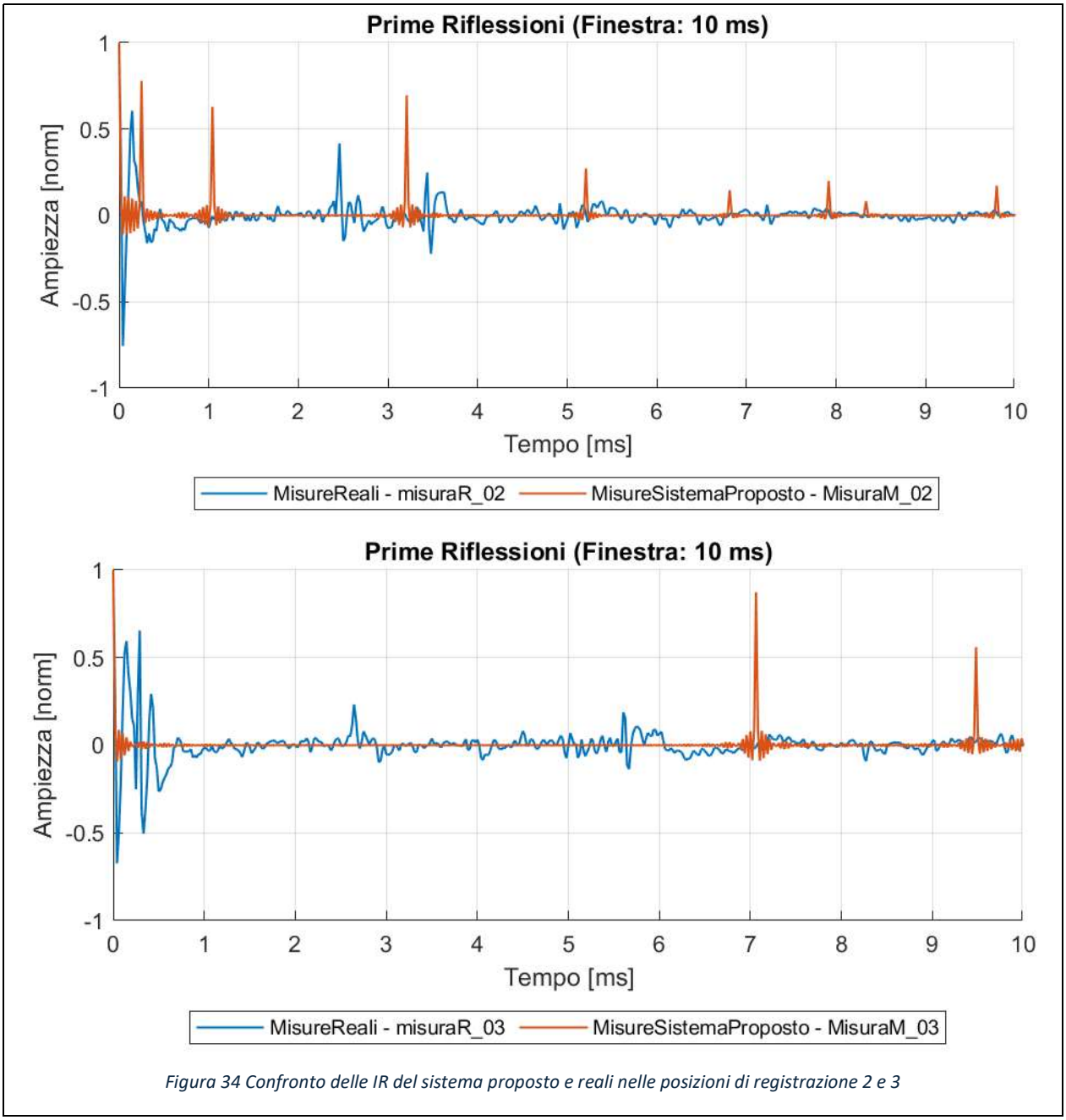

Figura 34 Confronto delle IR del sistema proposto e reali nelle posizioni di registrazione 2 e 3

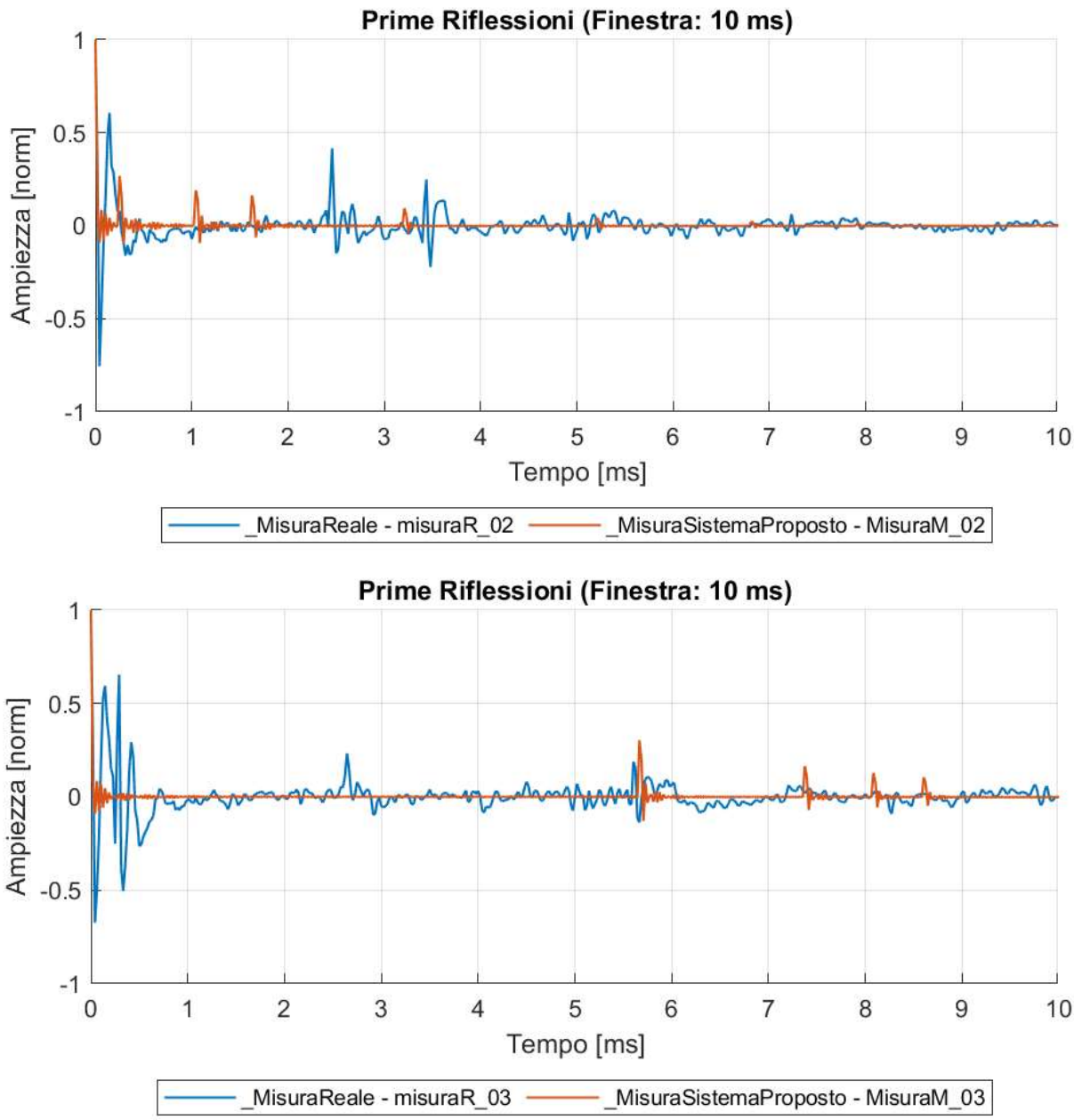

Figura 35 Confronto delle IR del sistema proposto e reali nelle posizioni di registrazione 2 e 3 con modifiche ai coefficienti di riflessione e di decrescita

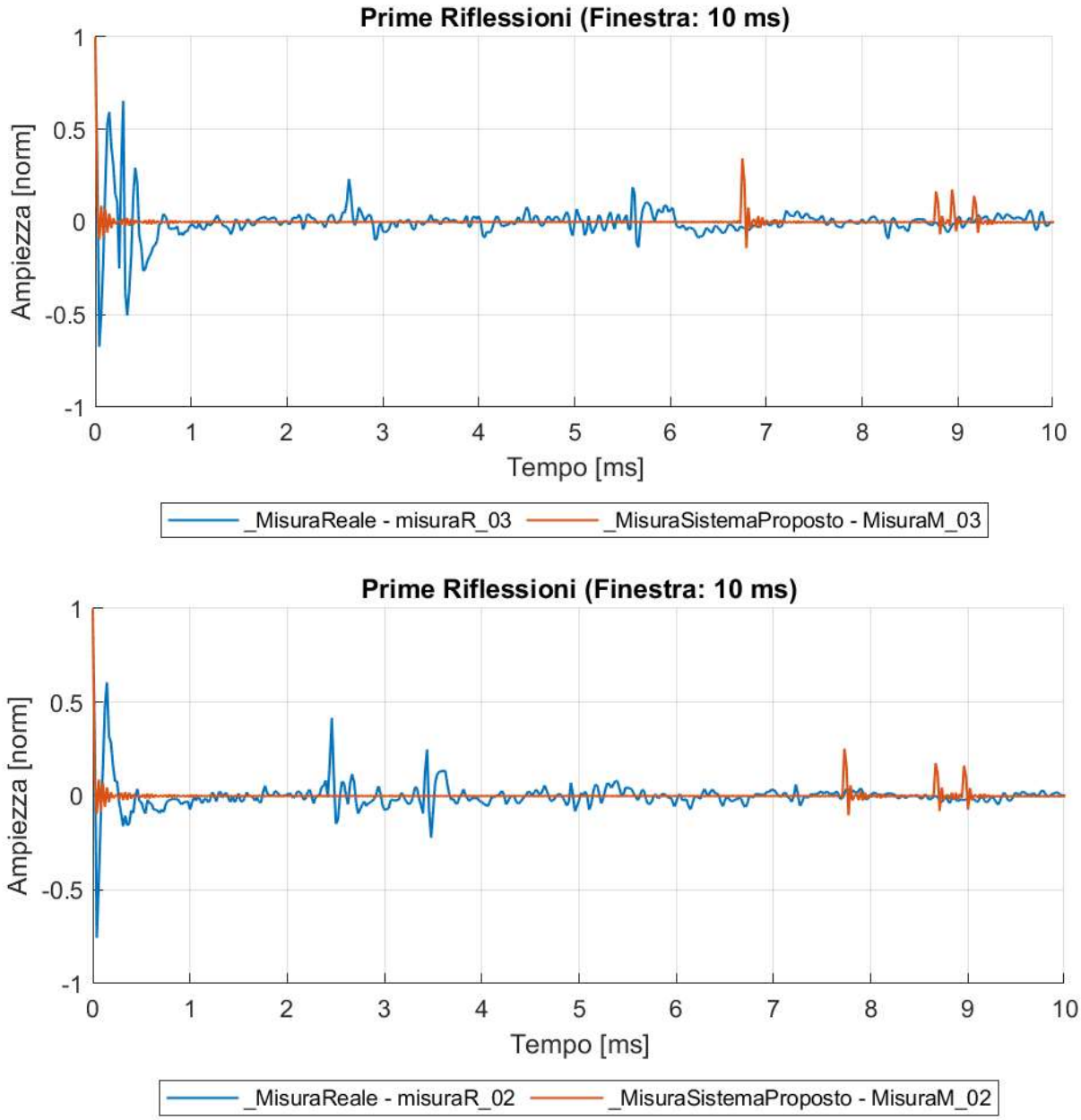

Figura 36 Confronto delle IR del sistema proposto e reali nelle posizioni di registrazione 2 e 3 con modifiche ai coefficienti e alla geometria del tavolo

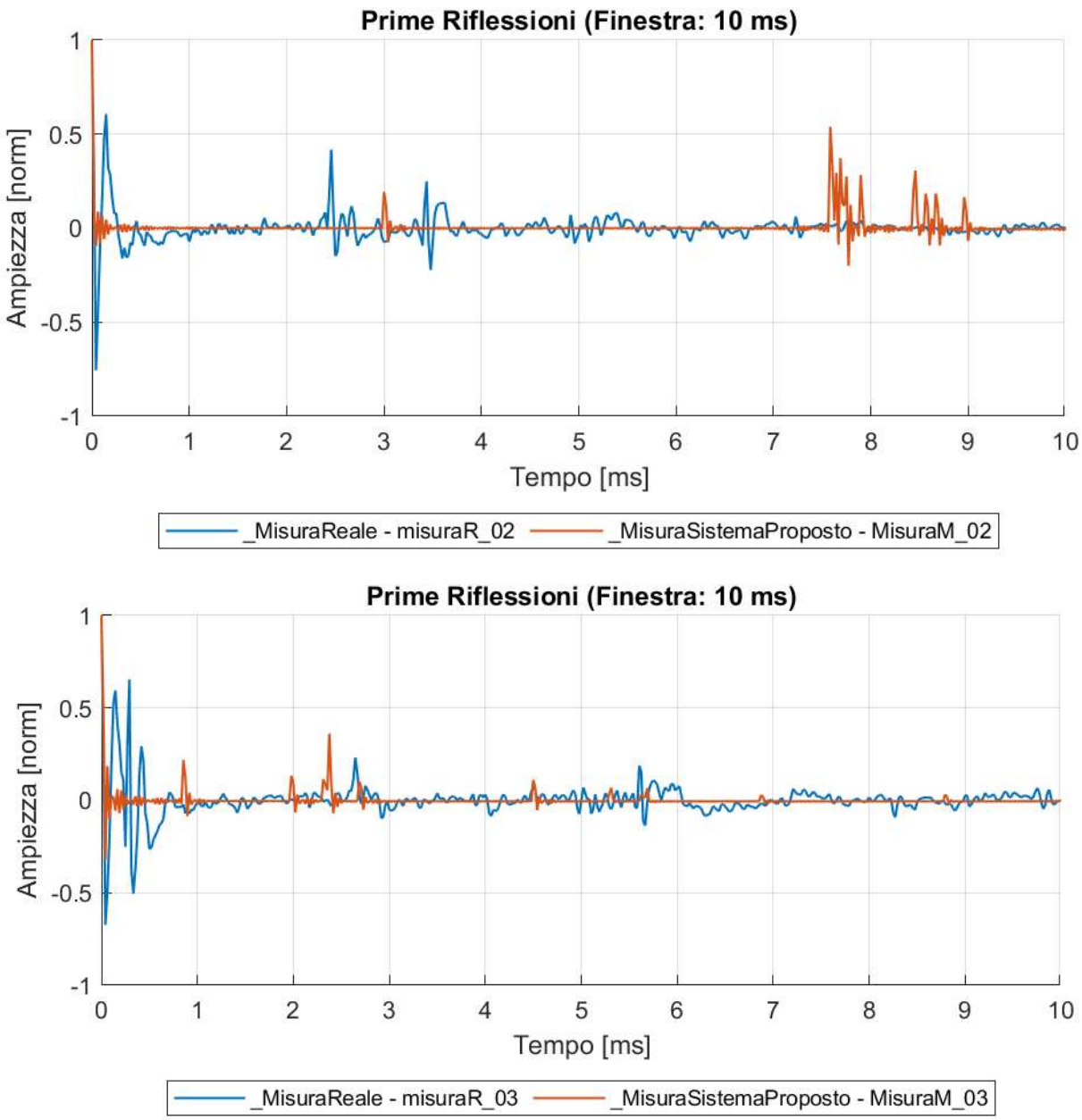

Figura 37 Confronto delle IR del sistema proposto e reali nelle posizioni di registrazione 2 e 3 con modifiche ai coefficienti e con aumento del numero di collider per la singola superfice del tavolo

Osservando i risultati ottenuti in figura 35 si nota che le modifiche ai coefficienti portano ad un miglioramento delle IR, mentre dai grafici in figura 36 e 37 le modifiche alla geometria non sembrano portare particolari miglioramenti, soprattutto ai grafici della seconda misura.

Si può quindi dire che modificare i coefficienti addetti al calcolo dell'intensità sonora delle sorgenti immagine porta un miglioramento alle IR a prescindere dalla posizione di ricevitore e sorgente, mentre la geometria sembra avere un comportamento dipendente dalla posizione dei due attori in scena.

7. LIMITI, CONCLUSIONI E POSSIBILI SVILUPPI FUTURI

Il presente lavoro di tesi ha riguardato lo studio, la progettazione e l'implementazione di un sistema di audio immersivo basato sull'Image Source Method (ISM) all'interno del motore di gioco Unity 3D. L'obiettivo principale è stato quello di estrarre i dati geometrici della scena virtuale per calcolare in modo deterministico le riflessioni sonore e generare IR coerenti con lo spazio simulato. Le IR ottenute sono state successivamente sottoposte a validazione sperimentale, confrontandole sia con le misurazioni acustiche effettuate in ambiente reale, sia con i risultati generati da una soluzione commerciale consolidata come il plugin per l'audio immersivo del Meta Quest SDK.

7.1. LIMITI DEL SISTEMA SVILUPPATO

Nonostante l'algoritmo implementato abbia dimostrato una buona efficacia nel tracciare le riflessioni acustiche primarie e secondarie, la fase di test ha evidenziato alcuni limiti intrinseci al metodo scelto e vincoli legati all'architettura software. Tra questi troviamo l'oneroso incremento del costo computazionale ad ordini elevati. L'ISM soffre di una crescita esponenziale dei cloni virtuali all'aumentare dell'ordine massimo di riflessione impostato e sebbene l'introduzione della tecnica di scansione conica abbia drasticamente ridotto i tempi di calcolo escludendo porzioni di spazio, il sistema tende comunque a saturare le risorse hardware in presenza di geometrie poligonali molto complesse. Un altro limite del sistema è l'assenza di fenomeni di diffrazione e scattering. Essendo un modello basato sull'approssimazione a raggi geometrici (ottica acustica), l'attuale sistema non simula i fenomeni d'onda tipici delle basse frequenze, come la diffrazione del suono attorno agli spigoli degli arredi o lo scattering causato dalle irregolarità delle superfici. Anche la semplificazione dello spettro in frequenza è un limite da tenere in considerazione. A differenza del plugin di Meta, che permette una discretizzazione dettagliata dei coefficienti di assorbimento su un grafico Intensità/Frequenza, il sistema sviluppato gestisce l'assorbimento e la riflessione mediante coefficienti globali scalari (α e β), non tenendo pienamente conto della risposta in frequenza dei materiali reali.

7.2. POSSIBILI SVILUPPI FUTURI

Sulla base dei risultati ottenuti e dei limiti riscontrati, si delineano diverse strade per l'evoluzione e il perfezionamento del sistema in lavori successivi.

Una naturale evoluzione del sistema prevede l'implementazione di componenti di filtraggio (filtri passa alto, passa basso e passa banda) per associare il calcolo geometrico delle riflessioni a una scomposizione del segnale in bande frequenziali. Questo permetterebbe di emulare più fedelmente l'assorbimento asimmetrico dei

materiali. Un'ottimizzazione tramite Multithreading o assegnando la computazione alla GPU potrebbe rendere il sistema scalabile anche in scenari di gioco complessi o simulazioni industriali ricche di poligoni. La logica di calcolo del modulo $ScanAmbient$ potrebbe essere spostata al di fuori del thread principale di Unity, oppure i calcoli per la proiezione dei raggi potrebbero essere assegnati direttamente alla GPU, così da migliorare le prestazioni.

7.3. CONCLUSIONI

La campagna di test sperimentali ha fornito dati fondamentali per valutare l'accuratezza del sistema sviluppato. Nelle prime fasi, i test preliminari sulle undici posizioni avevano mostrato discrepanze nette rispetto alle IR reali. L'ampiezza dei picchi e la loro posizione rispetto al tempo sono stati i principali aspetti critici individuati. La strategia di concentrare l'ottimizzazione sulla Posizione 1 ha permesso di comprendere l'impatto critico delle variabili ambientali, come i coefficienti di riflessione e di decadimento sonoro, ma anche di individuare un approccio efficace per aumentare la fedeltà delle IR simulate.

Intervenendo in modo mirato sulla calibrazione dei coefficienti di riflessione β e rifinendo le geometrie dell'ambiente 3D, si è ottenuto un miglioramento della similarità dei picchi simulati rispetto a quelli reali. Il confronto diretto con le posizioni 2 e 3 ha però confermato che la geometria della scena necessita di essere calibrata in base alla posizione della sorgente e del ricevitore. Questo porta alla conclusione che la soluzione proposta potrebbe essere un valido sistema per simulare le riflessioni sonore all'interno di un ambiente 3D, ma necessita di particolari accorgimenti nella creazione delle geometrie riflettenti in scena.

BIBLIOGRAFIA E SITOGRAFIA

[1] BAGNOLI, Mario; CIRSTEA, Silvia. Dynamic geometry-and material-dependent simulation of room impulse responses in a virtual gaming environment. In: *2017 International Conference on Optimization of Electrical and Electronic Equipment (OPTIM) & 2017 Intl Aegean Conference on Electrical Machines and Power Electronics (ACEMP)*. IEEE, 2017. p. 1095-1101.

[2] MARTIN, Vincent; ENGEL, Isaac; PICINALI, Lorenzo. Effects of geometrical acoustics model simplification on binaural reverberation. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

[3] PELZER, Sönke, et al. Integrating real-time room acoustics simulation into a cad modeling software to enhance the architectural design process. *Buildings*, 2014, 4.2: 113-138.

[4] KYRIAKAKIS, Chris. Fundamental and technological limitations of immersive audio systems. *Proceedings of the IEEE*, 1998, 86.5: 941-951.

[5] CUENCA, Jacques; GAUTIER, François; SIMON, Laurent. The image source method for calculating the vibrations of simply supported convex polygonal plates. *Journal of Sound and vibration*, 2009, 322.4-5: 1048-1069.

[6] SCHROEDER, Manfred R.; LOGAN, Benjamin F. " Colorless" artificial reverberation. *IRE Transactions on Audio*, 1961, 6: 209-214.

[7] MURPHY, Damian T.; BEESON, Mark J. Modelling spatial sound occlusion and diffraction effects using the digital waveguide mesh. 2003.

[8] MURPHY, Damian, et al. Acoustic modeling using the digital waveguide mesh. *IEEE Signal Processing Magazine*, 2007, 24.2: 55-66.

[9] FUNKHOUSER, Thomas, et al. A beam tracing approach to acoustic modeling for interactive virtual environments. In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 1998. p. 21-32.

[10] FUNKHOUSER, Thomas, et al. A beam tracing method for interactive architectural acoustics. *The Journal of the acoustical society of America*, 2004, 115.2: 739-756.

[11] KAJASTILA, Raine, et al. A distributed real-time virtual acoustic rendering system for dynamic geometries. In: *122th Audio Engineering Society Convention 2007*. AES, 2007.

[12] https://www.ea.com/ea-studios/motive/news/focus-on-deadspace-audio-design

[13] https://docs.unity.com/en-us

[14] https://dev.epicgames.com/documentation/unreal-engine/unreal-engine-5-8-documentation

[15] https://www.rhino3d.com/it/

[16] https://developers.meta.com/horizon/documentation/unity/meta-xr-acoustic-ray-tracing-unity-getting-started/

Dedica

A tutte le persone che mi hanno arricchito e fatto crescere, a tutti i miei amici protagonisti di tutte le mie avventure, a Mamma e Papà che hanno sempre creduto in me e sostenuto ogni mia scelta, a mia sorella Mariasole che mi ha sempre aiutato e ad Angela che mi riporta sempre con la testa a posto.