ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

RIPRODUZIONE DI MOVIMENTI 2D IN UNO SPAZIO TRIDIMENSIONALE

Elaborato in
REALTÀ VIRTUALE

Relatore
Prof. GUSTAVO MARFIA

Relatore estero
Prof. CECILIA MASCOLO

Corelatori
Dott. MATHIAS CILIBERTO
Dott. PASQUALE CASCARANO

Presentata da
SOFIA TOSI

Anno Accademico 2023 – 2024

"Tell me and I forget, teach me and I may remember, involve me and I learn." - Benjamin Franklin

Indice

0.1 Abbreviazioni v

1 Introduzione **1**

2 Stato dell'arte **2**
2.1 La fruizione della realtà virtuale nel contesto dell'apprendimento 2
2.1.1 L'anatomia umana nella realtà virtuale 2
2.2 La realtà virtuale nell'ambito della danza 4
2.2.1 VR_I: immersive virtual reality Dance Theatre 4
2.2.2 Un gioco di apprendimento della Labanotation 4
2.3 Contributo del progetto 5
2.4 Casi d'uso 5
2.5 Terminologia relativa ai visori 7
2.5.1 Differenza tra realtà virtuale, aumentata e mista 7
2.5.2 Gradi di libertà 7
2.5.3 Tipi tracciamento 8
2.5.4 Principali lenti per visori VR 9
2.5.5 Distanza interpupillare 11
2.5.6 Frequenza di aggiornamento 12
2.5.7 Campo visivo 13
2.5.8 Tipologie di display 13
2.5.9 Struttura dei subpixel 13
2.5.10 Stazioni base 13
2.6 I principali visori VR 17

3 Metodologia **20**
3.1 Calcolo dei dati riguardanti le posizioni delle giunture del corpo umano 20
3.2 Design dell'avatar 21
3.3 Mappatura del corpo: differenza tra il Motion Capture e Unity 23
3.4 Concetti fondamentali per una migliore comprensione della realizzazione del progetto 24
3.4.1 Cinematica diretta e inversa 24
3.4.2 Prodotto vettoriale 25
3.4.3 Quaternioni e angoli di Eulero 25
3.4.4 Gimbal lock 26
3.5 Riflessioni sui quaternioni 26
3.6 Tecnologia 26
3.7 Primo approccio: utilizzo delle posizioni 27
3.8 Secondo approccio: utilizzo delle rotazioni 28
3.8.1 Rappresentazione di un vettore 28

3.9 Calcolo di un vettore 30
3.9.1 Calcolo dell'angolo tra due vettori 31

4 Implementazione **32**
4.1 Calcolo dell'angolo tra due vettori, gestione dei casi particolari 32
4.2 Sistemi di coordinate left-handed e right-handed 34
4.3 Trasformazioni locali e globali 35
4.3.1 Avatar 3D 35
4.3.2 Configurazione visore in 3DOF 37
4.3.3 Assunzione delle posizioni corrette 37
4.3.4 Calcolo delle rotazioni delle giunture 38
4.3.5 Menù 39
4.4 Animate3D 40
4.5 Feedback con gli utenti 40

5 Conclusioni e sviluppi futuri **41**

Elenco delle figure

2.1 Cranio con fosse craniche evidenziate con animazioni virtuali . . . 3
2.2 VR_I: immersive virtual reality Dance Theatre. Sulla destra l'attrezzatura di tracciamento degli spettatori . . . 4
2.3 Interfaccia usata per imparare una coreografia . . . 5
2.4 Diagramma dei casi d'uso . . . 6
2.5 Differenza tra 3 Dof e 6 Dof . . . 7
2.6 Tipi di tracciamento, illustrazione di Daisy Dai (2024) . . . 8
2.7 Lenti Fresnel . . . 9
2.8 Effetto screen door. Come riporta Wikipedia (2024*c*), è un artefatto del display che si manifesta quando la distanza tra i pixel diventa evidente nell'immagine. I display di risoluzione bassa ne sono particolarmente affetti . . . 9
2.9 Effetto God rays spiegato da Pimax.com (2024): artefatto visivo che assomiglia a fasci di luce che si estendono da oggetti ad alto contrasto su sfondo scuro . 10
2.10 Lente pancake di Upload VR (2021) . . . 11
2.11 Lente asferica di VR Italia (2023) . . . 11
2.12 Frequenza di aggiornamento . . . 12
2.13 Sensori lighthouse di HTC, foto di Mechatech (n.d.) . . . 14
2.14 Stazioni base SteamVR . . . 15
2.15 Led posizionati sul visore e sui controllers del sistema di tracciamento del Oculus Rift, raffigurazione di Mechatech (n.d.) . . . 15
2.16 Sensori del sistema di tracciamento Costellation del Oculus Rift, illustrazione di Wikipedia (2024*b*) . . . 16

3.1 Le due fasi del framework MotionBERT . . . 20
3.2 Avatar utilizzati nel progetto . . . 21
3.3 Gerarchia parent-child degli avatar . . . 22
3.4 Scheletro dell'avatar . . . 23
3.5 Prodotto vettoriale con la rappresentazione dell'area, raffigurazione di Matthes and Drakopoulos (2022) . . . 25
3.6 Prodotto vettoriale e regola della mano destra, illustrazione di James Stewart (2015) . . . 25
3.7 Gimbal lock (Wikipedia (2024*a*)) . . . 26
3.8 Matrice di rotazione di un quaternione . . . 26
3.9 Effetto di "staccamento" delle parti dell'avatar . . . 28
3.10 Rappresentazione di un vettore, illustrazione di Hanna Pamuła (n.d.) . . . 29
3.11 I vettori rappresentati le parti del corpo del ballerino. La seconda immagine è corretta, mentre la prima vuole evidenziare un comune errore che causa il raggruppamento delle parti del corpo. . . . 30

3.12 I vettori rappresentati le parti del corpo del ballerino. La seconda immagine è corretta, mentre la prima vuole evidenziare un errore che porta a un'approssimazione del movimento. 31

4.1 Sistemi Right-handed e Left-handed, illustrazione di Learn OpenGL ES (2012) 34
4.2 Sistema globale e locale dell'avatar 35
4.3 I due avatar utilizzati nello sviluppo 36
4.4 Visualizzazione del frame 189 dell'animazione creata usando Matplotlib. . . . 37
4.5 Quaternion Online 38
4.6 Menù 39
4.7 Avatar che esegue la coreografia 39
4.8 Schema Animate3D 40

Elenco delle tabelle

2.1 Visori standalone 17
2.2 Visori non standalone parte 1 17
2.3 Visori non standalone parte 2 18

3.1 Mappatura tra giunture MotionBert e avatar di Unity 24

0.1 Abbreviazioni

VR	Virtual Reality
AR	Augmented Reality
MR	Mixed Reality
DoF	Degree of Freedom
FOV	Field of View
IPD	Distanza interpupillare
LCD	Liquid Crystal Display
OLED	Organic Light Emitting Diode
AMOLED	Active Matrix Organic Light Emitting Diode
QLED	Quantum Dot Light Emitting Diode
IMU	Inertial Measurement Unit
NaN	Not a Number

Capitolo 1

Introduzione

Apprendere nozioni e concetti in maniera passiva attraverso un libro è possibile, ma ciò che perdura nel tempo sono le emozioni che si provano nel svolgere un'azione, è quando si è veramente coinvolti che le nozioni rimangono impresse. Talvolta, anche una spiegazione di un insegnante può non essere sempre efficace e mantenere l'attenzione degli studenti. Anche nella poetica scientifica di Albert Einstein si evince che la sola fonte di conoscenza è l'esperienza. A tal fine, le tecnologie immersive come la realtà virtuale o mista posso essere un valido supporto. Negli ultimi anni, sono stati numerosi i progetti che hanno sfruttato questi strumenti per l'educazione e sono stati compiuti studi che ne dimostrano i benefici. Oltre all'insegnamento, posso essere utilizzati per creare delle esperienze suggestive che semplici video in 2D non possono offrire. In particolare, in questo applicativo verranno ricostruiti, mediante un avatar 3D, i movimenti di una persona a partire da video. Il contenuto visualizzato nell'ambiente tridimensionale, potrà essere arricchito con informazioni aggiuntive come i costumi (per evidenziare la moda o vestiti d'epoca), ambientazioni incantevoli o informazioni storiche e culturali. Si potranno incorporare anche curiosità e aneddoti o dare la possibiltà di personalizzare la scena o di rallentare o velocizzare i gesti. Uno dei possibili utilizzi di queste applicazioni è all'interno di un museo o di un'esibizione, per rendere l'esperienza di visita più coinvolgente e interattiva, ma gli ambiti che possono servirsi di queste tecnologie sono molteplici, dalla medicina, all'arte, all'ingegneria. In questo progetto si analizzeranno le potenzialità ottenute nel contesto della danza e l'applicativo utilizzerà i dati ottenuti da un sistema di motion capture.

Continuando la lettura di questo documento, nel Capitolo 2 si potrà analizzare lo stato dell'arte attraverso le scoperte e le ricerche esistenti in letteratura. Si analizzeranno i concetti principali di questo dominio, l'obiettivo e i casi d'uso di questo progetto. A seguire il Capitolo 3 descriverà la metodologia utilizzata e la situazione preliminare che rappresenta il punto di partenza di questo studio, mentre nel Capitolo 4 verrà illustrata l'effettiva realizzazione del progetto. Nelle conclusioni (Capitolo 5) verrà, infine, riassunto il lavoro svolto e verranno proposte possibili evoluzioni future.

Capitolo 2

Stato dell'arte

In letteratura, sono presenti diverse ricerche riguardanti la realtà virtuale. Alcune di esse permettono di enfatizzare il coinvolgimento e le forti emozioni che ne derivano come un teatro o un cinema nel mondo virtuale, altre invece sono più orientate all'apprendimento e all'educazione, come quelle dell'università del Maryland, la quale ha condotto uno studio "Virtual memory palaces: immersion aids recall", Eric Krokos, Catherine Plaisant, Amitabh Varshney (2018), in cui ha dimostrato che l'apprendimento tramite realtà virtuale porta a un miglioramento del 9 percento nella memoria e del 12 percento nel tempo di risposta rispetto all'apprendimento tramite schermo piatto.

In questo capitolo verranno presentati alcuni esempi, nello specifico, verranno proposte alcune ricerche correlate all'educazione e alla danza.

2.1 La fruizione della realtà virtuale nel contesto dell'apprendimento

2.1.1 L'anatomia umana nella realtà virtuale

Ricercatori dell'università di Salamanca e Madrid (Izard S. G., Juanes Méndez J. A., Palomera P. R. (2017)) hanno sviluppato un software per l'apprendimento dell'anatomia umana. A differenza di altre applicazioni che utilizzano video registrati a 360 gradi e, pertanto, possono essere mostrati solo con dispositivi in grado di riprodurre la registrazione in un ambiente sferico, l'applicativo sviluppato offre una visione tridimensionale. Grazie a questa tridimensionalità, si possono ammirare le ossa generate tramite l'utilizzo di sezioni radiologiche. L'applicativo è pensato sia per studenti, ma anche per aiutare i medici a prendere decisioni attraverso simulazioni, senza dover ricorrere a corpi reali.

Le interazioni con il software sono molteplici ed è presente un menù che permette di selezionare le attività da svolgere relative al cranio: ci sono tour guidati, test con domande e due tipi di assemblaggio delle ossa. Per quest'ultima, l'utente può scegliere tra due livelli di difficoltà, quello base con la visualizzazione dei punti di ancoraggio e quello avanzato che non prevede alcun aiuto. Ogni volta che viene inserito un osso nel posto corretto, viene fornita una spiegazione audiovisiva.

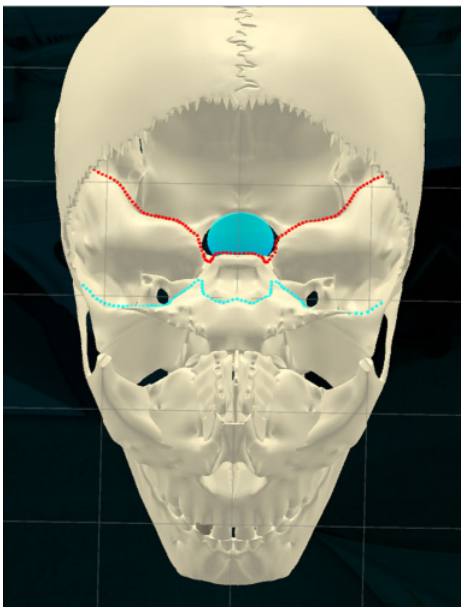


Figure 2.1: Cranio con fosse craniche evidenziate con animazioni virtuali

Per ottenere il cranio in Figura 2.1 è stato applicato l'algoritmo marching cubes, il quale permette di creare una mesh triangolare della superficie. Successivamente la mesh è stata semplificata e smussata. Infine, per riconoscere il movimento delle mani, è stato utilizzato un cellulare in quanto è un dispositivo alla portata di tutti. Questa applicazione è fondamentale non soltanto come allenamento, ma può essere usata anche per simulare interventi chirurgici e riuscire, in questo modo, a predirre eventuali problemi.

2.2 La realtà virtuale nell'ambito della danza

2.2.1 VR_I: immersive virtual reality Dance Theatre

Il coreografo Gilles Jobin e il team di Artanim nel 2017 hanno realizzato, grazie a una combinazione di arte e scienza, VR_I, un'esperienza di danza al teatro in realtà virtuale (Zhen and Luan (2021)). Nella versione offline sono supportate al massimo 5 persone, le quali possono facilmente muoversi in una stanza 8m x 5m con un visore e un computer sulla schiena e, grazie al sistema di motion capture, gli spettatori vengono tracciati agevolmente (fig:vri). Questo teatro digitale permette al pubblico di provare grandi emozioni e vivere un'esperienza

Figure 2.2: VR_I: immersive virtual reality Dance Theatre. Sulla destra l'attrezzatura di tracciamento degli spettatori

estremamente immersiva che con quello tradizionale non sarebbe possibile, il tutto con l'aiuto di diverse ambientazioni virtuali, ingrandimenti e rimpicciolimento dei ballerini e altri effetti speciali.

2.2.2 Un gioco di apprendimento della Labanotation

Una delle notazioni della danza più famose è la Labanotation e in Grecia è stato sviluppato un gioco per apprenderla (Rallis et al. (2018)). L'applicativo è in grado di riconoscere il movimento in real-time di un ballerino, grazie a sensori Kinect, e di generare un punteggio relativo all'accuratezza dell'azione rispetto alla notazione Laban. Il sensore estrae la variazione delle giunture nelle posizioni dei ballerini, mentre la direzione degli arti superiori e inferiori è calcolata in real-time durante il processamento dei dati relativi allo scheletro. Lo scheletro è composto da 20 giunture munite di coordinate 3D, parametri di rotazione e proprietà sullo stato del tracciamento.

Il framework proposto è composto da due parti: un sistema di motion capture che analizza e visualizza gli arti e un'applicazione interattiva che permette all'utente di eseguire una sequenza di pose partendo da simboli della notazione.

Il motion capture è costituito da più sottosistemi. Uno di essi è relativo alle trasformazioni geometriche e si occupa di calcolare l'angolo di piegamento degli arti, è, inoltre, presente un sottosistema per l'analisi del movimento e delle direzioni rispetto al sistema di coordinate locali. Infine, un'interfaccia interattiva permette la visualizzazione sia dei dati raccolti dai sensori Kinect che quelli processati.

Questo gioco è fondamentale per insegnare la notazione Laban in modo divertente e interattivo: all'utente viene mostrata una sequenza di passi di danza scritta secondo la notazione, il quale deve riprodurla correttamente. In questo caso un il simbolo verrà colorato di verde, altrimenti di rosso, Figura 2.3.

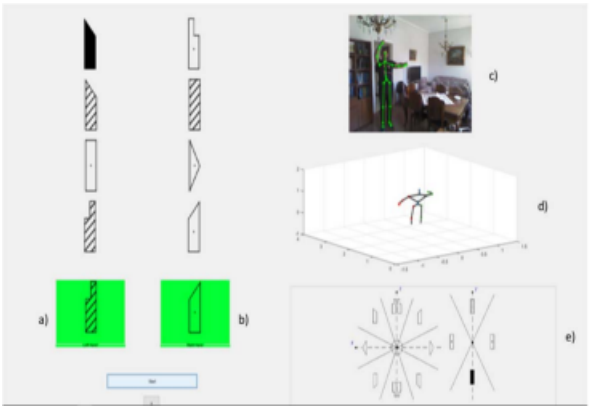


Figure 2.3: Interfaccia usata per imparare una coreografia

2.3 Contributo del progetto

Obiettivo: Realizzare un'applicazione in realtà virtuale che permetta di riprodurre, grazie a un avatar, i movimenti di una persona provenienti da video 2D.

Nel raggiungimento di questo obiettivo, verrà svolto uno studio incentrato sull'utilizzo delle rotazioni relative alle giunture delle parti del corpo, evitando di andare ad applicare direttamente le posizioni, in quanto, come dimostra uno studio che verrà riportato di seguito, provocherebbero un comportamento innaturale e scorretto. Verrà, inoltre, configurato un visore per la realtà virtuale in modalità Head tracking (3 gradi di libertà), estendibile, eventualmente, con la modalità Positional tracking (6 gradi di libertà).

2.4 Casi d'uso

Grazie al seguente schema (Figura 2.4) è possibile osservare i casi d'uso dell'applicazione che coinvolgono l'utente e il sistema.

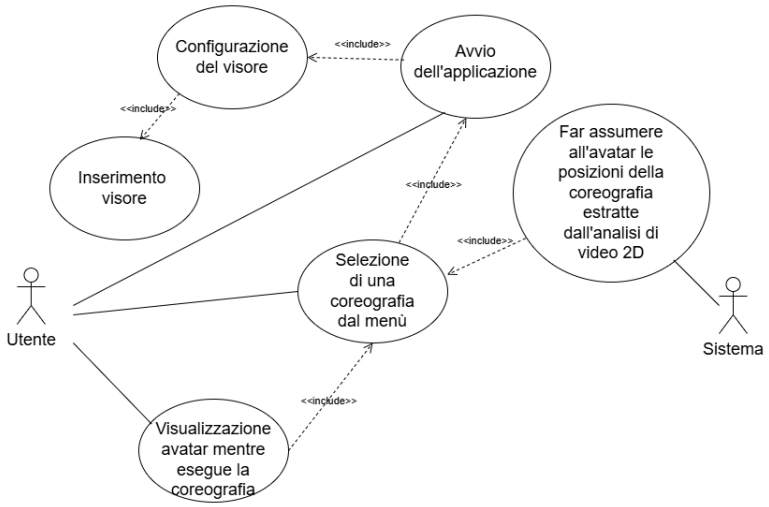


Figure 2.4: Diagramma dei casi d'uso

2.5 Terminologia relativa ai visori

2.5.1 Differenza tra realtà virtuale, aumentata e mista

La realtà virtuale è un ambiente completamente digitale nel quale l'utente si immerge tramite un visore. La realtà aumentata, invece, sovrappone elementi digitali all'ambiente reale, tramite l'utilizzo di dispositivi come, ad esempio, un cellulare o un tablet. Nella realtà mista l'utente può visualizzare e interagire con aspetti dell'ambiente virtuale e, contemporaneamente, mantenere un contatto con il mondo reale.

2.5.2 Gradi di libertà

I gradi di libertà (DoF) sono principalmente due: 3DoF o Head tracking e 6DoF o Positional tracking. Un'applicazione 3DoF permette di ruotare la testa in tutte le direzioni, ma non è permette il movimento, mentre quelle 6DoF permettono di muoversi nello spazio, oltre alla rotazione. Per maggior chiarimento si riporta la Figura 2.5 di Gold World (2022):

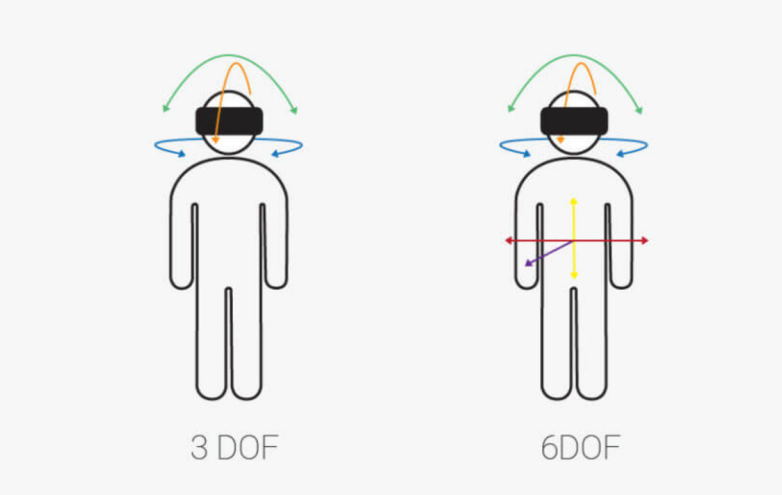
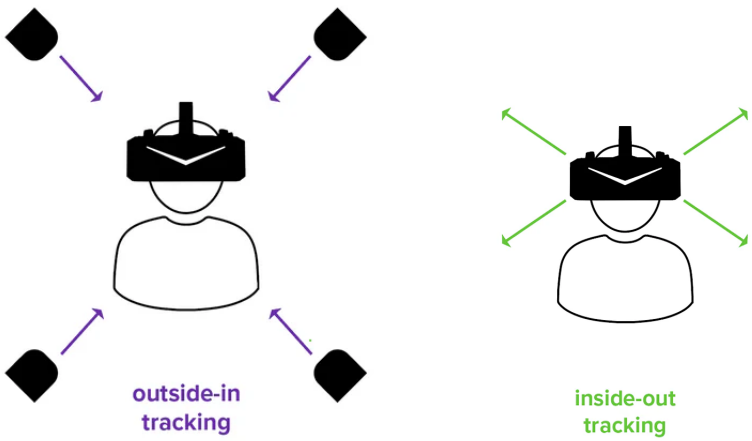


Figure 2.5: Differenza tra 3 Dof e 6 Dof

2.5.3 Tipi tracciamento

Il tracciamento dell'ambiente può essere svolto outside-in o inside-out. Nel primo caso, devono essere posizionati delle stazioni base con dei sensori che permettono di tracciare il movimento dell'utente, il quale deve indossare dei sensori o dei marcatori. Un vantaggio di questa tipologia è che non è soggetta a problemi di ostruzione da parte di oggetti posizionati tra i sensori e l'utente. Inoltre, può tracciare un'area più grande. Per quanto riguarda inside-out, invece, non richiede sensori esterni, ma è il visore stesso a delineare il movimento dell'utente. Questa tipologia è più economica e portatile, ma può avere problemi in ambienti con scarsa illuminazione. Per maggior chiarezza, si riporta un'illustrazione (Figura 2.6).



(a) Tracciamento outside-in (b) Tracciamento inside-out

Figure 2.6: Tipi di tracciamento, illustrazione di Daisy Dai (2024)

2.5.4 Principali lenti per visori VR

Secondo la prima community italiana sulla realtà virtuale, VR Italia (2023), la lente più utilizzata è la **Fresnel** (Figura 2.7). È una lente estremamente economica, si costruisce con pochi materiali, ha un ampio campo visivo e la quantità di luce che penetra è elevata. Nel caso la distanza focale sia corta l'effetto screen door (Figura 2.8), chiamato anche effetto zanzariera, è impercettibile. Tuttavia, se i raggi luminosi non si concentrano nella parte centrale della lente, si possono avere problemi che coinvolgono la perdità di nitidezza o l'effetto God rays (Figura 2.9). Questo è dovuto alla forma peculiare della lente e alla presenza di "creste" in corrispondenza degli angoli.

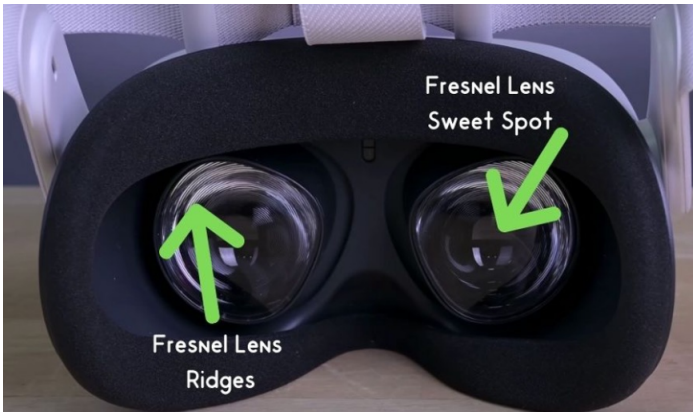


Figure 2.7: Lenti Fresnel

Figure 2.8: Effetto screen door. Come riporta Wikipedia (2024*c*), è un artefatto del display che si manifesta quando la distanza tra i pixel diventa evidente nell'immagine. I display di risoluzione bassa ne sono particolarmente affetti

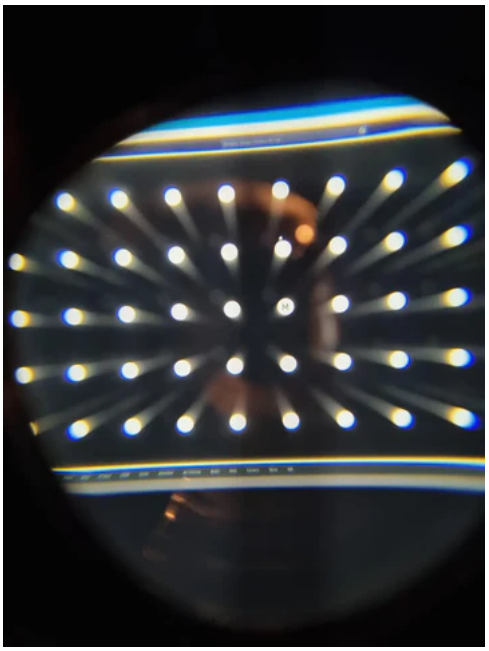


Figure 2.9: Effetto God rays spiegato da Pimax.com (2024): artefatto visivo che assomiglia a fasci di luce che si estendono da oggetti ad alto contrasto su sfondo scuro

Un'altra tipologia di lente è quella **pancake** (Figura 2.10). È costituita dall'unione di due lenti le quali permettono di ridurre la distanza tra display e lente. Grazie ad essa, si riesce ad avere una buona tolleranza per quanto riguarda il posizionamento del visore. Inoltre, il prezzo non è elevato e non sono presenti effetti God rays, essendo la lente è piatta. Questa forma, senza creste laterali, aumenta la nitidezza e il campo visivo, ma può essere soggetta a distorsioni sui bordi della lente.

Un grande svantaggio di questa tipologia è il fatto che la luce non penetra a sufficienza, pertanto, bisogna aumentare la luminosità del display, il che può portare a un aumento del consumo energetico. Nonostante esista una soluzione a questo problema (utilizzare la retroilluminazione led direzionale che permette di indirizzare la luce verso i pixel del display in maniera più efficiente insieme a una pellicola a riflessione diffrattiva che permette di riflettere la luce verso angoli specifici come quelli in corrispondenza dell'occhio dell'utente), per poterla applicare il prezzo della lente incrementerebbe notevolmente.

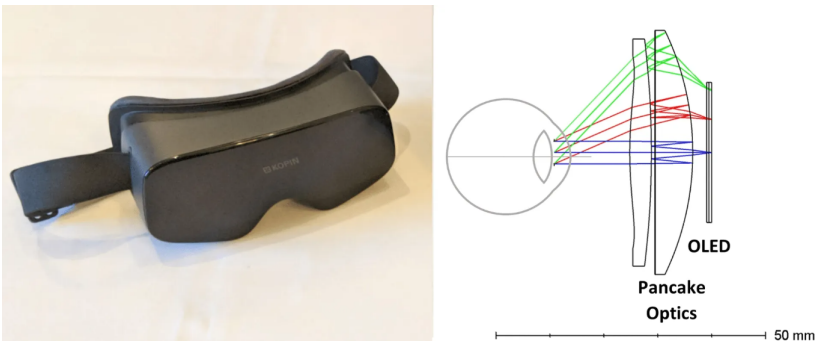


Figure 2.10: Lente pancake di Upload VR (2021)

I visori più costosi possono essere dotati di lenti **asferiche** (Figura 2.11) che, grazie all'assenza di creste, eliminano l'effetto God rays. Sebbene queste lenti siano costose e pesanti, hanno un campo visivo più ampio delle lenti a pancake e per produrle vengono utilizzati materiali di alta qualità.

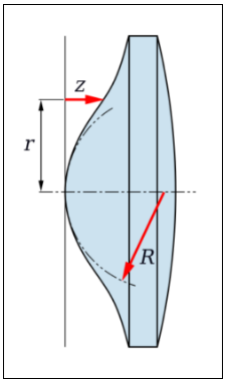


Figure 2.11: Lente asferica di VR Italia (2023)

2.5.5 Distanza interpupillare

La distanza interpupillare o IDP misura lo spazio tra le pupille negli occhi espressa in millimetri. Questo parametro è fondamentale per la regolazione delle lenti affinchè non ci siano problemi di visione e l'utente non provi mal di testa e affaticamento visivo. Questo parametro può essere configurato sia manualmente che automaticamente a seconda del visore.

2.5.6 Frequenza di aggiornamento

La frequenza di aggiornamento (refresh rate) è il numero di volte in cui l'immagine cambia su uno schermo in un secondo. Harvey Isitt (2024) ha realizzato un'illustrazione che mostra l'immagine al variare della frequenza di aggiornamento (Figura 2.12):

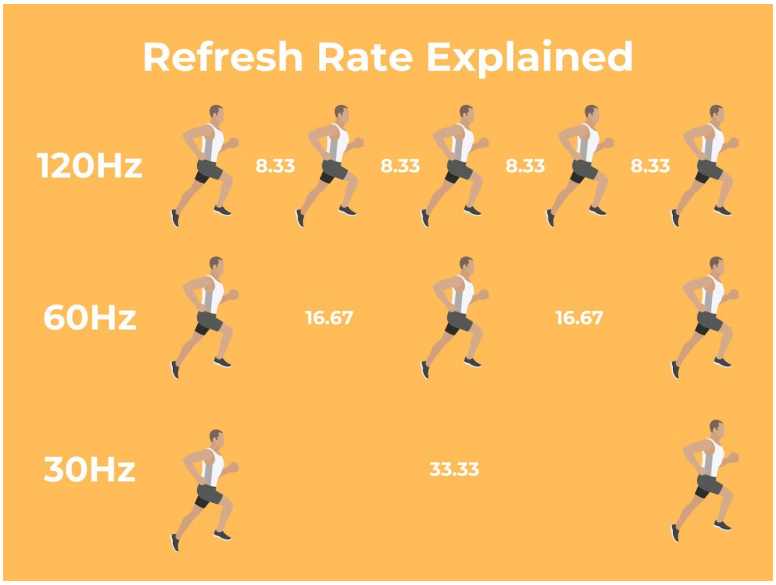


Figure 2.12: Frequenza di aggiornamento

2.5.7 Campo visivo

Il campo visivo (Field of View) si misura sia verticalmente che orizzontalmente e ce ne sono di due tipi: visibile o renderizzato. In quello visibile è l'angolo di visuale, espresso in gradi, che l'occhio umano riesce a percepire, sono escluse le parti coperte dal dispositivo. Mentre in quello renderizzato, l'angolo visualizzato è quello che viene calcolato dal motore grafico.

2.5.8 Tipologie di display

I display più utilizzati nei visori VR sono: OLED e LCD. Questi due display differiscono nelle modalità con le quali vengono realizzate le immagini. Per gli LCD (Liquid Crystal Display) vengono prodotte al passaggio della luce e gli schermi sono, come si può intuire dal nome, composti da cristalli liquidi. Per quanto rigurda gli OLED (Organic Light Emitting Diode), questi sono costituiti da diodi e, al passaggio degli elettroni, emettono fotoni e, di conseguenza, luce. Inoltre, gli OLED possono gestire ogni singolo pixel individualmente permettendo una riproduzione dei colori estremamente accurata e una maggior frequenza di aggiornamento. Si possono trovare anche visori con display AMOLED che sono una particolare tipologia di OLED, la quale utilizza una matrice attiva: ogni pixel è controllato da uno o più transistor che permettono di gestire in modo più efficiente l'accensione e lo spegnimento.

Talvolta, si possono trovare anche display QLED (Quantum Dot Light Emitting Diode) o Mini LED, che sono una versione migliorata degli LCD. I Mini LED hanno un diametro estremamente piccolo (0.2 mm) per tanto è possibile inserirne in maggior quantità e raggiungere contrasti più elevati. Invece, i QLED usano un filtro di particelle semiconduttori chiamati quantum dot che permettono di ottenere colori più brillanti e una maggiore luminosità.

2.5.9 Struttura dei subpixel

La struttura dei subpixel più diffusa in questi visori è la RGB stripe, in cui ogni pixel è composto da tre subpixel (rosso, verde e blu) disposti in fila. Un'altra conformazione è la Pentile Diamond. Pentile si riferisce al fatto che i subpixel possono essere condivisi tra pixel, diminuendo il numero di pixel necessari. Nella disposizione a diamante quelli verdi sono i più numerosi e sono organizzati in modo da formare un diamante. Essi sono responsabili della nitidezza dell'immagine, in quanto l'occhio umano è maggiormente sensibile al verde. La pentile diamond permette di ridurre il consumo di energia, ma in schermi con un minor numero di pixel, può portare a un effetto screen door più evidente.

2.5.10 Stazioni base

Alcuni visori necessitano di sensori perimetrali, chiamati stazioni base, per il tracciamento dei movimenti dell'utente. Questa tipologia di delineamento degli spazi risulta particolarmente efficace in ambienti molto vasti. Ogni produttore ha la proprie stazioni base: HTC per il suo HTC Vive Pro ha le Vive Base Station, le quali utilizzano sensori chiamati lighthouse (Figura 2.13). Questi dispositivi sono costituiti sia da LED che da laser. Il LED inizia a lampeggiare e, successivamente, i laser creano un fascio di luce nella stanza. Il visore e i controller hanno dei fotosensori che rilevano il bagliore. A seguire viene calcolata la posa dell'utente dalla quale si possono estrarre dati posizionali.

Figure 2.13: Sensori lighthouse di HTC, foto di Mechatech (n.d.)

Le stazioni base più diffuse sono le SteamVR le quali perlustrano la stanza con dei laser e delle pulsazioni. Essi calcolano, grazie a formule trigonometriche, la posizione di ogni sensore presente sul visore e nei controller. Inoltre, anche grazie alla presenza di IMU, l'unità di misura inerziale, si riescono a calcolare la velocità. la velocità angolare e l'orientamento. Di seguito si riporta un'illustrazione (Figura 2.14, Steam games (n.d.)) del sistema:

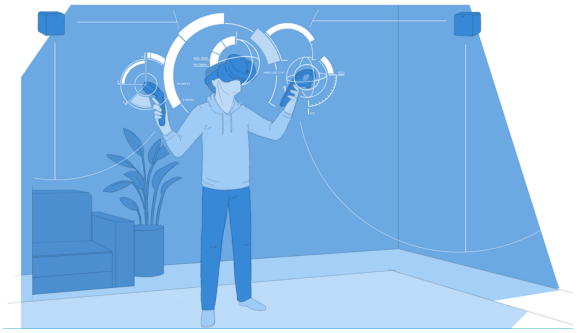


Figure 2.14: Stazioni base SteamVR

Si distingue dai dispositivi precedenti l'Oculus Rift che usa come sistema di tracciamento il Costellation. Il visore e i controllers sono dotati di LED infrarossi (Figura 2.15) che vengono rilevati da due sensori (Figura 2.16) in grado di riconoscere la loro luce e convertire la loro collocazione in dati posizionali.

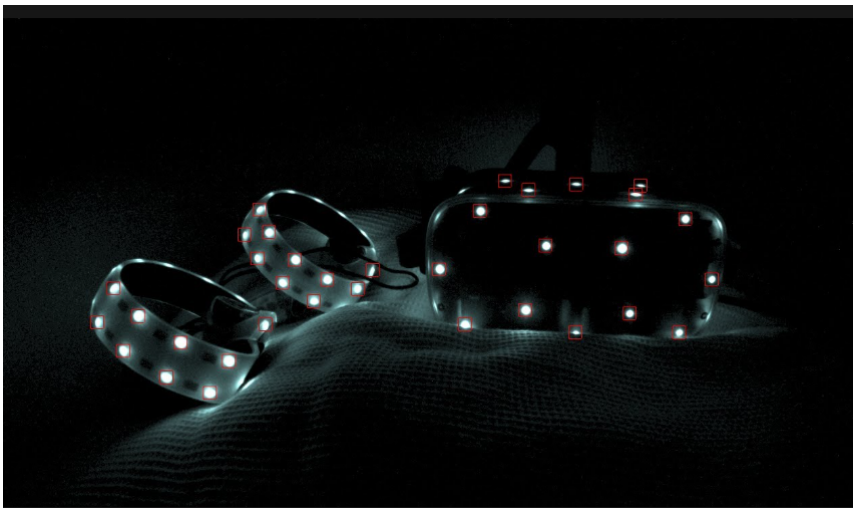


Figure 2.15: Led posizionati sul visore e sui controllers del sistema di tracciamento del Oculus Rift, raffigurazione di Mechatech (n.d.)

Figure 2.16: Sensori del sistema di tracciamento Costellation del Oculus Rift, illustrazione di Wikipedia (2024*b*)

In questa sezione verranno riportate le caratteristiche dei principali visori presenti nel 2024. Le informazioni sono state ottenute da Rory Brown (n.d.).

2.6 I principali visori VR

I visori si possono suddividere in due categorie: standalone (Table 2.1) e non standalone (Table 2.2, Table 2.3).

Caratteristica					
Nome	Oculus Quest 1	Meta Quest 2	Meta Quest 3	Meta Quest 3S	Pico 4
Risoluzione dell'immagine	1440 x 1600	1832 x 1920	2064 x 2208	1832 x 1920	2160 x 2160
Lenti	Fresnel	Fresnel	Pancake	Fresnel	Pancake
Refresh rate	72 Hz	120 Hz	120 Hz	120 Hz	90 Hz
Durata batteria	3 ore	3 ore	2.2 ore	2.5 ore	3 ore
IDP (ref:0.1 mm)	58-72	58-68	58-71	58-68	62-72
FOV verticale visibile	93 gradi	93 gradi	96 gradi	93 gradi	103 gradi
FOV orizzontale visibile	93 gradi	97 gradi	110 gradi	97 gradi	104 gradi
Tipo display	OLED	LCD	LCD	LCD	LCD
Struttura dei subpixel	Pentile Diamond	RGB stripe	RGB stripe	RGB stripe	RGB stripe
Base station	No	No	No	No	No
Tipo di tracciamento	Inside-out	Inside-out	Inside-out	Inside-out	Inside-out
Gradi di libertà	6	6	6	6	6

Table 2.1: Visori standalone

Caratteristica				
Nome	Varjo Aero	Somnium VR1	Pimax Crystal Light	HTC Vive Pro
Risoluzione dell'immagine	2880 x 2720	2880 x 2880	2880 x 2880	1440 x 1600
Lenti	Asferiche	Asferiche	Asferiche	Fresnel
Refresh rate	90 Hz	120 Hz	120 Hz	90 Hz
IDP (ref:0.1 mm)	57-73	60-76	58-72	61-72
FOV verticale visibile	73 gradi	100 gradi	105 gradi	98 gradi
FOV orizzontale visibile	102 gradi	125 gradi	115 gradi	98 gradi
Tipo display	Mini LED	QLED	QLED	AMOLED
Struttura dei subpixel	RGB stripe	RGB stripe	RGB stripe	Pentile Diamond
Base station	2 x SteamVR 2.0	2 x SteamVR 2.0	No	2 x Vive Base Station
Tipo di tracciamento	Marker-based	Marker-based	Inside-out	Marker-based
Gradi di libertà	6	6	6	6

Table 2.2: Visori non standalone parte 1

Caratteristica			
Nome	Valve index	Bigscreen Beyond	Oculus Rift
Risoluzione dell'immagine	1440 x 1600	2560 x 2560	1080 x 1200
Lenti	Fresnel	Pancake	Fresnel
Refresh rate	144 Hz	90 Hz	90 Hz
IDP (ref:0.1 mm)	58-70	55-72	59-72
FOV verticale visibile	104 gradi	90 gradi	88 gradi
FOV orizzontale visibile	108 gradi	102 gradi	87 gradi
Tipo display	LCD	Micro OLED	AMOLED
Struttura dei subpixel	RGB stripe	RGB stripe	Pentile Diamond
Base station	2 x SteamVR 2.0	2 x SteamVR 2.0	Costellation sensors
Tipo di tracciamento	Marker-based	Marker-based	Outside-in
Gradi di libertà	6	6	6

Table 2.3: Visori non standalone parte 2

I visori standalone, chiamati anche All-in-one, sono dispositivi che non necessitano di un computer per funzionare, in quanto tutto il processamento è svolto all'interno del visore stesso. Come spiega Patrick R (2023), questi visori sono più maneggevoli e portatili, ma hanno limiti grafici: non avendo a disposizione un computer, devono affidarsi al hardware integrato e non riescono ad avere la stessa potenza grafica di quelli non standalone. Inoltre, il tipo tracciamento utilizzato è inside-out, il che permette di raggiungere l'indipendenza dalle base stations, ma l'area delimitata è minore. Sebbene la tipologia all-in-one sia dipendente dalla durata della batteria, l'assenza di cavi permette all'utente una maggior libertà di movimento.

Capitolo 3

Metodologia

Il progetto, come spiegato precedentemente, si pone come obiettivo quello di portare in un ambiente 3D i movimenti effettuati da un soggetto ottenuti da video 2D. In particolare, i video 2D che sono stati utilizzati sono passi di danza classica eseguiti da un ballerino professionista e la modellazione 3D è stata effettuata tramite l'uso di un avatar nel mondo virtuale. I dati dai quali si è partiti sono le coordinate delle posizioni delle parti del corpo del ballerino, ottenute tramite un sistema di motion capture. In questo capitolo verranno affrontate nel dettaglio le sfide e le soluzioni adottate durante lo sviluppo del progetto.

3.1 Calcolo dei dati riguardanti le posizioni delle giunture del corpo umano

È stato utilizzato il framework MMPose (Open MMLab (2024)) per effettuare una stima delle posizioni 3D delle giunture del corpo di una persona in un video 2D dato in input. In particolare, i video utilizzati per questo progetto sono coreografie di passi di danza classica presenti su YouTube. Successivamente, è stato applicato il modello MotionBert con una rete neurale per migliorare i dati ottenuti e normalizzarli. La rete neurale è in grado di calcolare relazioni spazio-temporali a lungo raggio tra le giunture del corpo e di imparare le rappresentazioni del movimento. In particolare, il framework proposto da Peking University and Shanghai AI Laboratory (2023) è a due fasi (Figura 3.1): una di pretraining e una di finetuning.

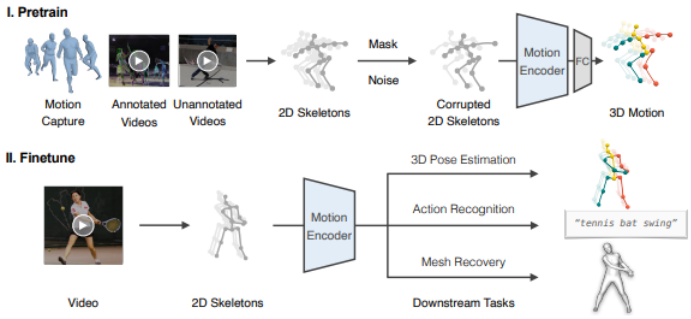


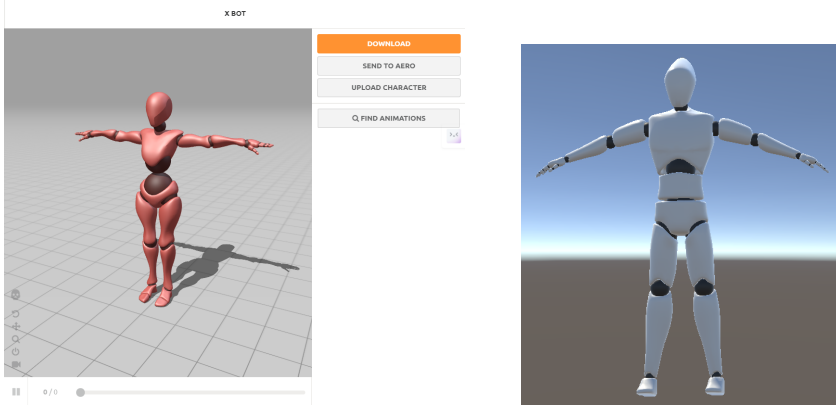
Figure 3.1: Le due fasi del framework MotionBERT

Durante la fase di pretraining, vengonno estratte sequenze dello scheletro in 2D da sorgenti dati eterogenee e vengono alterate con maschere e rumore. Successivamente è stato allenato l'encoder di movimento Dual stream Spatio-temporal transformer per ottenere movimenti

3D partendo dagli scheletri 2D. Infine si è eseguito il finetuning sulle rappresentazioni del movimento usando un MLP per ottenere mesh 3D.

3.2 Design dell'avatar

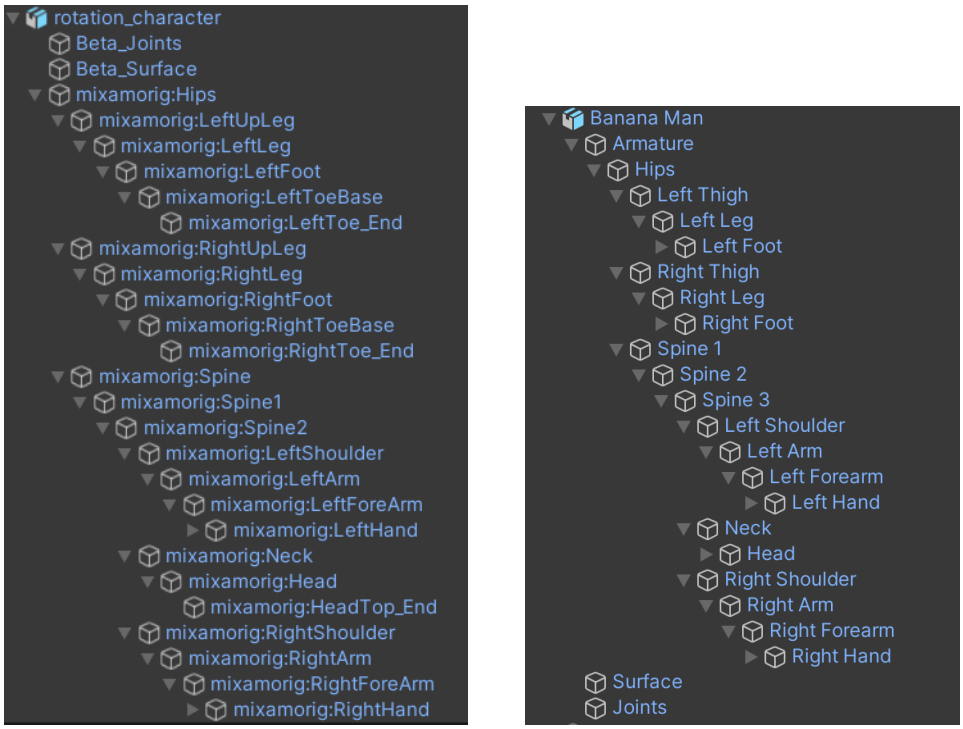
Nel realizzare questo progetto sono stati utilizzati due modelli (Figura 3.2). Il primo a sinistra trovato su Mixamo (n.d.) e il secondo a destra scaricato dall'Asset Store di Unity.



(a) Avatar X Bot (b) Avatar BananaMan

Figure 3.2: Avatar utilizzati nel progetto

Questi avatar sono oggetti complessi perchè sono costituiti da altri oggetti più semplici strutturati secondo una gerarchia parent-child. Come si può osservare nelle immagini (Figura 3.3), gli avatar sono costituiti da "Joints", "Surface" e dagli elementi dell'anatomia umana.



(a) Gerarchia di X Bot (b) gerarchia di BananaMan

Figure 3.3: Gerarchia parent-child degli avatar

Si può notare come l'oggetto con il ruolo di contenitore è Hips, il bacino dell'individuo. Esso, che rappresenta la radice, ha un ruolo di padre nei confronti di tutti gli altri oggetti: a partire dai figli diretti Spine e Left/Right Upleg/Thigh (che rappresentano il tratto lombare della spina dorsale e le due cosce) fino a figli indiretti come Left/Right Foot (i piedi). Questa organizzazione viene definita struttura gerarchica parent-child. In Unity, un oggetto complesso come un avatar è costituito da tre parti:

- Mesh: definisce la forma geometrica in termini di vertici, spigoli e facce. Questo componente fornisce solo la struttura dei dati. È poi richiesto un MeshFilter per fornire questi dati al MeshRenderer, il responsabile della resa grafica.
- Material: definisce l'aspetto dell'oggetto, come la texture, il colore e come interagisce con le fonti luminose.
- Skeleton: permette di creare relazioni fisiche tra oggetti per simulare le meccaniche del mondo reale.

In particolare, nella Figura 3.4 sottostante si può osservare lo scheletro dell'avatar: È colorato

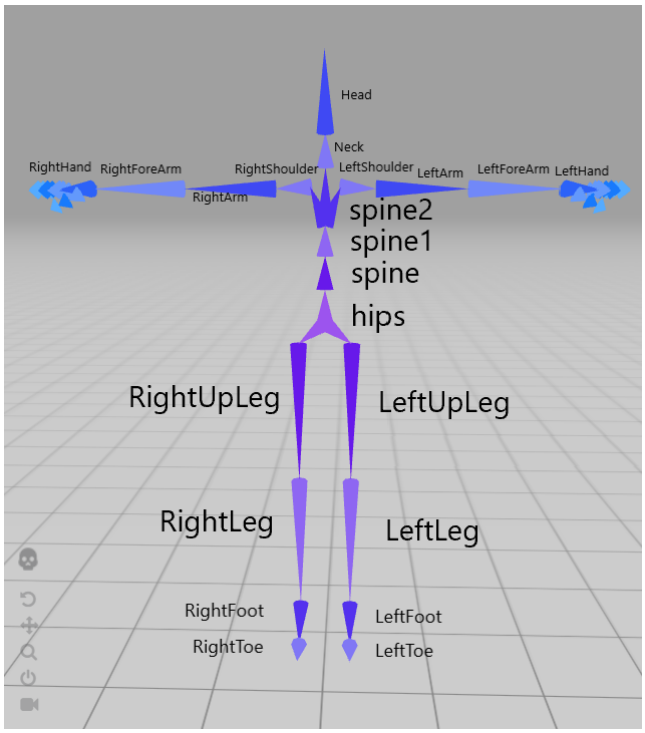


Figure 3.4: Scheletro dell'avatar

con un gradiente che va dal lilla fino all'azzurro. Il lilla rappresenta il padre, nonchè la radice della gerarchia, fino a raggiungere il viola, blu e l'azzurro con i figli più lontani.

3.3 Mappatura del corpo: differenza tra il Motion Capture e Unity

È stata applicata una mappatura tra le giunture provenienti da MotionBert e quelle degli avatar di Unity per apprendere a pieno la corelazione:

MotionBert	AvatarBotRosa	AvatarBananaMan
Root	Hips	Hips
Right Hip	RightUpLeg	Right Thigh
Right Knee	RightLeg	Right Leg
Right Ankle	RightFoot	Right Foot
Left Hip	LeftUpLeg	Left Thigh
Left Knee	LeftLeg	Left Leg
Left Ankle	LeftFoot	Left Foot
Torso	Spine	Spine1
Neck	Neck	Neck
Nose	-	-
Head	Head	Head
Left Shoulder	LeftArm	Left Arm
Left Elbow	LeftForeArm	Left Forearm
Left Wrist	LeftHand	Left Hand
Right Shoulder	RightArm	Right Arm
Right Elbow	RightForeArm	Right Forearm
Right Wrist	RightHand	Right Hand

Table 3.1: Mappatura tra giunture MotionBert e avatar di Unity

3.4 Concetti fondamentali per una migliore comprensione della realizzazione del progetto

3.4.1 Cinematica diretta e inversa

Le tecniche di animazione cinematica inversa (inverse kinematics) e quella diretta (forward kinematics) sono due metodi per controllare i movimenti e le posizioni di più oggetti. Entrambe posizionano un oggetto in uno specifico punto dello spazio e, successivamente, calcolano la posizione degli altri oggetti correlati. La differenza tra le due è che la cinematica diretta parte dall'osso più vicino alla radice, che rappresenta l'inizio della catena, e prosegue fino all'estremità, mentre la cinematica inversa calcola la componente finale e risale fino all'origine. In pratica, la cinematica diretta posiziona prima la spalla dell'avatar per poi risalire fino alla mano, mentre la cinematica inversa posiziona la mano e, successivamente, calcola la posizione degli arti terminando con la spalla.

3.4.2 Prodotto vettoriale

Il prodotto vettoriale tra due vettori a,b è rappresentato da un vettore perpendicolare ad entrambi ed equivale all'area del parallelepipedo che ha come lati le due componenti coinvolte nel prodotto (Figura 3.5). La direzione di questo nuovo vettore è lungo l'asse di rotazione in grado di allineare il primo vettore con il secondo. Per calcolare la direzione può essere usata la regola della mano destra (Figura 3.6): l'indice deve essere allineato al primo vettore e il medio sul secondo. Seguendo questa regola il pollice punterà nella direnzione del vettore rappresentante il prodotto vettoriale. La formula che permette di calcolarlo è:

$$|a \times b| = |a||b|sin(\theta)$$

dove θ ($0 <= \theta <= \pi$) è l'angolo di rotazione tra i due vettori a, b.

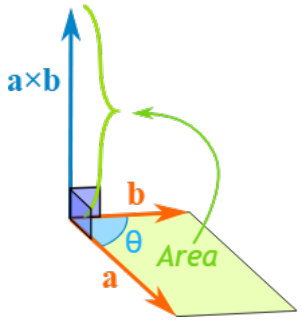


Figure 3.5: Prodotto vettoriale con la rappresentazione dell'area, raffigurazione di Matthes and Drakopoulos (2022)

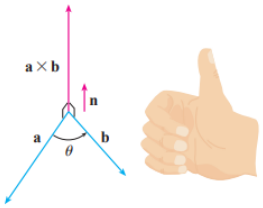


Figure 3.6: Prodotto vettoriale e regola della mano destra, illustrazione di James Stewart (2015)

Inoltre, si riporta di seguito un corollario preso da James Stewart (2015), il quale tornerà utile in seguito:

Due vettori non nulli a e b sono paralleli se e solo se:

$$a \times b = 0$$

3.4.3 Quaternioni e angoli di Eulero

Due rappresentazioni diverse per la rotazione di un oggetto nello spazio tridimensionale sono gli angoli di Eulero e i quaternioni. I primi sono tre angoli che rappresentano la rotazione attorno agli assi X, Y e Z. Invece i quaternioni, appartenenti alla famiglia dei numeri complessi, sono composti da quattro valori (x, y, z, w), dove x, y e z sono gli assi di rotazione e rappresentano le componenti immaginarie e w, la parte reale, è il coseno di metà dell'angolo di rotazione. Il formato più utilizzato per le grafiche in 3D e simulazioni di rotazioni è quello dei quaternioni.

Inoltre Unity, il software utilizzato per il progetto (Capitolo 4), offre un'interfaccia utente nella quale le rotazioni sono espresse in angoli di Eulero, ma internamente il motore utilizza i quaternioni, i quali sono modificabili tramite uno *script*. Infatti, essi permettono di evitare alcuni problemi legati alle rotazioni come il *gimbal lock*.

3.4.4 Gimbal lock

Il problema del Gimbal lock si verifica quanto due assi ruotano nella stessa direzione finendo per allinearsi e bloccarsi a vicenda (Figura 3.7). Questo sovrapposizione causa la perdità di un grado di libertà rendendo l'oggetto incapace di ruotare attorno a uno degli assi provocando, quindi, una rotazione in uno spazio bidimensionale degenere. Si noti che nella figura l'asse

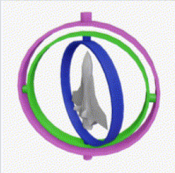


Figure 3.7: Gimbal lock (Wikipedia (2024*a*))

verde y (pitch) e quello magenta z (yaw) si sovrappongo.

3.5 Riflessioni sui quaternioni

$$R = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{bmatrix}$$

Figure 3.8: Matrice di rotazione di un quaternione

La matrice riportata in figura (Figura 3.8) è la matrice di rotazione del quaternione q(w,x,y,z). Ogni elemento rappresenta la rotazione che il quaternione esegue nei confronti di un vettore 3D. Inoltre, per ruotare un vettore usando un quaternione si può usare la formula:

$$v' = qvq^{-1}$$

dove v è il vettore, q il quaternione e q^{-1} il suo coniugato.

3.6 Tecnologia

- **Unity** (2022.3.44f1): in quanto la curva di apprendimento è più bassa rispetto ad altri motori grafici, la documentazione e asset store sono molto vasti. Inoltre, c'è un grande supporto per l'integrazione nativa dei visori e le funzionalità del progetto possono essere estese con numerosi pacchetti.
- **Pacchetto Animation Rigging**: permette di realizzare animazioni realistiche e avanzate tramite l'utilizzo di vincoli e cinematica inversa che controllano le varie parti del corpo.
- **Pacchetto XR Interaction Toolkit**: permette di gestire i movimenti del visore e dei controllers.

- **MotionBERT**: permette di computare, grazie a una rete neurale, le posizioni delle giunture del corpo umano in uno spazio 3D partendo da dei video 2D.
- **MMPose**: permette di stimare le pose di una persona a partire da video 2D.

3.7 Primo approccio: utilizzo delle posizioni

Avendo a disposizione dati che rappresentano coordinate di posizioni delle parti del corpo, in un primo tentativo si è provato a utilizzarle per muovere l'avatar. I vari motori grafici come Unity o Unreal Engine mettono a disposizione dei vincoli per modellare i movimenti. In particolare, Unity fornisce un pacchetto chiamato Animation Rigging, il quale comprende costrutti per controllare le animazioni, inclusi quelli che utilizzano la cinematica.

Durante una sessione di Problem Solving, si è valutata la possibilità di usare i seguenti meccanismi:

- **Two Bones IK Constraint**: permette di posizionare in modo preciso un oggetto target corrispondente a una parte finale di un arto, come una mano o un piede, e ne computa in modo automatico la posizione delle due componenti restanti, la parte dell'arto superiore (Coscia/Braccio) e quella inferiore (Polpaccio/Avambraccio).
- **Chain IK Constraint**: è simile al Two Bones IK Constraint, ma permette di gestire una catena più lunga di ossa.
- **Position Constraint**: a differenza degli altri due, non è un vincolo relativo alla cinematica inversa. Permette di posizionare un oggetto su un target. Questo costrutto non modifica direttamente tutte le articolazioni della catena.

I vincoli relativi alla cinematica inversa e diretta sono stati ritenuti non ottimali per questo progetto, in quanto calcolano in modo automatico la posizione delle articolazioni restanti. In questo specifico caso, si hanno a disposizione precise posizioni di tutte le parti del corpo e si vuole che l'avatar si collochi perfettamente, rispettando i dati. Per lo stesso motivo sono state scartate animazioni predefinite da usare con la componente Animator, in risposta all'avvenimento di input. È importante, quindi, ricordarsi di rimuovere tale componente, altrimenti i dati verranno sovrascritti.

Per ottenere un controllo più specifico si è poi pensato di utilizzare il Position Constraint tuttavia, utilizzando questo vincolo, si andava a peggiorare le prestazioni e complicare la soluzione. Si sarebbero dovuti creare degli oggetti target relativi alle parti del corpo, dai quali l'avatar sarebbe stato dipendente. Inoltre, ogni vincolo sarebbe stato analizzato per ogni frame, aumentando il carico d'elaborazione. Invece, per spostarsi in una determinata posizione, si ottengono risultati simili con un semplice script, ottenendo però i risultati in modo più leggero e ottimizzato, senza andare a creare oggetti aggiuntivi.

Utilizzando un script basato sulle posizioni sono, tuttavia, sorte alcune problematiche. L'avatar è un oggetto complesso le cui parti hanno una relazione parent-child. Andando a modificare direttamente le singole componenti, non si rispettava correttamente la gerarchia e si andava a creare un effetto di "staccamento" (Figura 3.9).

Si è tentato di risolvere il problema con i seguenti due approcci:

1. **Computazione delle posizioni non assolute, calcolando la differerenza rispetto alla posizione precedente**
2. **Aggiornamento delle giunture in relazione al genitore**

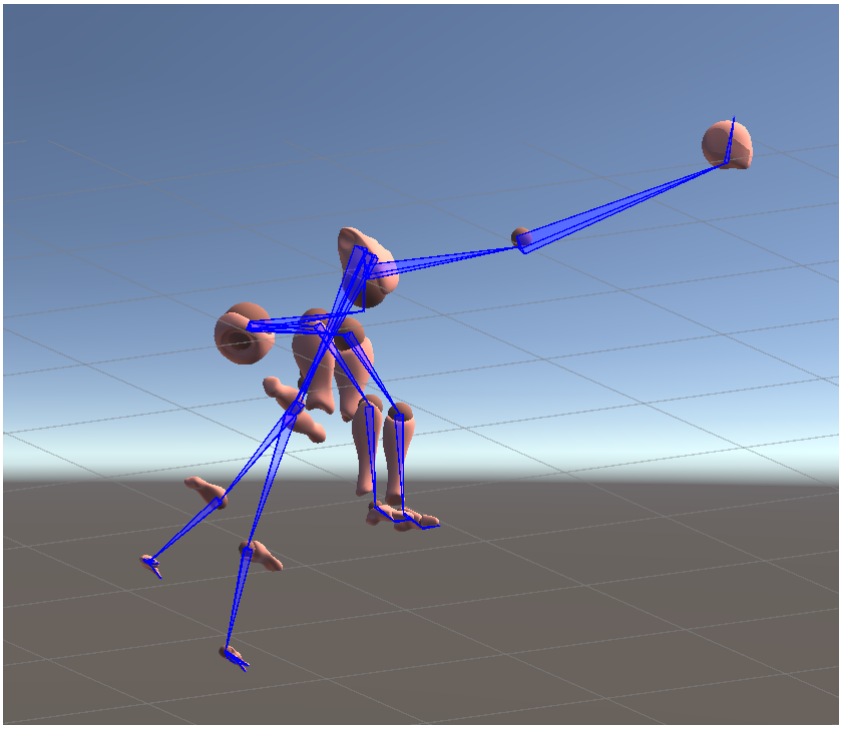


Figure 3.9: Effetto di "staccamento" delle parti dell'avatar

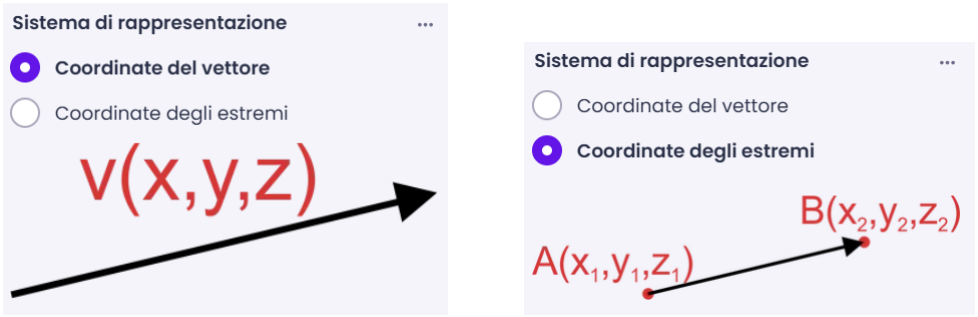
Nel primo caso si è calcolata e applicata la differenza (delta) tra la posizione attuale e quella precedente, mentre nel secondo le posizioni vengono aggiornate in maniera più fluida, partendo dalla radice fino agli ultimi figli. Tuttavia, entrambi i metodi non hanno portato al risultato sperato e le parti risultavano ancora staccate.

3.8 Secondo approccio: utilizzo delle rotazioni

Si è pensato, allora, di scegliere una strada alternativa, che non coinvolgesse le posizioni, ma bensì le rotazioni. Esse non sono altro che angoli, quindi si è deciso di ottenere dalle coordinate delle posizioni i rispettivi vettori. Utilizzando questi vettori si è calcolato l'angolo tra quello del frame attuale e il successivo.

3.8.1 Rappresentazione di un vettore

Un vettore può essere rappresentato in due modi: o attraverso le coordinate (Figura 3.10(a)) o le due coordinate dei suoi estremi (Figura 3.10(b)).



(a) Coordinate del vettore (b) Coordinate degli estremi

Figure 3.10: Rappresentazione di un vettore, illustrazione di Hanna Pamuła (n.d.)

Nel primo caso non sono necessari calcoli aggiuntivi, nel secondo caso bisogna sottrarre alla coordinate dell'estremo finale quelle del punto iniziale.

3.9 Calcolo di un vettore

Per poter calcolare le rotazioni è necessario ottenere i vettori rappresentanti le giunture del corpo. A prescindere dalla rappresentazione usata, è importante avere ben chiaro come devono essere, altrimenti l'animazione risultante sarà soggetta a problemi.

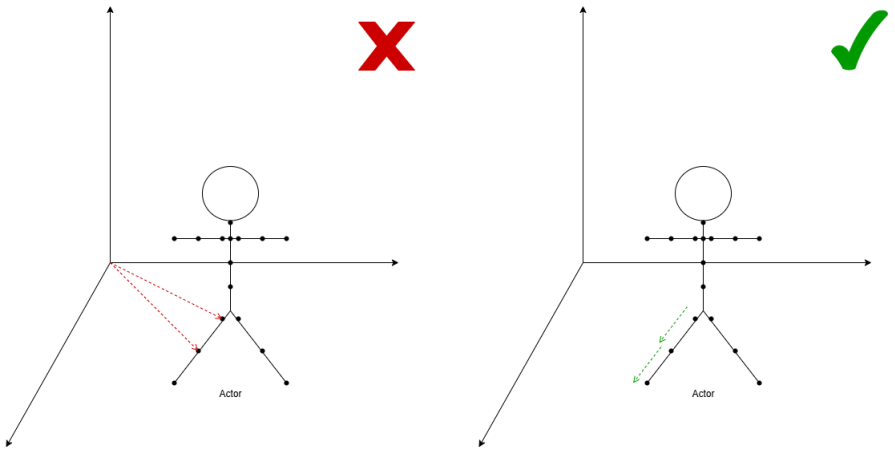


Figure 3.11: I vettori rappresentati le parti del corpo del ballerino. La seconda immagine è corretta, mentre la prima vuole evidenziare un comune errore che causa il raggruppamento delle parti del corpo.

Nella parte a sinistra della Figura 3.11, l'estremo iniziale coincide con l'origine del sistema di riferimento, mentre quello finale è sovrapposto alla componente del corpo. Questa versione risulta sbagliata e, nel caso venga applicata, si otterrà un avatar tutto accartocciato su se stesso. Infatti, i valori provenienti da MotionBert sono normalizzati tra -1 e 1 e questo implica che la distanza delle varie giunture del ballerino non sarà rispettata, ma, al contrario, le componenti saranno tutte raggruppate.

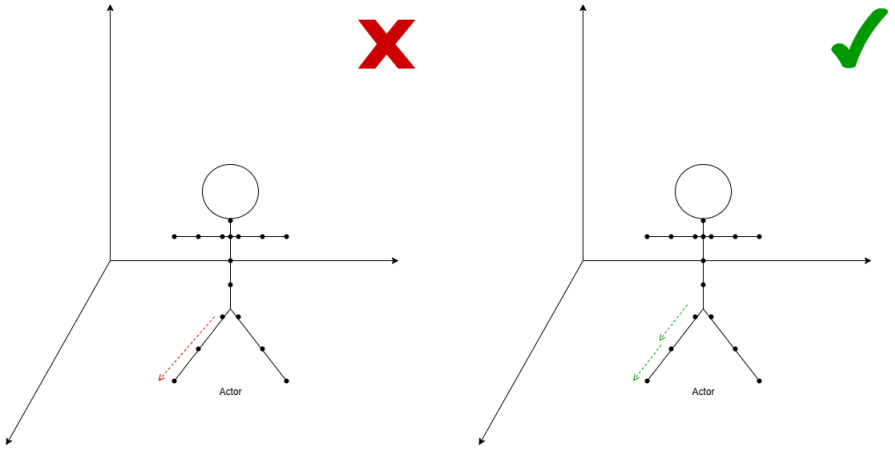


Figure 3.12: I vettori rappresentati le parti del corpo del ballerino. La seconda immagine è corretta, mentre la prima vuole evidenziare un errore che porta a un'approssimazione del movimento.

Un altro modo sbagliato per considerare i vettori lo mostra la prima parte della Figura 3.12. In questo caso, l'estremo iniziale è la parte del corpo della quale si vuole calcolare la rotazione, mentre quello terminale è il figlio finale della gerarchia delle parti del corpo. In questa casistica l'errore consiste nel non rispettare la struttura gerarchica dell'avatar. La relazione tra le singole giunture viene ignorata, portando a una grande approssimazione, in quanto non vengono considerate le rotazioni intermedie. L'animazione risultante sarà, perciò, molto più rigida dell'originale.

Per evitare di commettere questi errori, bisogna considerare gli estremi come mostrato nell'illustrazione a destra nelle due immagini (Figura 3.11 e Figura 3.12): quello iniziale è la parte del corpo della quale si vuole calcolare la rotazione, mentre quello finale è la parte successiva.

3.9.1 Calcolo dell'angolo tra due vettori

Una volta ottenuti i vettori, la formula per il calcolo dell'angolo è la seguente:

Dati a,b due vettori dei quali si vuole calcolare l'angolo,

$$\alpha = \arccos\left(\frac{a \times b}{|a||b|}\right)$$

dove

$$a \times b = a_x b_x + a_y b_y + a_z b_z$$

è il prodotto scalare tra a e b,

$$|a||b|$$

è il prodotto vettoriale (in inglese cross product) della magnitudine dei vettori a, b che si esprime con:

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Nel progetto b sarà il frame corrente, a il vettore rappresentante la rotazione zero nel sistema di *Unity*.

Capitolo 4

Implementazione

Una particolare sfida che si è affrontata durante lo sviluppo del progetto è stata quella di calcolare le rotazioni delle giunture dell'avatar.

4.1 Calcolo dell'angolo tra due vettori, gestione dei casi particolari

Si sono analizzati quattro casi specifici che si sono dovuti trattare separatamente per garantire un corretto funzionamento.

- **Vettori paralleli**: Nel caso di vettori paralleli, la rotazione non avviene. Per tanto l'angolo è zero gradi.

```
            #parallel vectors
            if np.linalg.norm(cross_product) == 0.0:
                return np.array([1, 0, 0, 0])

```

  Il prodotto vettoriale dei due vettori creerebbe problemi: essendo uguale a zero, si andrebbe ad eseguire la divisione

```
        normalized_rotation_axis = cross_product / np.linalg.norm(
    cross_product)

```

  con lo zero al denominatore provocando un risultato indefinito. Teoricamente parlando, è come se si volesse calcolare l'area di un parallelepipedo che ha come lati dei vettori paralleli: dal punto di vista geometrico, non è possibile misurare l'area. In python, inoltre, si genererebbe un *warning* in fase di esecuzione avente come risultato NaN (Not a Number). Per evitare ciò si è controllato che il prodotto vettoriale non fosse 0. Questo caso è stato gestito restituendo come valore (1, 0, 0, 0) che rappresenta il quaternione della rotazione nulla, quella con zero gradi.

- **Vettori anti-paralleli**: Sono vettori paralleli ma con la direzione opposta. L'angolo tra di essi è di 180 gradi.

```
            if np.isclose(dot_product, -1.0): #anti-parallel vectors
                # Return 180 degrees rotation around any orthogonal
    axis (perpendicular)
                perpendicular_axis = np.cross(A, [1, 0, 0])
                if np.linalg.norm(perpendicular_axis) == 0:  # If A
    is along x-axis, use y-axis
```

```
11                      perpendicular_axis = np.cross(A, [0, 1, 0])
12                  perpendicular_axis /= np.linalg.norm(
    perpendicular_axis)
13                  return axis_angle_to_quaternion(perpendicular_axis,
    np.pi)
14
```

I due vettori sono antiparalleli se il loro prodotto scalare è approssimabile a -1. In tal caso, viene calcolato il prodotto vettoriale tra il vettore A e quello unitario relativo all'asse X. Se l'asse X è parallelo ad A, ne verrà usato un altro. Si noti nel codice che il vettore perpendicolare per essere usato come asse di rotazione occorre che abbia modulo uguale a uno. Per ottenere questo risultato è stata applicata la normalizzazione. Infine, viene calcolato il quaternione che rappresenta la rotazione di 180 gradi attorno all'asse perpendicolare al vettore A.

- **Vettori degeneri**: se uno dei due vettori è approssimabile a (0, 0, 0) allora l'angolo tra i due vettori è indefinito.

```
15              if np.allclose(A, [0, 0, 0]) or np.allclose(B, [0, 0,
    0]): #degenerate vectore case
16                  return np.array([1, 0, 0, 0]) #the identity
    quaternion means no rotation
17
```

In tal caso, la rotazione è nulla e viene restituito il quaternione che indica zero gradi.

- **Angolo maggiore di 180 gradi**: Nel teorema che definisce la formula del prodotto vettoriale, (Sezione 3.4.2), l'angolo teta è compreso tra 0 e 180 gradi. Tuttavia, nel codice implementato viene utilizzata la funzione di Numpy

```
18          np.cross(v1, v2)
19
```

, la quale se l'angolo è maggiore di 180 gradi, adopera la regola della mano destra per determinare la direzione di rotazione più corta. Per ruotare attorno all'arco più lungo, invece, è sufficiente invertire la sua direzione di rotazione.

Essendo l'avatar un oggetto complesso, le rotazioni relative a un arto che rappresenta il parent si sono dovute gestire diversamente da quelle figlie. Infatti, il genitore quando ruota influenza anche il sistema di coordinate degli arti situati in un livello di gerarchia più basso, i quali non saranno più allineati con i dati di MotionBert. Per tale motivo, prima di eseguire la rotazione ai figli, si è dovuto procedere andando a rimuovere quella del padre. Per effettuare ciò, si è applicata, in primo luogo, la rotazione inversa del genitore e, successivamente, quella calcolata.

```
20      for i in range(0, len(coordinates[0]), 4):
21          child = np.array([coordinates[j][i], coordinates[j][i + 1],
    coordinates[j][i + 2], coordinates[j][i + 3]], dtype=float)
22          parent = np.array([parents_coordinates[j][i], parents_coordinates[
    j][i + 1], parents_coordinates[j][i + 2], parents_coordinates[j][i +
    3]], dtype=float)
23          child = normalize_quaternion(child)
24          parent = normalize_quaternion(parent)
25
26              # Remove parent rotation from child
27          child = R.from_quat(child, scalar_first=True) * R.from_quat(parent
    , scalar_first=True).inv()
28          child = child.as_quat(scalar_first=True)
```

```
29
30              # Update the parts array with the new child rotation
31          coordinates[j][i:i + 4] = np.round(child, 3)
```

4.2 Sistemi di coordinate left-handed e right-handed

Un sistema di coordinate con gli assi x, y, z può essere *left-handed* o *right-handed* (Figura 4.1). Per determinarlo si può usare la regola della mano destra, per sistemi *right-handed*, o sinistra per quelli *left-handed*: si allinea il pollice con l'asse x positivo, l'indice con l'asse y positivo e il medio con l'asse z positivo. Nel caso si usi la mano destra, le rotazioni sono positive in senso antiorario, per la sinistra in senso orario. Inoltre, i sistemi *right-handed* sono i più utilizzati in grafica 3D.

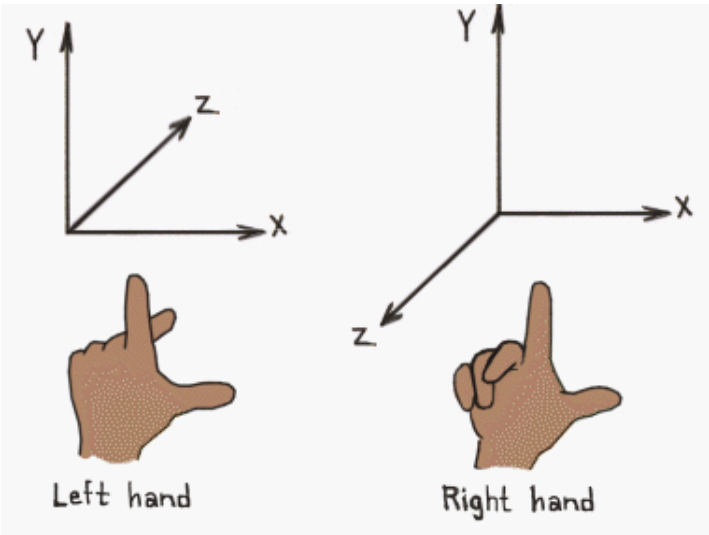


Figure 4.1: Sistemi Right-handed e Left-handed, illustrazione di Learn OpenGL ES (2012)

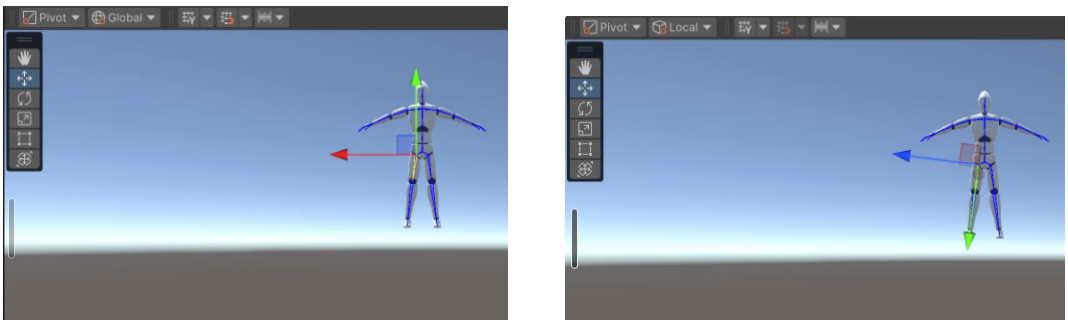
per quanto riguarda il progetto, il framework MotionBert è basato su un sistema di coordinate *right-handed*, mentre Unity è *left-handed*, quindi è stato necessario applicare una conversione ai quaternioni:

```
Quaternion actualRotation = new Quaternion(-(float) allFrames[frameIndex
    ][1], -(float) allFrames[frameIndex][3], -(float) allFrames[frameIndex
    ][2], (float) allFrames[frameIndex][0]);
```

Il codice mostra l'inversione di segno per le componenti x, y, z, ma non per la w e un cambio di posizione tra la componente y e z.

4.3 Trasformazioni locali e globali

In Unity una trasformazione, come ad esempio una rotazione, può essere applicata sia globalmente che localmente. Si noti in Figura 4.2 che il sistema globale, foto a sinistra, coincide



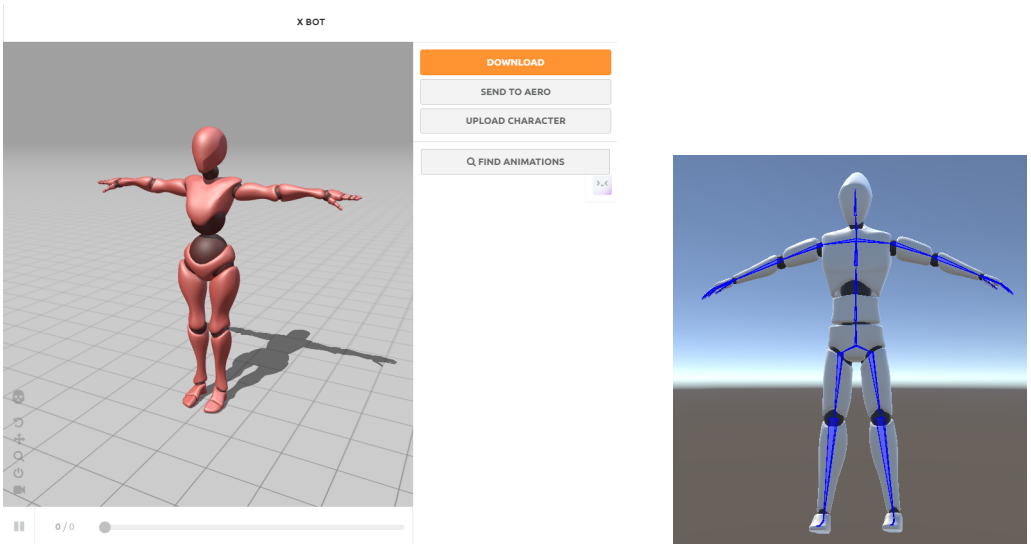
(a) Sistema globale dell'avatar (b) Sistema locale dell'avatar

Figure 4.2: Sistema globale e locale dell'avatar

con quello di Unity con l'asse y positivo verso l'alto, mentre quello locale, a destra, è relativo alla coscia destra e ha l'asse y positivo verso il basso. Il sistema locale è senz'altro utile per applicare le trasformazioni in relatizione al genitore, ma in questo progetto si vuole evitare questo comportamento in quanto i dati ottenuti da MotionBert sono globali e, perciò, la rotazione calcolata è globale. Le posizioni di MotionBert non sono locali in quanto non sono espresse in funzione della posizione del genitore.

4.3.1 Avatar 3D

Come riporta il Paragrafo 3.2 del Capitolo 3, si è scelto di usare i due avatar X Bot e Banana Man in quanto hanno una struttura corporea simile a quella dello scheletro ottenuto con il framework MotionBERT (Figura 4.3).



(a) Avatar X Bot (b) Avatar BananaMan

Figure 4.3: I due avatar utilizzati nello sviluppo

4.3.2 Configurazione visore in 3DOF

L'applicazione è stata eseguita su un Oculus Quest 1 per verificarne il funzionamento, il quale è avvenuto con successo. Sebbene la modalità head tracking permetta di tracciare le rotazioni della testa, non rileva gli spostamenti. Pertanto, per avere un'esperienza più coinvolgente e per potersi muovere agilmente nel mondo virtuale, in futuro, si prevede la configurazione della modalità Positional tracking o sei gradi di libertà.

4.3.3 Assunzione delle posizioni corrette

Per controllare che le pose assunte dall'avatar siano coerenti con il video, è stata di grande aiuto la libreria grafica Matplotlib (creata da The Matplotlib development team (n.d.)), la quale permette di visualizzare interazioni e animazioni.

In Figura 4.4 è possibile osservare, in un sistema in 3D, i punti delle posizioni delle giunture. Il programma scritto con questa libreria non solo permette di ingrandire, rimpicciolire e cambiare punto di vista del sistema, ma anche di selezionare specifici frame dell'animazione per avere un'idea chiara sulla posa del ballerino in quel momento.

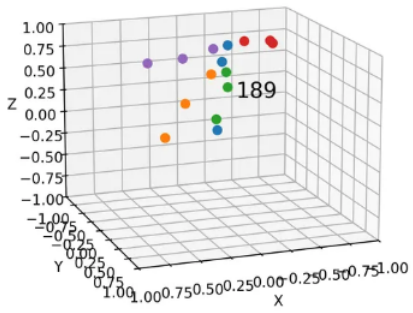


Figure 4.4: Visualizzazione del frame 189 dell'animazione creata usando Matplotlib.

4.3.4 Calcolo delle rotazioni delle giunture

Il calcolo delle rotazioni viene eseguito con degli script in Python, dei quali se ne sono riportati i dettagli nel Capitolo 4. Per assicurarsi della correttezza dei risultati, si è utilizzato lo strumento Quaternions Online di Daniel Gallenberger (n.d.), il quale permette di visualizzare i quaternioni e di ottenere il valore corrispondente in angoli di Eulero (Figura 4.5).

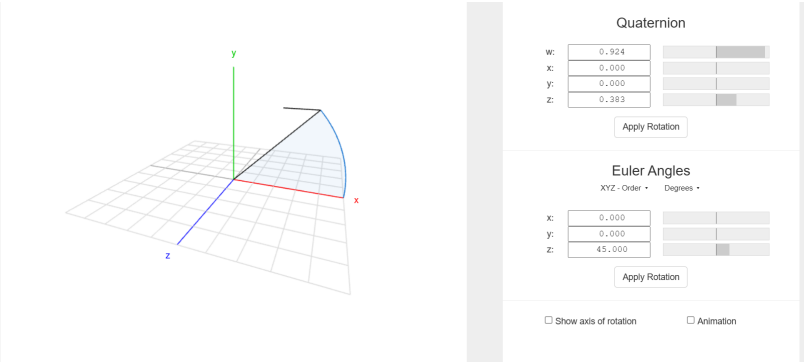


Figure 4.5: Quaternion Online

4.3.5 Menù

Per l'applicazione è stato deciso di realizzare un menù iniziale (Figura 4.6) che appare al momento dell'avvio. Esso mostra un'interfaccia dalla quale è possibile selezionare, tramite dei bottoni, la coreografia che si vuole osservare. Infatti, cliccando su uno dei pulsanti, si avvierà un'altra scena nella quale è presente l'avatar che eseguirà i movimenti corrispondenti al ballo scelto (Figura 4.7).

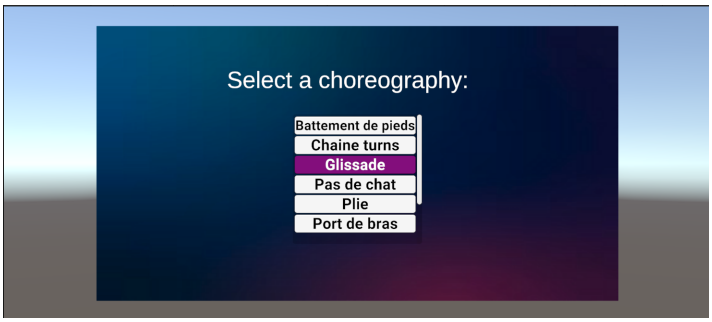


Figure 4.6: Menù

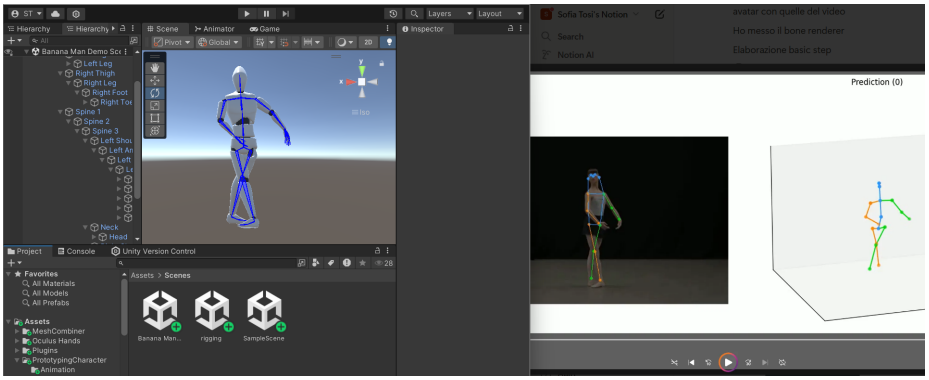


Figure 4.7: Avatar che esegue la coreografia

Durante la realizzazione del menù sono state tenute in considerazione le linee guida sull'accessibilità: i colori scelti per il testo, lo sfondo e gli elementi grafici sono stati scelti in modo da garantire un contrasto sufficiente per permettere anche a persone con difficoltà visive di leggere con facilità. Si è utilizzato lo strumento Contrast Checker (Utah State University (2024)), dal quale è emerso un livello AAA secondo le linee guida standard WCAG.

4.4 Animate3D

Infine, dopo aver svolto il lavoro descritto precedentemente, si è deciso di utilizzare e confrontare il progetto con il framework di animazione dei modelli tridimensionali Animate3D (Yanqin Jiang, Chaohui Yu, Chenjie Cao, Fan Wang, Weiming Hu, Jin Gao (2024)). Questo software ha un modello diffusion MV-VDM che, allenato sul database di grandi dimensioni MV-Video che fornisce video multivista, è in grado di generare dati da del rumore utilizzando video di oggetti 3D ripresi da più angolazioni. Sfruttando MV-VDM e la tecnica 4D score distillation sampling, il framework riesce a generare le animazioni. In particolare, questa tecnica tiene conto della quarta dimensione, quella temporale, dell'evoluzione delle immagini allo scorrere del tempo, e permette di perfezionare sia l'apparenza che il movimento dell'oggetto. Di seguito si riporta uno schema sulla struttura di Animate3D (Figura 4.8).

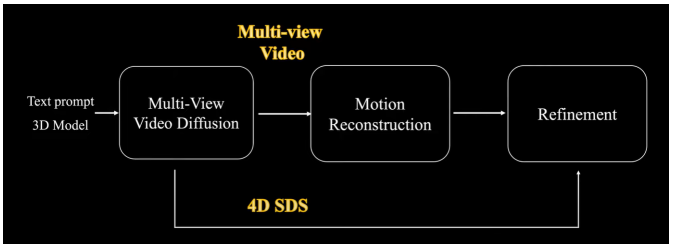


Figure 4.8: Schema Animate3D

Grazie a quest'approccio si ottengono risultati più realistici, più coerenti grazie all'utilizzo di un vasto dataset e animazioni fluide mediante interpolazioni avanzate.

Invece, il metodo sviluppato in questo progetto permette di avere un maggior controllo, perchè si definiscono, con precisione, rotazioni e posizioni dell'avatar, garentendo corrispondenza con i dati. Inoltre, si può correggere ogni frame senza dipendere da un sistema automatico, accelerare e decelerare l'animazione e si può adattare il movimento a qualsiasi esigenza, con variazioni di stile.

4.5 Feedback con gli utenti

Sono stati coinvolti sei utenti che non avevano esperienza con la realtà virtuale e sono stati raccolti dei riscontri. In primo luogo, è stato chiesto ad ognuno di loro di utilizzare l'applicazione senza assegnare istruzioni sul funzionamento per verificarne l'intuitività. Ne è risultato che ogni partecipante è stato in grado di utilizzarla senza difficoltà. Nessuno ha avvertito il bisogno di informazioni testuali che notificassero il caricamento della scena con l'avatar, in quanto il tempo di caricamento è risultato, in tutti i sei utilizzi, minore di trenta secondi. Due degli utenti hanno segnalato la mancanza di un pulsante per poter tornare alla schermata iniziale, pertanto, è stato inserito nella lista delle funzionalità da aggiungere in futuro. I movimenti dell'avatar risultano fluidi, ma è emersa la richiesta di poter regolare la velocità di esecuzione della coreografia. Infine, è stato apprezzato il riavvio automatico del ballo.

Capitolo 5

Conclusioni e sviluppi futuri

La tesi mostra come sia possibile riprodurre nel mondo virtuale i movimenti di una persona provenienti da video 2D. Grazie a questo sistema, è possibile cambiare il modo di fruizione dei contenuti multimediali, rendendoli più coinvolgenti.

In particolare, un ruolo chiave nella realizzazione della soluzione l'hanno svolto le rotazioni. Infatti, l'applicazione diretta delle coordinate posizionali non permetteva di ottenere un risultato adeguato: ogni giuntura dell'avatar si staccava dal corpo ottenendo un effetto innaturale. Un altro aspetto, a cui bisogna prestare attenzione, è come sono orientati gli assi. I sistemi, difatti, possono essere *left-handed* o *right-handed*. I dati provenienti da *MotionBert* sono *right-handed*, mentre *Unity* è *left-handed*. Per questo motivo la y e la z sono state invertite.

Tuttavia, MotionBert ha delle limitazioni, infatti, il framework non fornisce alcune delle coordinate, come quelle delle mani o dei piedi. Questi dati, nella maggior parte delle attività, sono importanti. Si pensi ad un ballerino, è fondamentale conoscere se appoggia i piedi a terra o si alza sulle punte. Inoltre, non è stato possibile considerare, con i dati a disposizione, l'orientamento delle articolazioni. Queste informazioni possono essere ottenute applicando sul corpo della persona dei sensori indossabili.

È proprio questa restrizione che può stimolare un ulteriore sviluppo. Infatti, in futuro, lo stesso sistema può essere integrato con dati provenienti da altri dispositivi come *smartwatch* per il polso, *earable* per le orecchie o *smartring* per le dita. Inoltre, il contenuto può essere arricchito con informazioni supplementari o curiosità. Può essere trasformato in un sistema di apprendimento; nel caso della danza si possono aggiungere tutorial per insegnare alcune notazioni delle coreografie come quella di Stepanov o di Laban. Per coinvolgere ancora di più l'utente, possono essere inserite ambientazioni suggestive o costumi realistici. Infine, due funzionalità importanti possono essere un regolatore della velocità, per dare la possibilità di osservare i movimenti in modo adeguato alla persona che sta usando l'applicazione, oppure la sostituzione di video preregistrati con una telecamera che inoltri gli spostamenti in tempo reale.

Ci si augura, con questo progetto, di aver fornito uno stimolo allo sviluppo di applicazioni sulla realtà virtuale. Specificatamente, la capacità di riprodurre movimenti bidimensionali in uno spazio 3D presenta molte opportunità, ancora in fase di scoperta e ricerca in diversi ambiti, dall'educazione, all'arte, alla scienza.

Ringraziamenti

Questa tesi non rappresenta per me un semplice progetto svolto per concludere il mio percorso di studi all' università, ma è un ricordo della bellissima avventura che ho vissuto qui a Cambridge, grazie alla quale ho avuto la possibilità di conoscere gente nuova, scoprire culture diverse e migliorare con l'inglese.

A livello accademico, ho potuto vivere un'esperienza splendida grazie al mio supervisor e amico Mathias che, sempre gentile e disponibile per chiarimenti e dubbi, mi ha guidata in questo percorso, durante il quale mi ha insegnato il metodo su come affrontare le sfide e risolverle. A volte, anche una semplice chiacchierata con qualcuno può aiutare a trovare l'idea necessaria.

Questi quattro mesi in Inghilterra sono stati possibili grazie alla Prof.ssa Cecilia, che mi ha accolta con grande ospitalità e gentilezza. Ringrazio anche il mio relatore Prof. Gustavo Marfia e il Dott. Pasquale Cascarano.

Un grazie speciale a Matte, per supportarmi sempre anche quando le mie decisioni implicano una grande distanza. Grazie per aver collaborato con me in alcuni progetti e per avermi fornito l'utilissimo template. Grazie di cuore a Gaia, una persona importante sulla quale posso sempre contare e Giulia, per i continui aggiornamenti, anche da distanti ho sempre sentito la tua presenza.

Infine, ringrazio la mia famiglia per avere finanziato questa bellissima esperienza.

Bibliografia

Daisy Dai (2024), 'Pose tracking methods: Outside-in vs inside-out'.
URL: *https://pimax.com/blogs/blogs/pose-tracking-methods-outside-in-vs-inside-out-tracking-in-vr?*

Daniel Gallenberger (n.d.), 'Quaternions online'.
URL: *https://quaternions.online/*

Eric Krokos, Catherine Plaisant, Amitabh Varshney (2018), 'Virtual memory palaces: immersion aids recall'.
URL: *https://obj.umiacs.umd.edu/virtual_reality_study/10.1007-s10055-018-0346-3.pdf*

Gold World (2022), '3dof e 6dof: che differenza c'è?'.
URL: *https://goldworld.it/248235/tech/vr/3dof-e-6dof-che-differenza-ce/*

Hanna Pamuła (n.d.), 'Calcolatore per l'angolo tra due vettori'.
URL: *https://www.omnicalculator.com/it/matematica/angolo-tra-due-vettori*

Harvey Isitt (2024), 'Optimise your visuals: Refresh rate explained'.
URL: *https://www.whatsthebest.co.uk/tech/electronics/refresh-rate/*

Izard S. G., Juanes Méndez J. A., Palomera P. R. (2017), 'Virtual reality educational tool for human anatomy'.
URL: *https://pubmed.ncbi.nlm.nih.gov/28326490/*

James Stewart (2015), 'Calculus: Early transcendentals'.
URL: *https://patemath.weebly.com/uploads/5/2/5/8/52589185/james-stewart-calculus-early-transcendentals-7th-edition-2012-1-20ng7to-1ck11on.pdf*

Learn OpenGL ES (2012), 'Understanding opengl's matrices'.
URL: *https://www.learnopengles.com/tag/right-handed-coordinate-system/*

Matthes, D. and Drakopoulos, V. (2022), 'Line clipping in 2d: Overview, techniques and algorithms', *Journal of imaging* **8**(10), 286.
URL: *https://europepmc.org/articles/PMC9605407*

Mechatech (n.d.), 'How do common virtual reality tracking systems work?'.
URL: *https://www.mechatech.co.uk/journal/how-do-common-virtual-reality-tracking-systems-work*

Mixamo (n.d.), 'Mixamo'.
URL: *https://www.mixamo.com*

Open MMLab (2024), 'Mmpose'.
URL: *https://github.com/open-mmlab/mmpose*

Patrick R (2023), 'Standalone vr: What you need to know about all-in-one headsets'.
URL: *https://blog.vive.com/us/all-in-one-vr-what-you-need-to-know/*

Peking University and Shanghai AI Laboratory (2023), 'Motionbert: A unified perspective on learning human motion representations'.
URL: *https://arxiv.org/pdf/2210.06551*

Pimax.com (2024), 'What are glare, god rays, and distortion in vr'.
URL: *https://pimax.com/blogs/blogs/what-are-glare-god-rays-and-distortion-in-vr?*

Rallis, I., Langis, A., Georgoulas, I., Voulodimos, A., Doulamis, N. and Doulamis, A. (2018), An embodied learning game using kinect and labanotation for analysis and visualization of dance kinesiology, *in* '2018 10th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)', pp. 1–8.

Rory Brown (n.d.), 'Vr compare'.
URL: *https://vr-compare.com/*

Steam games (n.d.), 'Steamvr tracking'.
URL: *https://partner.steamgames.com/vrlicensing*

The Matplotlib development team (n.d.), 'Matplotlib'.
URL: *https://matplotlib.org/*

Upload VR (2021), ''breakthrough' pancake lenses could bring compact headsets'.
URL: *https://www.uploadvr.com/kopin-all-plastic-pancake-optics/*

Utah State University (2024), 'Contrast checker'.
URL: *https://webaim.org/resources/contrastchecker/*

VR Italia (2023), 'I tipi di lente in vr: pregi e difetti di ogni soluzione'.
URL: *https://www.vr-italia.org/i-tipi-di-lente-in-vr-pregi-e-difetti-di-ogni-soluzione/*

Wikipedia (2024*a*), 'Gimbal lock'.
URL: *https://en.wikipedia.org/wiki/Gimbal_lock*

Wikipedia (2024*b*), 'Oculus rift cv1'.
URL: *https://en.wikipedia.org/wiki/Oculus_Rift_CV1*

Wikipedia (2024*c*), 'Screen-door effect'.
URL: *https://en.wikipedia.org/wiki/Screen-door_effect*

Yanqin Jiang, Chaohui Yu, Chenjie Cao, Fan Wang, Weiming Hu, Jin Gao (2024), 'Animate 3d'.
URL: *https://animate3d.github.io/*

Zhen, W. and Luan, L. (2021), 'Physical world to virtual reality – motion capture technology in dance creation', *Journal of Physics: Conference Series* **1828**(1), 012097.
URL: *https://dx.doi.org/10.1088/1742-6596/1828/1/012097*