

POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria del Cinema e dei Mezzi di
Comunicazione



**Politecnico
di Torino**

Tesi di Laurea Magistrale

Low Cost Automation in realtà virtuale: confronto tra diversi sistemi di interazione

Relatore: Andrea SANNA

Correlatore: Federico MANURI

Federico DECATALDO

Matricola n. 290333

Anno Accademico 2022/2023

Abstract

Fin dalla sua nascita la Realtà Virtuale (RV) è stata, ed è tutt'ora, oggetto di un processo di sviluppo e innovazione che ha permesso di superarne gran parte dei limiti iniziali rendendo l'esperienza dell'utente sempre più realistica e/o coinvolgente. Oltre allo sviluppo tecnologico è stata interessata anche da un ampliamento degli ambiti di utilizzo: se, inizialmente, era relegata prevalentemente al settore dell'entertainment, con il tempo ha iniziato ad essere impiegata in ambiti differenti quali il training, la medicina, la psicologia, l'educazione e molti altri.

Un settore che ha recentemente scoperto i vantaggi derivanti dalla Realtà Virtuale è quello dell'Industria 4.0, termine utilizzato per la prima volta nel 2011 durante una fiera ad Hannover, in Germania. Per Industria 4.0 si intende il risultato della quarta rivoluzione industriale il cui tratto caratteristico consiste nel rendere le realtà industriali sempre più automatizzate, connesse e flessibili.

La Realtà Virtuale può svolgere un ruolo chiave in attività come la prototipazione; la prototipazione è soggetta ad errori, passi indietro e rivalutazioni che, nella realtà, comportano un dispendio di tempo e risorse economiche. In un mondo virtuale, invece, si avrebbe molta più flessibilità: si potrebbero modificare forma, dimensione, materiale, colore e altre caratteristiche del prototipo senza che questo si traduca in spreco di materiale (e, quindi, di risorse economiche) e di tempo (le modifiche si possono effettuare in tempo reale durante la simulazione/prototipazione).

L'oggetto di questa tesi è il processo di prototipazione virtuale di sistemi di Low Cost Automation (LCA) durante il quale l'utente indossa un casco per la Realtà Virtuale (ad esempio, un HTC Vive Pro) e può interagire con oggetti virtuali tramite opportune interfacce. La simulazione è stata sviluppata in Unity3D.

Inoltre, lo scopo del presente lavoro è quello di valutare differenti strumenti di interazione nel processo di prototipazione; nello specifico HTC Vive Controller e MANUS Gloves, guanti che tramite i sensori presenti all'interno riescono a ricostruire la posa assunta dalla mano dell'utente. La mappatura delle azioni sui tasti dei controller diventa una mappatura sulle pose della mano: se questa è chiusa a pugno si può afferrare un oggetto, per selezionare voci da menu è sufficiente indicarle con l'indice. Sono stati condotti dei test su un campione di utenti a cui sono state fatte provare entrambe le versioni e a cui è stato sottoposto un questionario per valutare l'usabilità delle interfacce e ottenere un riscontro su eventuali preferenze di un sistema di interazione sull'altro.

Lo stesso applicativo basato sui MANUS Gloves è stato adattato per funzionare all'interno di un CAVE (Cave Automatic Virtual Environment), un sistema composto da quattro schermi proiettati (un frontale, due laterali e uno che funge da pavimento) su cui

viene ricostruito l'ambiente virtuale in 2D la cui visione tridimensionale (che permette all'utente di sperimentare un senso di immersione nell'ambiente virtuale) viene resa con l'utilizzo di appositi occhiali 3D.

Risultati preliminari derivanti dai test condotti sull'usabilità delle interfacce conferisce al sistema basato sui MANUS Gloves un punteggio maggiore: interagire con gli oggetti sfruttando le pose delle mani, infatti, è stato valutato dagli utenti più intuitivo, più fedele alla realtà rispetto a mappare le diverse funzionalità sui pulsanti degli HTC Vive Controller.

Indice

Introduzione.....	1
1.1 La Realtà Virtuale (RV).....	1
1.2 Tipi di realtà virtuale.....	1
1.3 Immersione, interazione, presenza.....	2
1.4 Ambiti di applicazione della VR.....	4
1.5 Industria 4.0	5
1.6 Tecnologia 4.0 - Le Smart Technologies	7
1.7 Competence Center	8
1.8 Obiettivo della tesi	9
1.9 LCA	9
1.10 Organizzazione dei capitoli	10
 Capitolo 2 – Realtà Virtuale: concetti generali	13
2.1 Storia della Realtà Virtuale	13
2.1.1 Panoramica dei dispositivi dagli anni ‘60 ad oggi	13
2.1.2 Reality-Virtuality Continuum	17
2.3 Sistemi di interazione per la VR e loro funzionamento	22
2.3.1 Tracker meccanici	24
2.3.2 Tracker magnetici.....	24
2.3.3 Tracker acustici	24
2.3.4 Tracker ottici	25
2.3.5 Interfacce inerziali.....	25
2.3.6 Tracker ibridi.....	26
2.3.7 Motion capture	27
2.3.8 Interfacce tattili	27
2.3.9 Interfacce non convenzionali	30
2.3.10 Interfacce tangibili.....	31
2.4 Interfacce utente per la VR	31

2.4.1 Navigazione.....	32
2.4.2 Selezione e manipolazione	34
2.4.3 Controllo del sistema.....	35
 Capitolo 3 – Stato dell’arte	37
3.1 Realtà Virtuale e prototipazione	37
3.1.1 IKEA VR Experience.....	37
3.1.2 ShapesXR.....	38
3.1.3 #NEXTGen (BMW Group Platform).....	39
3.2 Hand Tracking in VR.....	40
3.2.1 Auto Hand – VR Physics Interaction	40
3.2.2 Meta Presence Platform	41
3.2.3 Leap Motion	43
3.2.4 Confronto tra controller interaction e hand interaction	44
 Capitolo 4 – Tecnologie utilizzate	51
4.1 Hype cycle di Gartner	51
4.2 Architetture hardware e software.....	52
4.3 Hardware.....	54
4.3.1 HTC Vive Pro.....	55
4.3.2 HTC Vive Controller.....	56
4.3.3 Vive Trackers 2.0	58
4.3.4 MANUS Prime II – Xsens Edition.....	59
4.3.5 CAVE (Cave Automatic Virtual Reality)	61
4.4 Software e Plugin.....	63
4.4.1 Unity.....	63
4.4.2 Visual Studio	66
4.4.3 SteamVR	66
4.4.4 MANUS Core e MANUS Dashboard	69
4.4.5 MANUS Unity Plugin.....	71
4.4.6 Motive	72
4.4.7 Mirror Unity Library	75

Capitolo 5 – Design e sviluppo	77
5.1 Progetto per HMD	77
5.1.1 Controllo del sistema.....	78
5.1.2 Select Interaction System	82
5.1.3 Info Panel	82
5.1.4 Differenze nel Player.....	91
5.1.5 Differenze di script per rendere gli oggetti interagibili.....	94
5.1.6 Gestione delle pose della mano con i MANUS Prime II	95
5.1.7 Gestione menu.....	97
5.2 Progetto per il CAVE.....	100
 Capitolo 6 – Test e risultati	107
6.1 Questionario SUS (System Usability Scale).....	109
6.2 Questionario NASA – TLX	110
6.3 Risultati ottenuti.....	111
 Capitolo 7 – Conclusioni e sviluppi futuri	115
 Appendice.....	117
 Bibliografia.....	123

Lista delle figure

1.1: postazione per Realtà Virtuale Desktop	2
1.2: Realtà Virtuale Immersiva.....	2
1.3: elementi che costituiscono la “presenza” nella VR	3
1.4: le quattro Rivoluzioni Industriali.....	5
1.5: tecnologie Industria 4.0	6
1.6: esempio di un Low Cost Automation (LCA)	9
2.1: Sensorama, 1992.....	13
2.2: Sayre Glove, primo guanto capace di misurare la flessione delle dita	14
2.3: Aspen Movie Map. Visione basata su foto scattate.....	15
2.4: Aspen Movie Map. Visione basata sul modello 3D pre renderizzato.	15
2.5: Mattel Power Glove.....	15
2.6: Oculus Quest 2.....	16
2.7: HTC Vive Pro.....	17
2.8: HTC Vive XR Elite	17
2.9: PSVR2 con joystick PSVR2 Sense	17
2.10: Reality-Virtuality Continuum di Milgram e Kishino, 1994	18
2.11: esempio della struttura di un CAVE.....	20
2.12: step pipeline rendering grafico	21
2.13: informazioni di posizione e orientamento tracciate.....	23
2.14: tecnologia Outside Looking In (sinistra) e Inside Looking Out (destra)...	25
2.15: Cyber Glove.....	28
2.16: Leap Motion	28
2.17: esempio pipeline di rendering tattile	29
2.18: SandScape sviluppato presso il MIT	31
2.19: guardian system di Oculus Quest 2	33
2.20: esempio di teletrasporto in VR	33
2.21: Virtual Hand	34
2.22: esempio di Raycasting	35

2.23: alcuni esempi di interfacce	36
3.1: IKEA VR Experience	38
3.2: Shapes XR	39
3.3: #NEXTGen, BMW Group Platform.....	40
3.4: Auto Hand in Unity	41
3.5: Meta Presence Platform.....	42
3.6: Leap Motion Controller	43
3.7: metodi di interazione utilizzati: mouse (a sinistra), hand tracking (al centro), controller e stylus (a destra).....	45
3.8: prototipo del guanto utilizzato nello studio presentato.....	47
3.9: HTC Vive utilizzato nella versione HMD dello studio descritto	49
3.10: occhiali 3D utilizzati nella versione per il CAVE dello studio descritto...	49
4.1: Hype Cycle di Gartner	51
4.2: HTC Vive Pro.....	55
4.3: HTC Vive Controller	57
4.4: pulsanti presenti su un HTC Vive Controller	57
4.5: HTC Vive Trackers 2.0	58
4.6: elementi presenti sugli HTC Vive Tracker 2.0.....	59
4.7: MANUS Prime II - Xsens Edition.....	59
4.8: proiettore Barco UDM 4k15 presente nel sistema CAVE utilizzato.....	61
4.9: Volfoni Edge VR utilizzati per la visione 3D all'interno del CAVE.....	61
4.10: OptiTrack PrimeX 22 utilizzate per il tracking di oggetti nel CAVE	62
4.11: interfaccia di Unity versione 2020.3.44f1	64
4.12: pannello SteamVR Input in cui poter definire delle Action da mappare sui pulsanti del controller	67
4.13: Pose Editor del componente SteamVR_Skeleton_Poser.....	69
4.14: interfaccia della MANUS Dashboard.....	70
4.15: interfaccia del software Motive	74
4.16: Calibration Wand.....	74
4.17: Calibration Square	75

5.1: gerarchia oggetti presenti nella scena Unity	78
5.2: pannello per la creazione e il salvataggio di azioni da mappare sui pulsanti dei controller	79
5.3: interfaccia che permette di associare ai pulsanti dei controller le Action definite in Unity	79
5.4: Open Hand Pose	80
5.5: Grab Pose.....	81
5.6: Pointing Pose	81
5.7: pannello per scegliere il sistema di interazione	82
5.8: gerarchia degli elementi che compongono il GameObject Info Panel	83
5.9: introduzione tutorial per sistema basato sugli HTC Vive Controller	83
5.10: introduzione tutorial per sistema basato sui MANUS Prime II.....	83
5.11: lista delle funzioni applicabili su oggetti singoli e/o composti.....	84
5.12: pannello con riproduzione dei controller e relativi pulsanti, visione dall'editor di Unity	84
5.13: pannello dei controller mentre viene premuto il pulsante Touch, visione dall'editor di Unity	84
5.14: pannello con riproduzione dei controller e relativi pulsanti, visione dall'applicativo in esecuzione.....	85
5.15: pannello dei controller mentre viene premuto il pulsante Touch, visione dall'applicativo in esecuzione	85
5.16: menu principale contenente gli oggetti istanziabili, visione dall'editor di Unity	85
5.17: apertura del menu principale, visione dall'applicativo in esecuzione.....	86
5.18: pannello con le informazioni riguardanti un oggetto singolo e le funzioni su di esso applicabili, visione dall'editor di Unity	87
5.19: pannello con le informazioni riguardanti un tubo singolo e le funzioni su di esso applicabili, visione dall'applicativo in esecuzione.....	87
5.20: pannello con le informazioni riguardanti un oggetto composto e le funzioni su di esso applicabili, visione dall'editor di Unity	88
5.21: pannello con le informazioni riguardanti un LCARuota e le funzioni su di esso applicabili, visione dall'applicativo in esecuzione.....	89

5.22: pannello contenente l'Highlighter con cui innescare il comportamento di determinate funzioni, visione dall'editor di Unity	90
5.23: pannello contenente l'Highlighter con cui innescare il comportamento della funzione SetDimension, visione dall'applicativo in esecuzione	90
5.24: struttura del Player utilizzato per gli HTC Vive Controller	91
5.25: struttura del GameObject SteamVRObjects	91
5.26: struttura del Player utilizzato nel caso di impiego dei MANUS Prime II .	92
5.27: struttura del GameObject ManusHand_L (analoga a ManusHand_R).....	93
5.28: pannello per oggetti composti legati a strutture nel caso di utilizzo dei guanti MANUS, visione dall'editor di Unity. La versione con i Vive Controller è mostrata in fig. 5.16	99
5.29: pannello di un LCARuota legato ad una struttura nel caso di utilizzo dei guanti MANUS, visione dall'applicativo in esecuzione	99
5.30: pannello mostrato quando si vuole spostare verticalmente un elemento della struttura complessa utilizzando i guanti MANUS	100
5.31: struttura su cui sono stati inseriti i marker per configurarli all'interno di Motive con il fine di tracciare la posizione dei guanti MANUS	101
5.32: gerarchia degli elementi che costituiscono il progetto Unity per CAVE	102
5.33: struttura del prefab Cave System.....	102
5.34: struttura dell'oggetto Cave	104
 6.1: LCA che l'utente deve riprodurre in fase di testing.....	108
6.2: rappresentazione delle risposte al SUS per il sistema basato sugli HTC Vive Controller.....	113
6.3: rappresentazione delle risposte al SUS per il sistema basato sui MANUS Prime II	113
6.4: confronto risultati NASA-TLX per entrambi i sistemi testati	114

Elenco delle tabelle

4.1: Architettura del sistema sviluppato per HMD	52
4.2: Architettura del sistema sviluppato per il CAVE	54
6.1: Suddivisione dei punteggi test SUS.....	109
6.2: Tempi di prototipazione dell’LCA per entrambi i sistemi di interazione..	111
6.3: Risultati SUS del sistema con gli HTC Vive Controller	112
6.4: Risultati SUS del sistema con i MANUS Prime II.....	112

Acronimi

2D: Bi-dimensional, Bidimensionale

3D: Three-dimensional, Tridimensionale

AR: Augmented Reality, Realtà Aumentata

AV: Augmented Virtuality, Virtualità Aumentata

CAVE: Cave Automatic Virtual Environment

CIM 4.0: Competence Industry Manufacturing 4.0

CPU: Central Process Unity, Unità di elaborazione centrale

DLP: Digital Light Processing, Elaborazione digitale della luce

DoF: Degrees of Freedom, Gradi di libertà

FOV: Field Of View, Campo visivo

FPS: Frame Per-Second, Fotogrammi per secondo

GPU: Graphics Process Unit, Unità di elaborazione grafica

HMD: Head-Mounted Display

HMI: riferito a informazioni visibili in sovrimpressione

IoT: Internet of Things

IPD: Inter-Pupillary Distance, Distanza interpupillare

IR: Infra Red, radiazione infrarossa

IT: Information Technology, Tecnologia dell'Informazione

LCA: Low Cost Automation

LCD: Liquid Crystal Display, Schermo a cristalli liquidi

LED: Light Emission Diodes, Diodi ad emissione di luce

LTS: Long Term Support, Supporto a lungo termine (riferito a versioni di un software)

LVD: Large Volume Display, Display di grandi dimensioni

MR: Mixed Reality, Realtà Mista

OLED: Organic Light-Emitting Diode, Diodo organico a emissione di luce

OT: Operational Technology, Tecnologia Operativa

RA: Realtà Aumentata

RV: Realtà Virtuale

RVI: Realtà Virtuale Immersiva

SDK: Software Development Kit

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

UI: User Interface, Interfaccia utente

VR: Virtual Reality, Realtà Virtuale

VRPN: Virtual Reality Peripheral Network

Glossario

Big Data: nel mondo della statistica e dell'informatica, con il termine Big Data si fa riferimento a raccolte di dati estese sia in termini di volume che di velocità e varietà.

Bounding box: si tratta di un riquadro che viene utilizzato per identificare nello spazio un oggetto e tramite il quale è possibile riconoscere e trattare fenomeni quali le collisioni con altri elementi.

Cloud Computing: fa riferimento a servizi di un fornitore la cui erogazione, su richiesta, avviene tramite rete internet. Tali servizi possono essere inerenti l'elaborazione, la trasmissione e l'archiviazione di dati.

Degrees of Freedom (DoF): fa riferimento alla libertà di movimento che ha un corpo rigido nello spazio. I movimenti possono riguardare principalmente la traslazione lungo gli assi X, Y e Z e la rotazione attorno ai medesimi assi.

Esposizione di una camera: indica la quantità di luce che viene catturata dal sensore di una camera. È influenzata da tre fattori: tempo di scatto, apertura del diaframma e sensibilità del sensore (ISO).

Field Of View (FOV): rappresenta la porzione di mondo visibile attraverso gli occhi nel momento in cui la testa resta immobile. Nell'uomo il FOV è pari a 120° in verticale e a 170° in orizzontale se si considera un solo occhio; nel momento in cui vengono combinate le informazioni ricavate da entrambi gli occhi il FOV orizzontale diventa pari a 220° . Lo stesso concetto lo si può applicare ai sistemi di visione utilizzati nel mondo della VR dove con FOV si identifica la porzione di mondo virtuale visualizzabile in contemporanea.

Frame Rate (fps): fa riferimento alla frequenza con cui le immagini vengono catturate per produrre un video o alla frequenza con cui vengono riprodotte le immagini che compongono un filmato. Per far sì che l'occhio umano percepisca un video come percepirebbe la realtà è necessario che venga riprodotto con un frame rate di 24/30 fps.

Inter-Pupillary Distance (IPD): rappresenta la distanza tra i centri delle pupille degli occhi. Varia da persona a persona e si misura in millimetri.

Machine Learning: con questo termine ci si riferisce a quei sistemi che apprendono e migliorano le proprie performance basandosi sui dati che acquisiscono e utilizzano.

Refresh Rate: indica il numero di volta al secondo con cui un'immagine viene ricreata su un display.

Rendering: si tratta del processo tramite cui viene generata un'immagine partendo dalla descrizione matematica di una scena tridimensionale attraverso l'utilizzo di determinati algoritmi.

Shadowing: utilizzato nel campo della VR, fa riferimento a quelle situazioni in cui i dati dei tracker non possono essere recuperati.

Stereoscopia: si tratta della percezione di tridimensionalità degli oggetti. È dovuta al fatto che gli occhi catturano due immagini leggermente diverse del mondo che ci circonda; queste immagini vengono elaborate dal cervello permettendo la visione tridimensionale. Il senso di profondità viene, inoltre, favorito, da elementi quali posizione reciproca tra oggetti, ombre e occlusioni.

Virtual Embodiment: fa sì che un utente senta di essere fisicamente presente all'interno di un mondo virtuale. È generato dall'illusione di luogo (ovvero la sensazione di essere in un luogo pur non essendoci fisicamente) e dall'illusione di plausibilità (credibilità di ciò che sta avvenendo nel mondo virtuale).

Introduzione

1.1 La Realtà Virtuale (RV)

La Realtà Virtuale, o Virtual Reality (VR) permette, attraverso l'utilizzo di uno o più computer e di apparecchiature quali visori, guanti, controller e auricolari, di creare un ambiente completamente digitale che simuli la realtà in maniera il più realistica possibile, così come viene percepita dai nostri sensi. Con l'accesso a contenuti predefiniti, gli utenti possono immergersi in simulazioni tridimensionali e dinamiche all'interno delle quali, grazie ai progressi della tecnologia, è possibile muoversi in tempo reale e interagire con gli oggetti in essi contenuti [1].

Questo, tuttavia, comporta diverse difficoltà come la necessità di una potenza di calcolo elevata in quanto deve essere trattata una mole importante di dati: bisogna, infatti, gestire sia i modelli degli oggetti tridimensionali che compongono l'ambiente ma, nel caso in cui all'utente venga data la possibilità di interagire con alcuni di essi, bisogna capire come l'ambiente risulta modificato da queste azioni e progettare gli output corrispondenti all'interno della simulazione (output che possono riguardare diversi sensi).

1.2 Tipi di realtà virtuale

Come detto in precedenza, la caratteristica principale della Realtà Virtuale rispetto agli altri tipi di simulazioni, come la Realtà Aumentata (Augmented Reality, AR), sta nel fatto che la VR utilizza solo grafica di sintesi mentre non comprende alcun elemento appartenente alla realtà. Ci sono, tuttavia, diversi modi di implementarla i cui estremi (mostrati nelle figure 1.1 e 1.2) sono rappresentati da:

1. **Realtà Virtuale Desktop:** si tratta di sistemi di realtà virtuale “da scrivania”, utilizzano un pc, fisso o portatile, e mezzi tramite i quali interagire all'interno del mondo digitale costituiti principalmente da monitor, tastiera, mouse e joystick.
Dal punto di vista dell'immersività, ci sono dei limiti legati a questa implementazione, partendo dal fatto che il monitor attraverso il quale vengono fruite applicazioni di questo tipo è una piccola finestra all'interno del mondo reale, passando per la scarsa mobilità dell'utente.
Nonostante ciò gli strumenti necessari per implementare sistemi di questo tipo sono alla portata di tutti per via dei bassi costi;
2. **Realtà Virtuale Immersiva (RVI):** in questo caso la simulazione coinvolge il maggior numero di canali sensoriali possibili in modo da sostituirsi alla realtà nel miglior modo possibile.

Tra questi il più importante è sicuramente il canale legato alla vista; si cerca, infatti, di coinvolgere il maggior Field Of View (FOV) possibile e per far questo si possono utilizzare:

1. proiettori e schermi di grandi dimensioni anche se in questo caso non verrebbe coinvolta la visione periferica;
2. proiettori che circondino l'utente in modo che il FOV sia coperto al massimo;
3. Head Mounted Display (HMD), caschetti grazie ai quali si ha uno schermo per ogni occhio per coprire quasi del tutto il FOV dell'utente, il quale viene isolato dal mondo esterno.



Figura 1.1: postazione per Realtà Virtuale Desktop. Fonte: urly.it/3t4sr



Figura 1.2: Realtà Virtuale Immersiva. Fonte: urly.it/3t4ss

Ci sono altri aspetti di cui occuparsi come la profondità del mondo poiché la visione umana è stereoscopica; bisogna poi far sì che la scala della rappresentazione sia realistica.

Nel caso della RVI non si ha più l'obbligo di stare fermi davanti alla scrivania: in questo caso, infatti, i dispositivi tramite i quali poter interagire all'interno della simulazione possono essere tracciati nello spazio, in termini di posizione e rotazione, permettendo all'utente di muoversi effettivamente al suo interno. È necessario, quindi, dotare l'utente di dispositivi che si possano utilizzare in diverse posizioni e con i quali poter interagire all'interno dell'ambiente 3D nel modo più naturale possibile.

1.3 Immersione, interazione, presenza

Uno dei concetti più importanti legati alla VR è quello di “presenza”: con questo termine si fa riferimento alla sensazione mentale che fa percepire l'ambiente virtuale come reale, come se ci si trovasse davvero in quel posto. A questo senso di presenza

contribuiscono diversi fattori quali l'accuratezza con cui l'ambiente viene progettato, il tipo di interfacce che si decide di utilizzare per interagire all'interno della simulazione, le modalità di interazione previste e i feedback che il mondo virtuale invia all'utente (i quali dovrebbero interessare il maggior numero di sensi possibile). Ci deve essere, tuttavia, anche una certa predisposizione dell'utente stesso a distaccarsi dal mondo reale.

Non è facile misurare il livello di presenza che si è riusciti a creare poiché le reazioni degli utenti sono soggettive. Gli strumenti utilizzati sono principalmente due: metodi quantitativi che raccolgono informazioni numeriche legate, ad esempio, a dati fisiologici come battito del cuore, sudorazione, tempi di risposta, e metodi qualitativi basati su questionari.

La presenza dell'utente all'interno di una simulazione è direttamente proporzionale con l'immersione e l'interazione dell'utente con l'ambiente virtuale e dalla loro fusione nascono due concetti fondamentali che vanno a costituire il Virtual Embodiment (figura 1.3):

1. l'illusione di luogo, ovvero la sensazione di essere in un luogo pur non essendoci realmente sfruttando una serie di feedback dovuti all'interazione dell'utente con elementi appartenenti a quell'ambiente;
2. l'illusione di plausibilità, ovvero la percezione che ciò che sta avvenendo nell'ambiente virtuale stia accadendo nella realtà, sia reale e credibile.



Figura 1.3: elementi che costituiscono la “presenza” nella VR

Riuscire ad ingannare la mente fino a convincerla di trovarsi in un altro ambiente non è semplice; in particolare è utile tener conto di 3 fattori che sono:

1. qualità delle informazioni sensoriali: le informazioni che vengono inviate ai canali sensoriali devono essere di alto livello, realistiche;
2. mobilità e controllo dei sensori: l'utente deve poter muoversi agevolmente all'interno dell'ambiente e non essere vincolato ad un'area limitata;
3. controllo sull'ambiente: in riferimento al livello di interazione che viene concesso tra ambiente e utente.

Tutti i dati di cui si è parlato fino ad ora sono bidirezionali, vengono scambiati tra utente e ambiente virtuale e viceversa e gestiti attraverso diversi canali sensoriali quali il visivo, l'acustico, l'aptico e l'inerziale.

1.4 Ambiti di applicazione della VR

Ci sono diversi ambiti in cui la RV viene utilizzata, in alcuni da più tempo mentre altri ne hanno scoperto le potenzialità più di recente. I principali sono:

1. il più diffuso, quello dell'intrattenimento e, in particolare, quello dei videogiochi i quali hanno dato una forte spinta tecnologica al settore, anche in termini di riduzione dei costi;
2. l'ambito del training: grazie alla VR è possibile esercitarsi su attività all'interno di ambienti particolari o che potrebbero essere molto costosi nella realtà; è possibile, inoltre, addestrare in sicurezza il personale in situazioni in cui ci sarebbero dei reali pericoli (come evacuazioni in caso di terremoti o esplosioni);
3. l'ambito educativo, permettendo agli studenti di essere maggiormente coinvolti nel processo di apprendimento;
4. l'ambito della medicina in termini di addestramento che può usarsi in modi diversi: è possibile studiare procedure operatorie prima di attuarle su pazienti reali nel caso di operazioni molto complesse, può essere utilizzata per fare diagnosi in caso di procedure di particolare rischio ma anche come supporto nelle operazioni chirurgiche.

Ulteriore campo è quello della riabilitazione di tipo funzionale o motorio per poter anche monitorare i progressi che i pazienti fanno nel corso della terapia per poter personalizzare gli esercizi in base allo stato corrente del paziente;

5. l'ambito della terapia psicologica: molte ansie o paure sono dovute a stimoli dell'ambiente esterno e il meccanismo con cui questi disturbi si protraggono nel tempo consiste nell'evitare situazioni che li provocano. Il trattamento tradizionale consiste nel mettere il paziente gradualmente nella situazione che causa ansia in modo che ci si abitui pian piano.

Tramite la RV è possibile riprodurre gli stimoli ansiogeni in un ambiente controllato da cui è possibile uscire in qualsiasi momento nel caso in cui il paziente non dovesse reggere la situazione; altro aspetto vantaggioso è legato al fatto che il tutto avviene nello studio del terapeuta o a casa del paziente;

6. l'ambito dell'architettura in modo da rendere la rappresentazione interattiva e con la possibilità di andare a modificare gli oggetti presenti essendo questi parametrizzati;
7. l'ambito della valorizzazione del patrimonio culturale poiché permette sia di ricostruire monumenti distrutti o danneggiati per poterli meglio conservare e restaurare, sia di ricreare siti del passato che hanno subito molte modifiche negli

anni, sia per rendere fruibili e accessibili al pubblico dei manufatti o luoghi che non lo sono.

Uno degli ultimi settori ad aver scoperto le potenzialità della RV è quello industriale: è fondamentale per gestire situazioni di progettazione collaborativa nel caso in cui i progettisti dovessero trovarsi in luoghi diversi ma anche nel campo della prototipazione che permette di studiare vantaggi e svantaggi, limiti e opportunità di un prodotto prima che questo venga fisicamente realizzato, risparmiando molto spesso tempo ma soprattutto risorse economiche. Quanto appena descritto è uno dei diversi aspetti che caratterizzano l'Industria 4.0.

1.5 Industria 4.0

L'espressione Industria 4.0 è stata usata per la prima volta alla Fiera di Hannover nel 2011 in Germania e da lì si è arrivati alla presentazione di una serie di raccomandazioni al governo tedesco per permettere la transizione verso questo nuovo paradigma a cura di un gruppo di lavoro con a capo Dais, manager e azionista della Robert Bosch Industrietreuhand KG (RBIK) [2], e Kagermann della Acatech (Accademia tedesca delle Scienze e dell'Ingegneria) [3].

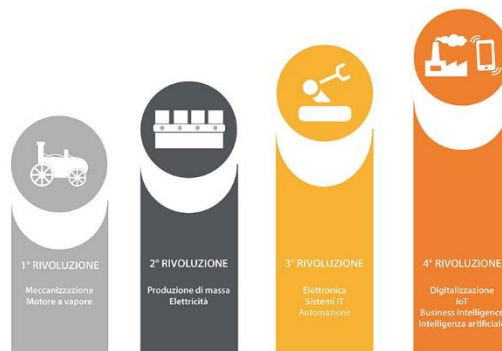


Figura 1.4: le quattro Rivoluzioni Industriali. Fonte: urly.it/3t4t7

Fino ad ora, il mondo occidentale è passato attraverso tre Rivoluzioni Industriali (riportate in figura 1.4):

1. la prima, nel 1784, che ha portato ad una meccanizzazione della produzione sfruttando la potenza dell'acqua e del vapore all'interno della macchina a vapore;
2. la seconda, nel 1870, ha visto l'utilizzo di elettricità, petrolio e l'introduzione del motore a scoppio che hanno reso più efficienti i processi di produzione e hanno portato alla nascita del concetto di produzione di massa;

3. la terza, nel 1970, ha portato all'inizio dell'era digitale con la nascita dell'informatica e l'introduzione nei processi produttivi di sistemi di automazione i quali hanno permesso un aumento della produttività.

E' in corso una Quarta Rivoluzione Industriale la cui data di inizio non è ancora stata stabilita ma che si caratterizza per aver portato i sistemi di produzione ad essere sempre più automatizzati ed interconnessi. Da questo aspetto nasce il concetto di Industria 4.0.

Le nuove tecnologie (mostrate in figura 1.5) hanno un forte impatto su quattro principali aspetti [4]:

1. il primo fa riferimento all'utilizzo dei dati, alla potenza di calcolo e alla connettività, il che va ad interessare ambiti quali i Big Data, il campo delle IoT e il Cloud Computing tramite il quale è possibile conservare i dati acquisiti in maniera centralizzata;
2. il secondo è quello degli analytics, ovvero l'analisi dei dati acquisiti grazie ai quali si possono ricavare delle informazioni utili in ambito produttivo ma anche dati che possono essere utilizzati per processi di Machine Learning;
3. il terzo ambito è quello che fa riferimento al miglioramento dell'interazione uomo – macchina attraverso interfacce con maggiore usabilità e che sfruttano tecnologie innovative come la Realtà Aumentata e la Realtà Virtuale;
4. il quarto ambito è quello che fa da tramite tra mondo digitale e mondo fisico e che ingloba la manifattura additiva, la stampa 3D, la robotica, le comunicazioni, le interazioni tra macchine e le nuove tecnologie grazie alle quali è possibile ottenere un abbassamento dei costi e un'ottimizzazione delle prestazioni.

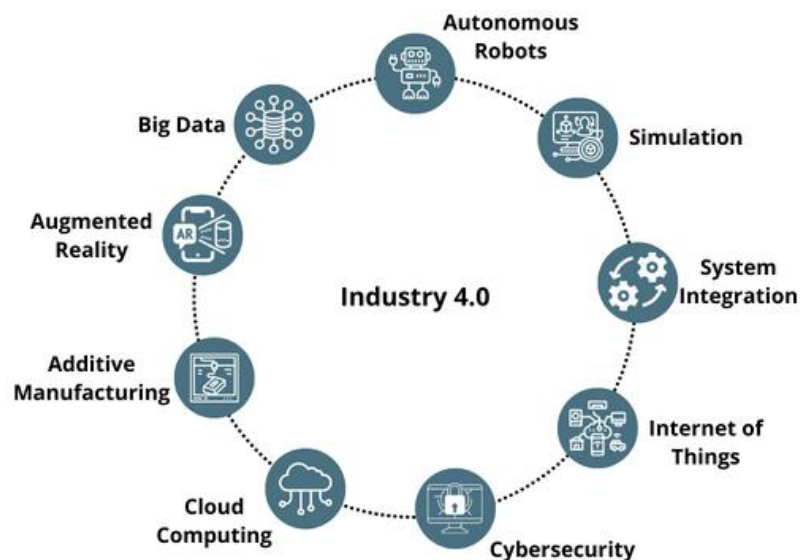


Figura 1.5: tecnologie Industria 4.0. Fonte: urly.it/3t4t9

I vantaggi che derivano dal paradigma dell'Industria 4.0 si possono riassumere come segue [5]:

1. è possibile avere una maggiore flessibilità dei processi di produzione, permettendo di ottenere piccoli lotti ma a costi vantaggiosi che caratterizzano la produzione su larga scala;
2. i processi di prototipazione possono essere notevolmente velocizzati grazie all'utilizzo di nuove tecnologie e tra queste, come detto prima, figura la RV;
3. il costante monitoraggio dei sistemi di produzione permette una riduzione degli errori e, quindi, meno casi in cui è necessario fermare le macchine, oltre al fatto che si riesce a far sì che dalla produzione risultino meno scarti;
4. l'utilizzo delle IoT permette di sfruttarne i vantaggi per ottenere un prodotto che sia più competitivo sul mercato.

Il cambiamento apportato da questo nuovo paradigma del mondo industriale porta con sé anche un cambiamento delle figure professionali che saranno richieste, principalmente inerenti al campo del management, dell'informatica e dell'ingegneria con competenze che vanno dal problem solving e quelle inerenti lo Smart Manufacturing che si occupa dell'innovazione digitale applicata a processi industriali.

1.6 Tecnologia 4.0 - Le Smart Technologies

Sono state individuate sei cosiddette Smart Technologies [6], ovvero computer o macchine che compiono azioni o prendono decisioni che di solito sono da riferirsi agli umani, per potersi inserire nel mondo dell'Industria 4.0. Sono state divise in due categorie:

1. la prima, più vicina al mondo dell'Information Technology (IT - riguarda archiviazione, elaborazione e trasmissione di dati) e include Industrial IoT, Industrial Analytics e Cloud Manufacturing;
2. la seconda, più vicina alle Operational Technologies (OT – fa riferimento ad hardware e software utilizzati per il controllo di dispositivi, processi fisici ed infrastrutture) e include Advanced Automation, Advanced Human Machine Interface e Additive Manufacturing.

Le Smart Technologies trovano applicazione in tutti i processi di un'azienda industriale e manifatturiera, dal processo di sviluppo di un nuovo prototipo con la relativa gestione del suo ciclo di vita e fornitori all'ambito della pianificazione di flussi fisici e finanziari fino ai veri e propri processi di produzione, di logistica sia interna che esterna, di manutenzione e sicurezza.

Di seguito sono riportati, più nel dettaglio, gli aspetti principali delle sei Smart Technologies elencate in precedenza:

1. Industrial IoT: attraverso la rete Internet ogni oggetto fisico può acquisire una sua controparte digitale; per realizzare questo passaggio vengono utilizzati oggetti intelligenti, con capacità di identificazione, localizzazione, diagnosi di stato, acquisizione di dati, elaborazione, attuazione e comunicazione, e reti intelligenti;
2. Industrial Analytics: nuove tecniche e strumenti per l'analisi dei dati, la loro simulazione e visualizzazione attraverso i quali ottenere informazioni da utilizzare per prendere decisioni in maniera rapida;
3. Cloud Manufacturing: attraverso Internet si ha un accesso diffuso, semplice e in tempo reale ad un insieme di informazioni e risorse utili sia nei processi di produzione sia nella catena di approvvigionamento;
4. Advanced Automation: fa riferimento ai recenti sistemi di produzione automatizzati capaci di adattarsi ed interagire con il contesto in cui si trovano ma anche di essere riconfigurabili e soggetti all'auto apprendimento. Un esempio di questi sistemi sono i robot collaborativi, sono progettati per lavorare in sintonia con gli operatori;
5. Advanced Human Machine Interface (Advanced HMI): con questo termine ci si riferisce al campo dei dispositivi wearable e delle nuove interfacce uomo - macchina tramite i quali è possibile sia acquisire che inviare informazioni sfruttando i canali vocale, visuale e tattile;
6. Additive Manufacturing: nota come Stampa 3D, modifica l'approccio classico dei processi produttivi basato sull'eliminazione o sulla deformazione del materiale poiché permette di creare un oggetto "stamandolo" uno strato dopo l'altro.

1.7 Competence Center

Parallelamente al piano Industria 4.0 è stata avviata anche la procedura per la creazione dei cosiddetti Competence Center, centri il cui compito è quello di svolgere attività di orientamento, formazione e supporto alle imprese circa la messa in atto di progetti di innovazione, ricerca e sviluppo con lo scopo di creare nuovi prodotti, di creare o migliorare processi o servizi sfruttando tecnologie avanzate con il fine ultimo di intraprendere il percorso di transizione verso l'Industria 4.0.

Il Competence Industry Manufacturing 4.0 (CIM 4.0) [7] è uno degli otto centri di competenza presenti sul territorio nazionale italiano. È un consorzio che vede tra i suoi membri il Politecnico di Torino [8] e l'Università di Torino [9], insieme a 23 importanti imprese private sia italiane che internazionali come General Motors [10], Avio [11], Thales Alenia [12] e Stellantis [13], multinazionale produttrice di autoveicoli nata dalla fusione tra Fiat Chrysler Automobiles e PSA; la società controlla quattordici marchi automobilistici tra cui Alfa Romeo, Chrysler, Citroën, FIAT, Jeep, Lancia, Maserati e Peugeot.

Per questo studio, il Politecnico di Torino ha messo a disposizione risorse umane, mentre CIM 4.0, oltre a risorse umane, ha accordato la possibilità di utilizzare i suoi spazi di ricerca e la strumentazione necessaria.

1.8 Obiettivo della tesi

L'obiettivo della tesi è quello di effettuare un confronto nell'utilizzo di due diversi sistemi di interazione all'interno di un'applicazione il cui scopo è quello di simulare il processo di prototipazione di un Low Cost Automation (LCA). In particolare, all'interno dell'applicativo, si è implementato il sistema di interazione basato sui guanti MANUS Prime II [14], che permette di interagire con oggetti del mondo virtuale riconoscendo le pose delle mani dell'utente, per valutarne eventuali vantaggi rispetto al sistema implementato precedentemente, quello degli HTC Vive Controller [15].

L'applicativo è stato sviluppato utilizzando il game engine open source Unity3D [16].

1.9 LCA

Come detto in precedenza, nel mondo dell'Industria 4.0, il concetto di automazione assume un ruolo centrale; tuttavia, i dispositivi a movimentazione autonoma comportano degli investimenti importanti che interessano il processo che va dalla loro progettazione fino alla realizzazione e manutenzione. Esiste una valida alternativa che richiede risorse decisamente più contenute, i Low Cost Automation (LCA, un esempio è mostrato nella figura 1.6).



Figura 1.6: esempio di un Low Cost Automation (LCA). Fonte: urly.it/3t4tg

L'idea su cui si basano gli LCA si rifà a quella del Karakuri Kaizen [17], teoria originaria del Giappone e utilizzata per la prima volta in Toyota, il cui scopo è quello di ottimizzare il flusso di materiali senza che ci siano sprechi di energia. I principi base sono i seguenti:

1. non spostare carichi in maniera manuale ma cercare di trovare soluzioni che lo facciano in automatico;
2. non investire grandi risorse (cosa che avverrebbe nel caso dei robot);
3. sfruttare la gravità, nonché l'esperienza e la creatività del personale dell'azienda;
4. sicurezza

Quando si vuole dare una definizione di automazione quello a cui si pensa subito è sostituire il lavoro dell'uomo con l'utilizzo di energia; tuttavia, questo può portare ad un suo spreco ed è qui che si inseriscono i dispositivi Karakuri Kaizen. Questi, infatti, sfruttano l'energia cinetica (dovuta, ad esempio, ad un contenitore industriale che scende lungo una corsia) rilasciata in modo che questa possa essere utilizzata per mettere in moto l'intero meccanismo. Si tratta di dispositivi che apportano diversi vantaggi in quanto hanno bisogno di meno tempo per essere assemblati (si parla di pochi giorni), hanno un numero di componenti minori, non consumano elettricità poiché non sono dotati di sensori e controller.

Essendo, inoltre, più semplici non hanno bisogno di una manutenzione particolare operata da specialisti ma allo stesso tempo possono essere adattati in base alle esigenze del momento.

1.10 Organizzazione dei capitoli

La tesi si compone della presente introduzione in cui sono stati inseriti i concetti generali di Realtà Virtuale, Industria 4.0 con un breve approfondimento delle tecnologie principali che la caratterizzano; sono stati, poi, introdotti brevemente gli LCA, oggetto dell'applicativo sviluppato.

Nel capitolo 2 si descrive più nel dettaglio la Realtà Virtuale con un piccolo accenno alla sua storia e con una descrizione dei sistemi di visione e di interazione esistenti mentre il capitolo 3 si focalizza sullo stato dell'arte e, quindi, sul legame già esistente e consolidato tra Realtà Virtuale e Industria 4.0.

Nel capitolo 4 viene fatta una panoramica sia della parte hardware che di quella software utilizzate per lo sviluppo del sistema in esame mentre nel capitolo 5 il sistema sviluppato viene spiegato più nel dettaglio, soffermandosi sui diversi elementi che lo compongono e come questi siano stati sviluppati affinché si integrassero al suo interno.

Il capitolo 6 presenta i test e i risultati che ne sono stati ricavati dopo aver fatto provare l'applicativo ad un campione di utenti che, al termine dell'esperienza, hanno risposto ad un questionario.

Infine, il capitolo 7 presenta le conclusioni che ne sono state tratte e i possibili sviluppi futuri del sistema.

Capitolo 2

Realtà Virtuale: concetti generali

2.1 Storia della Realtà Virtuale

In questo paragrafo viene riportata una breve panoramica sui dispositivi per la VR partendo dai primi prototipi risalenti agli anni '60 fino a dispositivi più avanzati utilizzati oggi. Viene, inoltre, presentato il Virtuality-Reality Continuum, il quale fornisce una tassonomia tra esperienze interamente reali e virtuali, classificando anche casi intermedi.

2.1.1 Panoramica dei dispositivi dagli anni '60 ad oggi

La VR affonda le sue radici già intorno agli anni '60 anche se si tratta di primi esperimenti che poco hanno a che fare con un'esperienza in Realtà Virtuale odierna; considerando, tuttavia, le disponibilità tecnologiche del periodo si tratta comunque di prodotti interessanti.

Il Sensorama (figura 2.1), un dispositivo del 1962 presentato alla fiera di New York, è considerato il primo apparecchio di RV e consiste in un simulatore di una passeggiata in motocicletta, aveva un sedile in movimento, dei phon che riproducevano l'aria e degli odori durante il percorso. Tuttavia, le immagini visualizzate non erano virtuali ma reali e mancava ancora l'interazione tra utente e ambiente virtuale.



Figura 2.1: Sensorama, 1992. Fonte: urly.it/3t4tk

Nel 1968 venne creato il primo caschetto immersivo stereoscopico in Realtà Virtuale. Le immagini erano proiettate su due schermi, uno per occhio; il dispositivo era fissato al soffitto e poi collocato sulla testa dell'utente. Nello stesso anno è nato BOOM: simile ad un binocolo, integrava due schermi ed era montato su un braccio meccanico che reggeva il binocolo e tramite il quale venivano inviate le informazioni ad un computer.

Nello stesso periodo fu presentato il dispositivo GROPE che prevedeva bracci meccanici che simulavano la resistenza al movimento dell'utente in modo da introdurre la componente tattile; la posizione di questi bracci era associata alla posizione di un oggetto nell'ambiente virtuale.

Nel 1977 fu inventato un guanto in grado di percepire e misurare la flessione delle dita, il Sayre Glove (figura 2.2). Si tratta di un prodotto realizzato da Daniel J. Sandin e Thomas DeFanti per il National Endowment for the Arts [18]. Si basano sull'utilizzo di sensori e tubi flessibili, ad un capo è presente una sorgente di luce mentre dall'altro è presente una fotocellula. Nel momento in cui le dita venivano piegate, la quantità di luce che colpiva la fotocellula cambiava e in base a questa variazione era possibile calcolare la flessione delle dita.



Figura 2.2: Sayre Glove, primo guanto capace di misurare la flessione delle dita. Fonte: urly.it/3t9d1

Nel 1978 il MIT (Massachusetts Institute of Technology) [19] realizzò l'Aspen Movie Map (due schermate di esempio sono presenti nelle figure 2.3 e 2.4) in grado di riprodurre una stazione sciistica da esplorare in tre modalità: inverno ed estate le quali si basavano su foto realizzate nelle due stagioni simile a StreetView, mentre la modalità poligoni era basata su un modello 3D prerenderizzando l'immagine da tutti i punti di vista non potendolo fare in tempo reale.



Figura 2.3: Aspen Movie Map. Visione basata su foto scattate. Fonte: urly.it/3t4jz

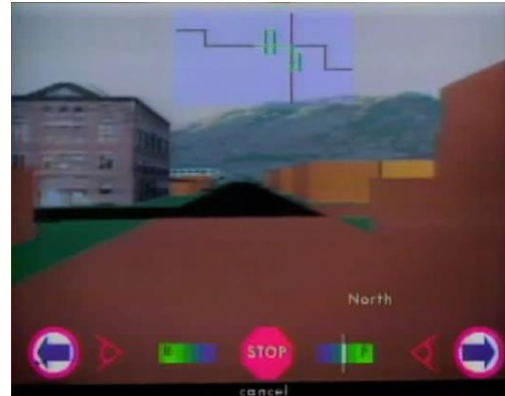


Figura 2.4: Aspen Movie Map. Visione basata sul modello 3D pre renderizzato. Fonte: urly.it/3t4jz

Nel 1979 viene presentato il 3SPACE, un sistema che permetteva di tracciare il movimento di un oggetto senza contatto con 6 gradi di libertà (posizione e orientamento), nonostante l'elevato rumore e la bassa precisione.

Il problema principale dei dispositivi fino a questo periodo consisteva nel fatto di essere stati progettati solo all'interno di laboratori di ricerca o in ambito militare. È nel 1987 che si ha la prima disponibilità in ambito commerciale grazie alla VPL con il suo DataGlove, un guanto sensibile che, tuttavia, aveva ancora problemi legati alla mancanza di feedback tattile e alla difficoltà di configurazione su diverse mani poiché era prodotto in una misura unica.

Successivamente, è stato il turno della Nintendo [20] con il Mattel PowerGlove (figura 2.5), un guanto che catturava info sul movimento della mano e delle dita e aveva un costo molto ridotto.



Figura 2.5: Mattel Power Glove. Fonte: urly.it/3t4tn

Il passo successivo è stato quello di integrare tutte le varie parti come nel caso del RB Model 2 che integrava un HMD con tracker magnetico, i dispositivi precedenti della

VPL ed era collegato ad un computer esterno che gestiva il rendering e ad un altro che gestiva l'audio 3D.

Negli anni '90, poi, c'è stato un disinteresse verso la VR dovuto in primo luogo alla difficoltà di rappresentare un ambiente complesso in realtime a causa della necessità di potenza di calcolo ma anche dovuto alla mancanza di interfacce ad elevata risoluzione e refresh rate e all'assenza di software in grado di gestire la progettazione di ambienti virtuali.

Un grosso impulso nell'interesse della VR si è avuto negli ultimi 15 anni dovuto fondamentalmente all'home computing, processo che consiste nello sviluppo di prodotti a basso costo in modo da essere accessibili a tutti ma in grado comunque di offrire una buona esperienza all'utente. Nonostante ciò, la RV non è ancora di massa per diversi motivi: innanzitutto è ancora molto difficile ingannare gli utenti e far sì che questi siano predisposti a credere di trovarsi in un mondo reale, ci sono problematiche legate al coinvolgimento sensoriale non ancora totale, spesso le esperienze sono monoutente e ci sono problemi legati all'invasività delle interfacce.

Il 2014 lo si può considerare come un anno di rinascita della Realtà Virtuale grazie anche all'interesse mostrato dalle grandi aziende, prima fra queste Facebook che acquista Oculus Rift [21], rilasciando il suo primo visore a marzo del 2016. Nello stesso periodo Google che annuncia il Google Cardboard [22], dispositivo ad un costo molto ridotto capace di trasformare uno smartphone in un visore di fascia bassa. Anche Sony [23], poi, annuncia di essere al lavoro su un visore per PS4 [24], il PlayStation VR [25], rilasciato anch'esso nel 2016.

Dall'Oculus Rift del 2016 si passa alla nuova famiglia di visori, i Quest che si presentano come più economici rispetto ai loro predecessori; nel 2019 viene messo in commercio l'Oculus Quest e un anno dopo l'Oculus Quest 2 (mostrato in figura 2.6). L'ultimo visore della casa Meta in commercio, il Meta Quest Pro [26] segna un altro punto di svolta in fatto di visori: si presenta, infatti, più orientato alla Realtà Mista e all'ambito lavorativo con un conseguente aumento di prezzo rispetto al Quest 2.



Figura 2.6: Oculus Quest 2. Fonte: urly.it/3t4tp

Per quanto riguarda casa HTC, invece, si sono susseguiti i visori della linea Pro con l'HTC Vive Pro (mostrato in figura 2.7) [27], il Pro 2 [28] e il Pro Eye [29], mentre nel febbraio del 2023 è stato rilasciato l'HTC Vive XR Elite (in figura 2.8) [30], un visore che mette insieme Realtà Virtuale e Realtà Aumentata.



Figura 2.7: HTC Vive Pro. Fonte: urly.it/3t4ts



Figura 2.8: HTC Vive XR Elite. Fonte: urly.it/3t4tv

Sempre a febbraio 2023 è uscito il PlayStation VR2 [31] (figura 2.9), il nuovo visore Sony [23] per PS5 [32] insieme al nuovo controller PSVR2 Sense in grado di inviare feedback aptici al giocatore.



Figura 2.9: PSVR2 con joystick PSVR2 Sense. Fonte: urly.it/3t4ty

2.1.2 Reality-Virtuality Continuum

Il Virtuality Continuum mostra lo spettro di soluzioni tecnologiche i cui estremi rappresentano da un lato il mondo fisico mentre dall'altro si ha un mondo totalmente digitale. Questo schema, messo a punto nel 1994 da Paul Milgram e Fumio Kishino, prende in considerazione solo aspetti che riguardano la rappresentazione visiva degli elementi.

La figura 2.10 mostra i quattro differenti stadi che compongono il Virtuality Continuum. Come detto prima, ai due estremi sono rappresentate la realtà fisica e la realtà virtuale. Nel mezzo si trovano situazioni che coinvolgono al loro interno sia il mondo fisico che quello digitale e, in base alla predominanza di uno o dell'altro, sono stati associati diversi termini. Si parla di Realtà Mista (Mixed Reality – MR) quando ci si vuole riferire all'insieme di Realtà Aumentata (Augmented Reality – AR) e Virtualità Aumentata (Augmented Virtuality – AV).

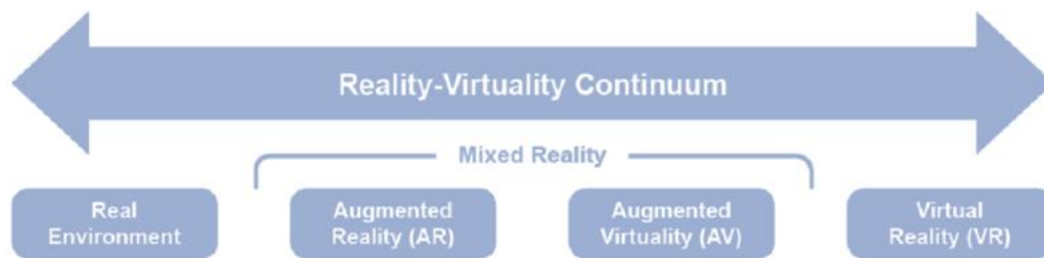


Figura 2.10: Reality-Virtuality Continuum di Milgram e Kishino, 1994. Fonte: urly.it/3t4vn

Con i termini Realtà Aumentata ci si riferisce ad una tecnologia che mette insieme elementi del mondo fisico con elementi in digitale dove, però, sono i primi a far da padroni; si ottiene, quindi, una rappresentazione della realtà con alcuni elementi virtuali. I dispositivi utilizzabili in questo caso possono essere smartphone, tablet e Google Glasses. Applicazioni in AR vengono molto spesso usate per inserire delle informazioni aggiuntive circa l'ambiente fisico all'interno del quale ci si trova come accade, ad esempio, nei musei.

Per Virtualità Aumentata (AV), invece, si intende un mondo costruito in digitale al cui interno figurano elementi appartenenti alla realtà; in questo caso, quindi, è la componente virtuale a prevalere su quella fisica.

Parlando di Realtà Mista (Mixed Reality – MR) il concetto cambia ulteriormente in quanto in questo parte reale e parte virtuale coesistono e, a differenza dei casi precedenti, interagiscono tra di loro. Dispositivi predisposti alla MR sono, ad esempio, i Microsoft HoloLens.

In un recente articolo [33], esponenti di università australiane e statunitensi hanno messo in risalto i limiti che questo modello ha al giorno d'oggi, limiti dovuti principalmente all'evoluzione che la tecnologia ha avuto dal 1994 ai giorni nostri. Il primo aspetto messo in risalto è relativo, come già accennato in precedenza, al fatto che il modello del Virtuality Continuum di Milgram e Kishino opera la classificazione solo in riferimento a ciò che viene mostrato, ignorando totalmente gli altri sensi che grazie alle tecnologie odierne possono essere, invece, stimolati. Un'altra criticità che viene messa in risalto da questa ricerca è relativa alla coerenza che ci dovrebbe essere tra gli elementi

reali e quelli virtuali, aspetto che non viene considerato all'interno del modello descritto in precedenza.

2.2 Sistemi di visione per la VR e loro funzionamento

Una simulazione immersiva deve cercare di coinvolgere il maggior numero di sensi, stimolati attraverso opportuni feedback. La componente principale è quella che fa riferimento alla vista.

Un display grafico è un'interfaccia che visualizza immagini di un mondo sintetico a uno o più utenti che interagiscono con il mondo virtuale. Si possono caratterizzare in base a diversi parametri quali tipo di visualizzazione (monoscopica o stereoscopica), risoluzione dell'immagine, FOV, tecnologia del display (LCD, LED, OLED), fattori ergonomici (peso...) e costi. Qualsiasi sistema ha sempre almeno un sistema di visualizzazione per definizione di realtà virtuale.

Oltre alle caratteristiche precedenti, un'altra categorizzazione è legata al numero di persone che possono usare contemporaneamente il display: esistono i Personal Graphic Display (monoutente come gli Head Mounted Display - HMD), i Desk Supported Display (multiutente), Large Volume Display (multiutente di grandi dimensioni).

Gli HMD sono caschetti con due display posizionati davanti agli occhi e molto vicini ad essi; per questo si usano delle tecniche che permettono di avere la sensazione che il monitor sia ad una distanza di almeno un metro.

Tra le caratteristiche da considerare rientrano il FOV che il caschetto è in grado di riprodurre, la risoluzione dei display, i pixel per grado da cui deriva la granularità dell'immagine, l'overlap binoculare e i gradi di libertà utilizzati per il loro tracciamento nello spazio.

Altro aspetto da considerare è il sistema di cablaggio in quanto ne esistono di due tipologie:

1. Tethered: dispositivi che necessitano di essere collegati al computer. Riescono ad elaborare un numero maggiore di dati poiché il carico computazionale viene lasciato al computer. Hanno, tuttavia, problemi legati al costo e alla presenza di cavi che limita la libertà dell'utente;
2. Untethered: non hanno bisogno di alcun collegamento ad un computer, integrano al loro interno tutti i componenti di cui necessitano. Conferiscono maggiore libertà di movimento ma hanno prestazioni minori in termini di elaborazione, di grafica, di accuratezza del tracking e di prestazione energetica.

I Large Volume Display (LVD), d'altra parte, permettono la visualizzazione multiutente e sono, generalmente, basati su proiettori dei quali bisogna considerare la Throw Distance, ovvero la distanza tra la lente e la superficie di proiezione.

Un esempio di queste strutture sono i CAVE (Cave Automatic Virtual Environment, figura 2.11), sistemi composti da un minimo di 2 ad un massimo di 6 pareti che fungono da superfici di proiezione e le cui immagini proiettate riferite all'ambiente della simulazione, in 2D, si adattano al punto di vista dell'utente che viene tracciato. Per ottenere la visione tridimensionale dell'ambiente e il tracciamento dell'utente si fa utilizzo di speciali occhiali 3D.

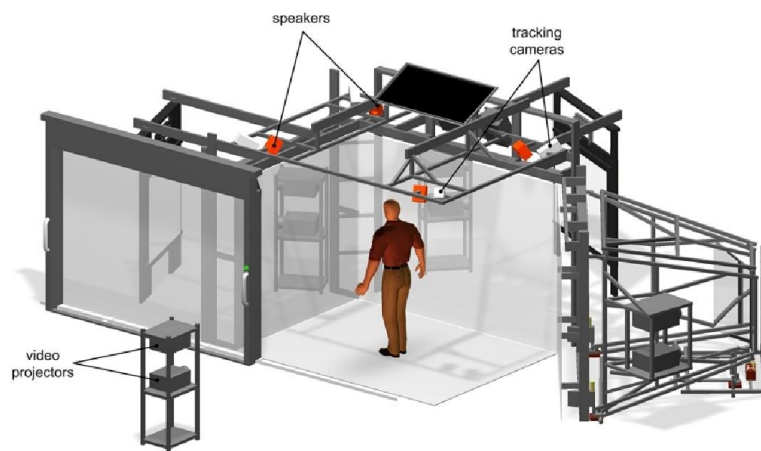


Figura 2.11: esempio della struttura di un CAVE. Fonte: urly.it/3t4v4

Il CAVE risale al 1992 [34], è frutto di un gruppo di ricercatori dell'Electronic Visualization Lab [35] dell'Università dell'Illinois [36] e venne mostrato alla conferenza sulla computer grafica in quello stesso anno. Si tratta di un sistema uno a molti, a differenza di quelli basati su HMD, e sfrutta schermi di proiezione di dimensioni importanti. Generalmente ha la forma di un cubo con display di dimensione di circa 3 metri per lato. Questi sono retroproiettati sfruttando, spesso, un sistema di specchi che permette di ridurre la distanza tra proiettore e schermo e, di conseguenza, anche lo spazio necessario all'implementazione del sistema. Talvolta, oltre ai display verticali e a quello orizzontale utilizzato come pavimento, se ne può avere anche un altro usato come soffitto, ottenendo una struttura a sei pareti.

Sugli schermi vengono proiettate due immagini 2D leggermente sfasate l'una dall'altra; il mondo virtuale tridimensionale viene reso indossando un paio di occhiali 3D che riescono ad introdurre la dimensione della profondità. Le lenti presentano un otturatore a cristalli liquidi che viene sincronizzato con la sequenza di proiezione. Tuttavia, gli occhiali hanno anche un'altra funzione, ovvero quella di permettere il

tracciamento della testa dell'utente che li indossa in modo tale che la visuale all'interno del mondo virtuale venga adattata di conseguenza.

Uno dei modi per interagire con il mondo mostrato all'interno del CAVE è quello di utilizzare una bacchetta (wand): si tratta di un controller che l'utente tiene tra le mani dal quale parte un raggio che permette di indicare degli elementi all'interno dell'ambiente digitale ed, eventualmente, selezionarli, modificarli, trascinarli in base allo scopo del sistema e a come questo viene programmato. Anche su questi controller, come sugli occhiali 3D, vengono posti dei marker in modo tale da poterne tracciare sia la posizione che la rotazione.

La componente visiva, tuttavia, non è detto che sia l'unica presente all'interno di un CAVE. Questo può presentare, infatti, anche una componente audio attraverso degli altoparlanti accuratamente posizionati all'interno della struttura e in posizioni strategiche in modo tale da fornire all'utente dei feedback il più vicini possibili alla realtà, aumentando il senso di immersione.

Un'altra componente eventualmente sfruttabile all'interno di un CAVE è quella tattile, sia in input che in output, tramite l'utilizzo di guanti da far indossare all'utente. Grazie ad essi è possibile interagire con oggetti virtuali in maniera più realistica di quanto non possa avvenire utilizzando, ad esempio, dei controller.

Alcuni guanti offrono anche la possibilità di fornire all'utente degli output come vibrazione o piccole scosse in seguito ad azioni compiute nell'ambiente. È possibile, inoltre, fornire dei feedback inerziali per simulare in maniera ancora più realistica l'azione di afferrare degli oggetti.

Tutti i dispositivi di input e output presentati vengono controllati da computer con prestazioni importanti che raccolgono i dati di input, li elaborano e producono gli output corrispondenti. Si tratta di computer, quindi, che devono garantire prestazioni avanzate ma soprattutto in realtime in modo da rendere l'esperienza dell'utente la più immersiva possibile.

La pipeline di rendering grafico consiste nel processo che permette la rappresentazione bidimensionale (quindi sul display) di elementi tridimensionali. Nella figura 2.12 ne è riportato un esempio.

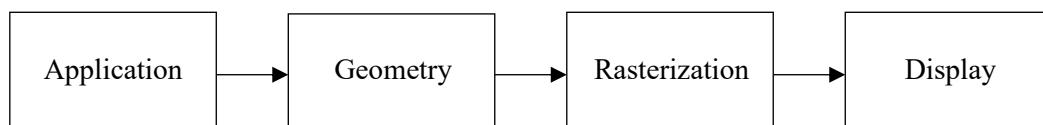


Figura 2.12: step pipeline rendering grafico

La pipeline di rendering grafico può essere vista come composta da tre macro blocchi:

1. Fase dell'Applicazione: gestita dalla CPU, si occupa di leggere i dati inerenti i modelli, quelli di input e in base alle informazioni ricavate aggiorna lo stato della simulazione;
2. Fase della Geometria: gestita dalla GPU, si occupa della geometria dei modelli, applica ad essi tutte le trasformazioni e calcola l'illuminazione della scena;
3. Fase di rasterizzazione: crea l'immagine finale 2D da mostrare sullo schermo.

La velocità finale con cui la pipeline di rendering viene portata avanti sarà definita da quella del blocco più lento, di conseguenza si possono avere diversi colli di bottiglia. Per evitare queste situazioni si possono operare delle ottimizzazioni quali la riduzione dei poligoni dei modelli, l'utilizzo del multitexturing, l'utilizzo di diversi livelli di dettaglio in base alla distanza a cui si trova l'oggetto rispetto al punto di vista dell'utente, in modo da ridurre il numero e la complessità delle operazioni per ottenere l'immagine finale.

2.3 Sistemi di interazione per la VR e loro funzionamento

Uno degli aspetti che contribuisce a rendere più realistiche le simulazioni è il poter interagire con gli elementi al loro interno e questo avviene attraverso diversi dispositivi di input. Innanzitutto, l'interazione può essere:

1. Diretta: l'utente interagisce in prima persona con l'ambiente virtuale;
2. Mediata: l'utente interagisce all'interno del mondo virtuale non più in prima persona ma in terza.

I dispositivi di input si possono dividere in 3 classi:

1. Dispositivi discreti: generano un evento singolarmente in base alle azioni dell'utente (mouse, tastiere, slider);
2. Dispositivi continui (o campionati): generano un flusso di dati legato o alle azioni dell'utente (attivi) o in maniera continua (passivi);
3. Ibridi: integrano entrambe le caratteristiche precedenti (il mouse reagisce al click ma genera anche un flusso di dati continuo legato alla posizione).

Un'altra classificazione dei dispositivi di input la si può avere in base ai gradi di libertà: si può andare da un grado di libertà come nel caso di pulsanti o tastiera a 6 (posizione e orientamento) fino a 6+N gradi di libertà.

Altra possibilità di classificazione è data dalla tecnologia usata per acquisire i dati. Quando si parla di applicazioni di tipo desktop i dispositivi di input principali sono mouse, tastiera e joystick; sono dispositivi che non sono stati progettati per la RV ma che sono stati adattati a quest'utilizzo. Ci sono altri dispositivi di tipo desktop, invece, che sono stati progettati per questo utilizzo come i mouse 3D. Dispositivi preferibili risultano essere quelli progettati appositamente per la RV come i tracker. Le informazioni che

servono per implementare queste interfacce sono le tre legate alla posizione e le tre legate all'orientamento (Yaw, Pitch e Roll), mentre la precisione dipende dalla tecnologia utilizzata.



Figura 2.13: informazioni di posizione e orientamento tracciate. Fonte: urly.it/3t4t-

I sistemi di tracciamento si dividono in 2 parti: un sensore che raccoglie informazioni sulla posizione e una componente che li elabora; questi dati vengono usati per ricavare la posizione dell'utente nel mondo virtuale e modificarne la vista di conseguenza.

Ci sono diversi aspetti che caratterizzano il funzionamento dei dispositivi di input:

1. Precisione: differenza tra posizione reale dell'oggetto tracciato e la posizione rilevata, minore è la differenza migliore è il funzionamento. Spesso i tracker sono più precisi quando si trovano al centro del sistema di riferimento;
2. Risoluzione: è il minimo cambiamento di posizione dell'oggetto che il tracker riesce a rilevare;
3. Rumore: variazione del valore misurato dal sensore quando l'oggetto non si muove nello spazio. Se il tracker fosse senza rumore il valore risultante dovrebbe essere costante; anche il rumore è legato alla distanza dal sistema di riferimento;
4. Deriva: indica l'incremento dell'errore fisso associato alla misurazione del sensore;
5. Latenza: ritardo che accumula il sensore per rispondere alla variazione di posizione dell'oggetto. La latenza non dipende solo dai sistemi di tracking ma anche nella pipeline di rendering determinata dal tempo che passa tra quando la pipeline riceve i nuovi dati e quando la nuova immagine è disponibile sul display;
6. Update rate: misurazioni al secondo che il dispositivo è in grado di fare, valore che dipende da fattori quali la tecnologia utilizzata.

2.3.1 Tracker meccanici

I tracker meccanici sono strutture cinematiche, in serie o in parallelo, i cui giunti contengono sensori per le misurazioni di angoli tridimensionali tra i segmenti. Nota la posizione del riferimento e le caratteristiche degli elementi della catena, è possibile calcolare la posizione e l'orientamento dell'oggetto. Tra i vantaggi figurano accuratezza costante sull'intera area di lavoro e latenza estremamente bassa, immunità alle interferenze dell'ambiente e allo shadowing. Gli svantaggi fanno riferimento al peso delle strutture e ai limiti della libertà di movimento dell'utente.

2.3.2 Tracker magnetici

Calcolano posizione e orientamento di un oggetto misurando le variazioni di campo magnetico originato da una posizione fissa. È necessario creare il campo magnetico attraverso l'utilizzo di 3 bobine ortogonali orientate lungo gli assi x, y e z; tali campi non vengono generati in contemporanea ma in sequenza altrimenti si influenzerebbero tra loro. È possibile utilizzare sia corrente continua che alternata.

Il principale problema legato a questa tipologia di dispositivi riguarda interferenze elettromagnetiche che ne influenzano il funzionamento, interferenze derivanti da eventuali oggetti metallici all'interno dei campi generati. Hanno il vantaggio, tuttavia, di essere meno ingombranti dei tracker meccanici e di funzionare anche nei casi in cui tra l'oggetto e il generatore del campo dovessero esserci degli ostacoli.

2.3.3 Tracker acustici

Il loro funzionamento si basa sulla generazione di ultrasuoni (non percepiti dall'uomo) nell'ambiente che vengono recepiti da un microfono e, in base al ritardo con cui questo li recepisce, si può calcolare la distanza tra la sorgente e l'oggetto. La sorgente è posta al centro del sistema di riferimento con altoparlanti posizionati ai vertici di un triangolo che emettono ultrasuoni in maniera sequenziale mentre il ricevitore ha tre microfoni; analizzando i dati raccolti è possibile calcolare posizione e orientamento del ricevitore nello spazio. Il vantaggio di questi tracker è il loro basso costo mentre gli svantaggi sono legati alla velocità del suono che varia con temperatura, pressione e umidità dell'aria; sono soggetti a interferenza con altre sorgenti di ultrasuoni e richiedono una visuale diretta tra trasmettitore e ricevitore. Sono, inoltre, più lenti di quelli magnetici per garantire la soppressione degli echi.

2.3.4 Tracker ottici

Attraverso le informazioni catturate da telecamere situate in diverse parti dell'area di tracciamento è possibile ricostruire la posizione degli oggetti. Si usano due tecnologie differenti (figura 2.14):

1. **Outside looking in:** le telecamere sono fisse nell'ambiente e riprendono gli oggetti nell'area interessata. Per identificarli si appongono su di essi dei marker che riflettono luce (passivi) o la emettono (attivi).
I marker passivi sono fatti di materiale che riflette i raggi infrarossi generati da degli illuminatori; questi raggi colpiscono il marker che li riflette creando degli spot, ripresi dalle camere, grazie ai quali è possibile calcolare la posizione degli oggetti.
I marker attivi, viceversa, funzionano essi stessi da illuminatori;
2. **Inside looking out:** le camere sono posizionate sull'oggetto in movimento mentre i marker vengono posti nell'ambiente di interesse. È possibile anche lavorare senza marker: in questo caso il funzionamento si basa sulla ricostruzione 3D dello spazio a partire dalle immagini catturate dalle camere.

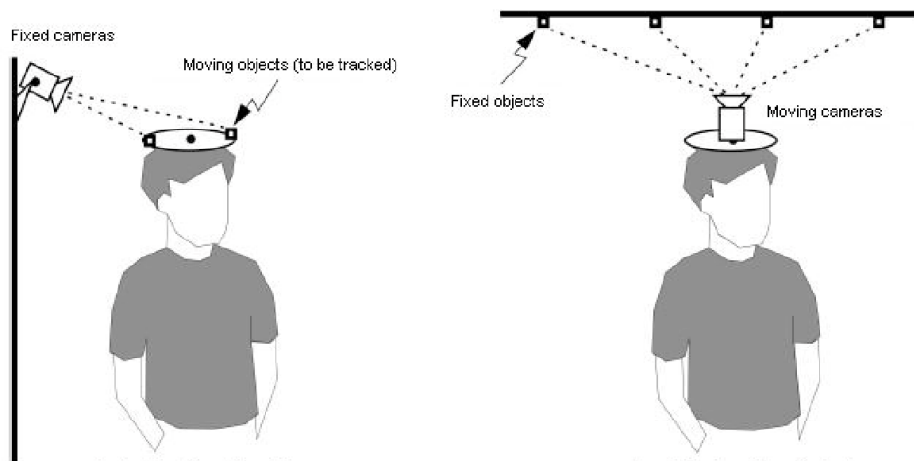


Figura 2.14: tecnologia Outside Looking In (sinistra) e Inside Looking Out (destra). Fonte: urly.it/3t4t_

I tracker ottici hanno lo svantaggio di soffrire di interferenze luminose ma lavorano con update rate molto alti e permettono di avere aree attive molto grandi.

2.3.5 Interfacce inerziali

Il loro obiettivo non è ricostruire perfettamente il movimento del corpo umano ma usare lo spostamento fisico dell'utente per navigare all'interno di un ambiente virtuale. Il problema è che non sempre si hanno spazi a sufficienza per muoversi. Per evitare questi problemi le interfacce odierne usano diverse tecniche come avere un tapis roulant

unidirezionale che faccia in modo che si possa camminare all'infinito senza che l'utente si sposti effettivamente oppure è possibile posizionare l'utente all'interno di una sfera cava supportata da altre sfere, senza alcuna limitazione al movimento ma con il vincolo di poter usare al suo interno solo tracker wireless.

Se, invece, non si vogliono utilizzare delle strutture fisiche per gestire il movimento esiste una tecnica molto utilizzata gestita all'interno delle simulazioni che è quella del teletrasporto, ovvero permette di spostare in maniera immediata l'utente da un punto di partenza ad una destinazione definita.

Esistono sistemi inerziali utilizzati per fornire feedback all'utente, lo scopo è quello di permettergli di percepire le forze di inerzia in accordo con quelle che sono le azioni e i movimenti che vengono effettuati all'interno dell'ambiente virtuale (simulatori di movimento).

Nella maggior parte dei casi si tratta di piattaforme mobili che hanno fino a 6 DoF con cui è possibile riprodurre spostamenti ma anche accelerazioni. In particolare, i movimenti che riproducono sono sincroni con quanto accade nella simulazione, lo scopo è quello di dare l'idea di trovarsi all'interno di un oggetto in movimento quale una macchina, una nave, un aereo e così via.

La tecnologia più usata è quella idraulica in cui ci sono piattaforme che si muovono o ruotano in accordo con quanto accade nel mondo virtuale. Nell'utilizzare questi dispositivi bisogna fare attenzione a non generare fenomeni di motion sickness causati da una non sincronizzazione di propriocettori (che reagiscono alle accelerazioni), senso dell'equilibrio e sistema visivo.

2.3.6 Tracker ibridi

Questi sistemi sfruttano i vantaggi di una tecnologia per ridurre gli svantaggi di un'altra. Un primo esempio sono i tracker che combinano dispositivi ultrasonici ed inerziali. In questo caso si vanno a compensare i problemi dei tracker inerziali dovuti alla scarsa accuratezza dei con quelli ultrasonici che sono molto più lenti ma accurati.

Un altro esempio è un sistema che integra tracker inerziali e a infrarossi: in questo caso i due sistemi si occupano di aspetti diversi poiché la parte a infrarossi viene utilizzata per calcolare la posizione del dispositivo mentre quella inerziale si occupa di calcolarne l'orientamento. È questo il meccanismo su cui si basa il WiiMote.

2.3.7 Motion capture

Questi sistemi catturano il movimento di una persona o di una sua parte del corpo. I dati catturati possono essere utilizzati o per controllare un avatar in tempo reale oppure, in un momento successivo, per l'animazione di personaggi sintetici per riprodurre fedelmente i movimenti del corpo.

Esistono sia sistemi ottici basati su marker, attivi o passivi, che sistemi markerless; hanno il problema, tuttavia, di non poter essere usati all'aperto per limitazioni dovute alle condizioni luminose. Un'alternativa sono gli esoscheletri, strutture utilizzabili all'aperto con svantaggi, però, legati alla loro intrusività e limitazione dei movimenti che ne deriva.

La motion capture può essere utilizzata anche per la cattura delle espressioni facciali. Si hanno sistemi, non utilizzabili in real-time, basati su marker attivi o passivi (da 10 a 100) posizionati sul volto del soggetto; in alternativa esistono caschi dotati di camere che funzionano con molti meno marker e danno un'idea del risultato in tempo reale.

2.3.8 Interfacce tattili

Si tratta di sistemi che permettono all'utente di interagire con elementi all'interno del mondo virtuale sfruttando sensazioni e movimenti del corpo. In questo paragrafo, in particolare, ci si sofferma su quelle che sfruttano i gesti e i movimenti delle mani come input, garantendo un'interazione il più naturale possibile. A tal proposito ne esistono di due tipi:

1. 3D probe: sono sonde meccaniche con 6 gradi di libertà, permettono di spostare uno stilo usato come sistema di riferimento che possiede un marker come contro parte 3D; sullo stilo sono presenti dei pulsanti tramite i quali poter inviare dei comandi al sistema. Forniscono feedback sia tattili che inerziali;
2. Guanti sensibili: dispositivi che misurano in tempo reale la posizione delle dita dell'utente per permettere un'interazione basata sui gesti. Uno dei problemi è legato al fatto che hanno un'unica misura e devono, quindi, essere adattati e ricalibrati in base dell'utente.

Un primo esempio di guanto sensibile è il Pinch Glove, non in grado di misurare la flessione delle dita ma solo di rilevare quali parti si toccano; queste info possono essere collegate a comandi ed eventi nell'ambiente virtuale.

Un sistema più avanzato è quello del 5DT Data Glove che si basa su fibre ottiche poste sulle dita: quando queste vengono piegate cambia la luce trasmessa da un capo all'altro della fibra ottica, rendendo possibile il calcolo della posizione delle falangi e dell'abduzione delle dita.

Il Cyber Glove (figura 2.15), infine, ha un numero elevato di sensori e permette di avere info su tutti i gradi di libertà della mano (ma non del polso); le info sulle posizioni angolari sono recuperate tramite sensori che misurano la variazione di resistenza in misuratori di tensione. Tuttavia, necessita di calibrazione per ogni utente oltre ad avere un alto costo.



Figura 2.15: Cyber Glove. Fonte: urly.it/3t4ta

Lo step successivo sarebbe quello di poter interagire con l'ambiente virtuale a mano nuda, senza indossare nessun tipo di sensore. Un primo dispositivo in questo senso è stato il LeapMotion [37] (figura 2.16), sviluppato nel 2013, e la sua successiva versione migliorata, il LeapMotion Orion.



Figura 2.16: Leap Motion. Fonte: urly.it/3t4v3

Attraverso dei dispositivi simili a quelli elencati in precedenza in questo paragrafo è possibile inviare dei feedback all'utente, i quali si dividono in due principali categorie:

1. Touch Feedback: forniscono informazioni circa temperatura, scivolosità, rugosità e geometria della superficie di contatto. Sono interfacce che usano sensori posti

in prossimità della pelle e non hanno nessun tipo di resistenza attiva al movimento dell'utente a seguito del contatto;

2. Force feedback: generano informazioni su resistenza, inerzia e peso della superficie di contatto, usando sensori su tendini e giunzioni per opporre resistenza attiva al movimento dell'utente a seguito del contatto. È importante tenere sotto controllo le forze che vengono generate in modo tale che non danneggino l'utente.

Le interfacce di tipo Touch Feedback possono essere di due tipi:

1. Desktop
2. Indossabili (guanti): oltre ad essere dispositivi di input possono anche inviare vibrazioni di diversa intensità al palmo della mano tramite vibratori. Con queste interfacce si possono associare agli elementi virtuali delle texture tattili, mappe che permettono di creare feedback tattili molto realistici. Per generare feedback più complessi si possono stimolare anche le dita come accade con il CyberTouch Glove.

Le interfacce Force Feedback, invece, possono classificarsi in base ai gradi di libertà che sono in grado di gestire, alle caratteristiche della struttura meccanica usata (se ne possono avere con struttura in serie o parallelo), in base ai punti di contatto, alla morfologia costruttiva (quelle antropomorfe riproducono fedelmente la struttura cinematica dell'arto), alla forza di picco e ai costi.

Non ci sono pipeline standard per il rendering di questi feedback, vengono implementate ad hoc e gestite via software con una mole di calcoli elevata dovuta alla necessità di gestire update rate alti. Un esempio è riportato nella figura 2.17.

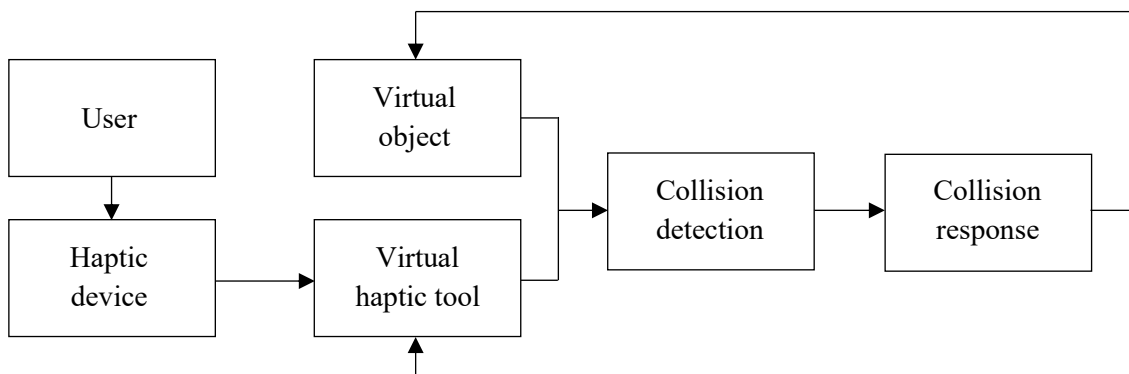


Figura 2.17: esempio pipeline di rendering tattile

Il primo step si occupa di verificare le collisioni tra i vari oggetti. L'identificazione delle collisioni è approssimata poiché implica calcolare le intersezioni tra oggetti che possono avere geometrie anche complesse. Per semplificare il processo si associa un

bounding box, fisso o variabile, all'oggetto e si verifica se i bounding box si intersecano; solo su questi vengono effettuati dei calcoli più precisi. La collision detection avviene in maniera discreta, bisogna quindi evitare di perdere delle collisioni: per far questo è importante che la distanza massima tra due bounding box sia minore del rapporto tra la velocità relativa degli oggetti e gli fps dell'applicazione.

Una volta identificata la collisione bisogna calcolarne gli effetti andando a deformare l'oggetto in base alle sue caratteristiche ma andando anche a produrre il feedback tattile corrispondente. Il calcolo delle forze in output si suddivide in tre fasi:

1. nella prima si calcolano le forze in base a diversi modelli di simulazione fisica come sistemi massa-molla e smorzamenti. Questo calcolo viene applicato una sola volta nel caso di oggetti con un singolo punto di contatto mentre nel caso di mesh più complesse queste vengono approssimate come un insieme di punti di contatto, per ognuno dei quali va effettuato il calcolo;
2. nella seconda fase ci si occupa di calcolare il corretto orientamento delle forze generate: la mesh è un'approssimazione della superficie dell'oggetto, di conseguenza la normale associata al punto di contatto non è del tutto corretta. Quello che si fa, quindi, è applicare lo smoothing delle forze, ovvero si calcolano le normali per i diversi punti di contatto e le si interpola;
3. lo step finale consiste nel mappare le forze calcolate sul dispositivo tattile che si sta utilizzando. Anche in questo caso non esiste un processo standard ma dipende dal dispositivo in questione.

Ultimo stadio della pipeline di rendering tattile è quello che calcola la componente touch feedback, ovvero temperatura e vibrazioni; per far questo molto spesso si usano delle texture tattili poiché ricreare l'andamento della superficie potrebbe essere molto dispendioso.

2.3.9 Interfacce non convenzionali

Per non convenzionali si intendono tutti quei dispositivi che si basano su tecniche non del tutto assestate. Un esempio di questo tipo sono le interfacce che riconoscono la voce, analizzando le onde sonore in ingresso e trasformandole in fonemi; funzionano bene in ambienti controllati e quando il set di comandi è limitato.

Un altro esempio sono i biosensori, dispositivi che rilevano parametri biometrici riferiti, ad esempio, all'attività muscolare, alla temperatura corporea, al battito cardiaco. Possono essere usati per interpretare le sensazioni di un utente in un mondo virtuale.

C'è un interesse nel catturare i segnali emessi dal cervello attraverso l'utilizzo di strumenti che monitorano l'attività cerebrale dell'utente. Possono essere utilizzati per compiere azioni in ambienti virtuali solo analizzando le onde cerebrali.

2.3.10 Interfacce tangibili

Per interfacce tangibili si intendono quelle interfacce attraverso cui l'utente manipola informazioni digitali tramite elementi presenti nell'ambiente reale (da non confondere con le interfacce grafiche che, invece, esistono solo nel mondo digitale).

Ci sono alcune caratteristiche chiave comuni a tutte le interfacce di questo tipo:

1. le rappresentazioni fisiche sono accoppiate ad informazioni digitali;
2. le rappresentazioni fisiche integrano meccanismi per interagire con esse. Alle interazioni con gli asset reali corrispondono dei cambiamenti all'interno del mondo digitale corrispondente;
3. le rappresentazioni fisiche integrano aspetti chiave dello stato del sistema digitale.

Un esempio è il SandScape (figura 2.18), sviluppato presso il MIT Media Lab [38]. Si tratta di un contenitore con all'interno della sabbia che può essere modellata dall'utente: tramite delle sorgenti ad infrarossi poste al fondo, una camera posta sul soffitto cattura le radiazioni che riescono a superare la sabbia, determinandone la geometria della superficie. Queste informazioni vengono poi elaborate in modo che attraverso un proiettore, posto nelle immediate vicinanze della camera, e in base alla modalità di visualizzazione scelta (Elevation, Slope, Shadow, Orientation e Waterflow) è possibile proiettare le relative informazioni sulla sabbia stessa. Ad esempio, se si è scelta la modalità Orientation, sulla sabbia viene proiettato del rosso nel caso di cumuli elevati, il giallo nel caso di cumuli di media altezza e il verde nel caso di cumuli bassi come se si trattasse di montagne, colline e pianure.



Figura 2.18: SandScape sviluppato presso il MIT. Fonte: urly.it/3t9m8

2.4 Interfacce utente per la VR

Come detto in precedenza, l'interazione all'interno del mondo virtuale è uno degli aspetti principali affinché la simulazione possa avvicinarsi il più possibile al mondo reale;

per questo motivo è fondamentale scegliere le modalità di interazione ottimali per il sistema che si sta progettando.

Nell'andare a definire che sistema di interazione inserire nel proprio applicativo si possono seguire tre diversi approcci:

1. si possono rimappare bottoni, mouse, tastiere su degli elementi del mondo tridimensionale. Ad esempio far sì che si possano utilizzare le frecce della tastiera per muoversi nello spazio e il mouse per cambiare la direzione di vista;
2. si possono sfruttare repliche di oggetti reali che garantiscano una corrispondenza immediata tra l'oggetto e la sua funzionalità, come l'utilizzo di volanti per controllare veicoli o ancora strumenti musicali e armi;
3. è possibile, infine, sviluppare interfacce basate sulla cattura di movimenti nello spazio come per il Kinect.

Per riuscire a sviluppare interfacce efficienti ed usabili si possono sfruttare le metafore dell'interazione, ovvero un insieme di elementi e procedure che sfruttano competenze che gli utenti hanno già in determinati contesti e che possono essere trasferiti in uno differente. Questo permette all'utente di capire in fretta come agire nell'ambiente in cui si trova, svolgendo operazioni non note sfruttando conoscenze che, invece, lo sono.

Esistono metafore in 2D o 3D che coinvolgono da 2/3 gradi di libertà (dispositivi desktop come mouse e joystick) fino a 6 gradi di libertà.

Le operazioni effettuabili all'interno di un mondo virtuale sono raggruppabili in tre categorie: navigazione, selezione e manipolazione, controllo del sistema.

2.4.1 Navigazione

Rappresenta l'operazione più comune effettuata all'interno dell'ambiente virtuale, scomponibile in due problemi: come effettuare lo spostamento da un punto ad un altro (come utenti che controllano avatar oppure che vengono spostati da un'auto) e in che direzione muoversi. Un altro aspetto da tenere in considerazione è relativo alla libertà di movimento concessa legata ai gradi di libertà del sistema che si è deciso di implementare: con un caschetto a 6 gradi di libertà è possibile esplorare l'ambiente in modo naturale a differenza di quanto accade con dispositivi con 3 gradi di libertà o desktop che limitano notevolmente il realismo degli spostamenti.

Una prima metafora di navigazione è il real walking, la più naturale tramite la quale si riescono a ridurre notevolmente i problemi di motion sickness. Il grande problema è legato al limitato spazio fisico in cui l'utente può muoversi; per ovviare a questo problema si applicano soluzioni quali l'aggiunta di muri virtuali. Un altro problema è legato alla sicurezza poiché l'utente potrebbe inciampare su elementi del

mondo reale; in questo caso si inseriscono dei guardian system (figura 2.19), barriere virtuali che limitano lo spazio in cui potersi muovere in sicurezza.

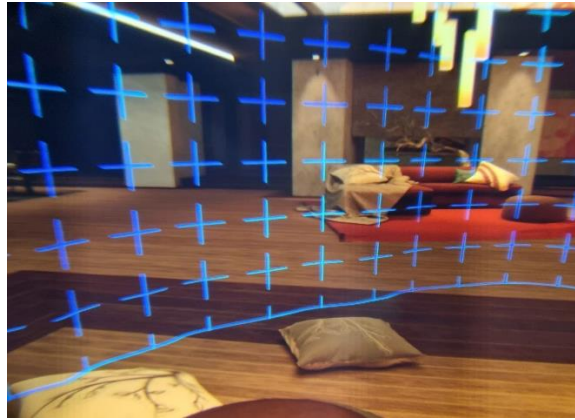


Figura 2.19: guardian system di Oculus Quest 2. Fonte: [urly.it/3t4v6](https://www.urly.it/3t4v6)

Un possibile modo per ridurre il problema dello spazio fisico è usare il walk in place, l'utente cammina sul posto, sfruttando l'orientamento della testa per capire in che direzione spostarsi. Ci sono, tuttavia, problemi legati al calcolo, ad esempio, della velocità con cui ci si muove che possono essere risolti attraverso l'utilizzo di accelerometri posti in dispositivi wearable.

Un'altra alternativa è quella del teletrasporto (figura 2.20) che consente nel puntare la posizione da raggiungere e spostarsi immediatamente nelle sue vicinanze, dopo aver premuto un pulsante sul controller. Per la selezione del punto finale si utilizza una linea o un arco che si propaga dal controller che dà un feedback sul punto in cui verrà teletrasportato l'utente. È una tecnica semplice, funziona bene in spazi piccoli ma può causare problemi di motion sickness in quanto l'utente si ritrova nella posizione di destinazione in maniera immediata.

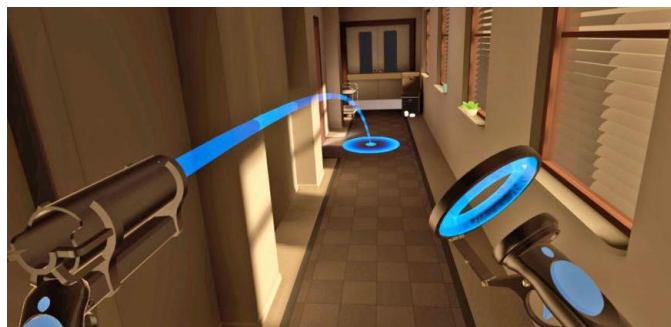


Figura 2.20: esempio di teletrasporto in VR. Fonte: [urly.it/3t4v8](https://www.urly.it/3t4v8)

2.4.2 Selezione e manipolazione

Per selezione si intende l'identificazione di un oggetto per un qualche scopo legato all'applicazione mentre manipolarlo vuol dire modificare una o più sue proprietà come posizione, orientamento, colore, scala e così via. Si tratta di operazioni fondamentali perché costituiscono il metodo principale per interagire con il mondo virtuale, la qualità con cui si riescono a manipolare è fondamentale per un buon risultato nell'interazione. Selezione e manipolazione spesso sono collegate.

La progettazione di metafore di selezione e manipolazione è complicata perché ci sono difficoltà legate allo spazio 3D in quanto gli oggetti si trovano a diverse distanze dall'utente e non sempre sono direttamente raggiungibili.

Una prima metafora è la Virtual Hand (figura 2.21) la quale prevede una selezione diretta: sfrutta una rappresentazione virtuale della mano dell'utente che controlla direttamente; la selezione di un oggetto avviene quando la mano virtuale lo interseca. Il vantaggio è che garantisce manipolazione e selezione naturale, il problema è legato proprio alla raggiungibilità dell'oggetto in quanto si ha il limite della lunghezza degli arti del corpo.

Una eventuale soluzione a questo problema è l'Arm Extension, ovvero un'estensione virtuale del braccio che semplifica il raggiungimento di alcuni oggetti anche se questo porta con sé problemi nella precisione dei dati di tracking.



Figura 2.21: Virtual Hand. Fonte: urly.it/3t4vb

Un'altra metafora è il Raycasting (detta anche Space Wand, figura 2.22): il dispositivo crea un raggio virtuale che si propaga nel mondo virtuale e di cui è possibile modificarne posizione e orientamento. Nel momento in cui il raggio interseca un oggetto, questo diventa l'oggetto della selezione. Si tratta di una tecnica immediata da implementare, intuitiva ed efficiente dal punto di vista dei calcoli. Ha lo svantaggio di non essere così efficiente nel momento in cui si vogliono selezionare oggetti totalmente o parzialmente occlusi.

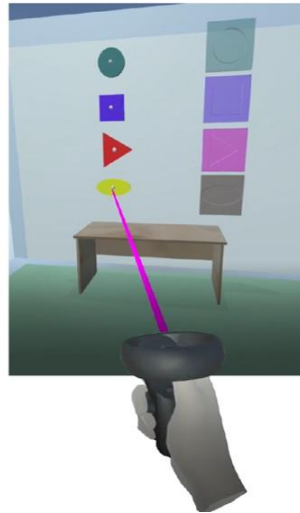


Figura 2.22: esempio di Raycasting. Fonte: urly.it/3t4vc

Direttamente legata al Raycasting c'è la metafora di manipolazione chiamata Homer: nel momento in cui l'utente seleziona un oggetto, ci si sposta nelle sue immediate vicinanze per averlo a portata di mano; terminata la manipolazione si torna nella posizione originale.

Un'altra metafora di manipolazione è lo Scale-world grab: una volta selezionato l'oggetto, il mondo viene ingrandito o rimpicciolito in modo che l'oggetto selezionato abbia le dimensioni della mano; al rilascio il mondo 3D riassume le dimensioni originali.

2.4.3 Controllo del sistema

Si tratta di operazioni che permettono di modificare lo stato attuale del sistema potendo modificare, ad esempio, il punto di vista, il caricamento di una scena o livello, la risoluzione delle immagini. Queste operazioni vengono gestite tramite dei comandi inviati al sistema tramite menù o pulsanti. Il controllo del sistema deve essere veloce e facile da imparare poiché comporta una distrazione dell'utente dai task che sta svolgendo, l'utente deve capire subito che il comando che ha inviato è stato ricevuto attraverso un'opportuna rappresentazione. Occorre distinguere due elementi diversi:

1. UI: con questo termine ci si riferisce ai metodi con cui l'utente interagisce con il controllo del sistema e alle interfacce usate;
2. HUD o barra di stato: rappresenta il metodo con cui le info di stato vengono trasmesse all'utente.

In base al fatto che la rappresentazione venga visualizzata all'interno del mondo virtuale o meno e che faccia effettivamente parte o meno del mondo, è possibile distinguere quattro tipi di interfacce (con alcuni esempi riportati nella figura 2.23):

1. Diegetiche: sono perfettamente integrate con l'ambiente creato, sono parte integrante della rappresentazione;
2. Non diegetiche: consistono in un overlay sulla schermata di gioco, generalmente rappresentate in 2D e slegate dalla narrazione;
3. Meta interfacce: sono simili a quelle non diegetiche, sono anch'esse in 2D ma si integrano con la narrazione;
4. Interfacce spaziali: sono presenti nel mondo 3D con lo scopo di fornire info aggiuntive ma mantenendo l'esperienza immersiva dell'utente; sono slegate dalla narrazione.

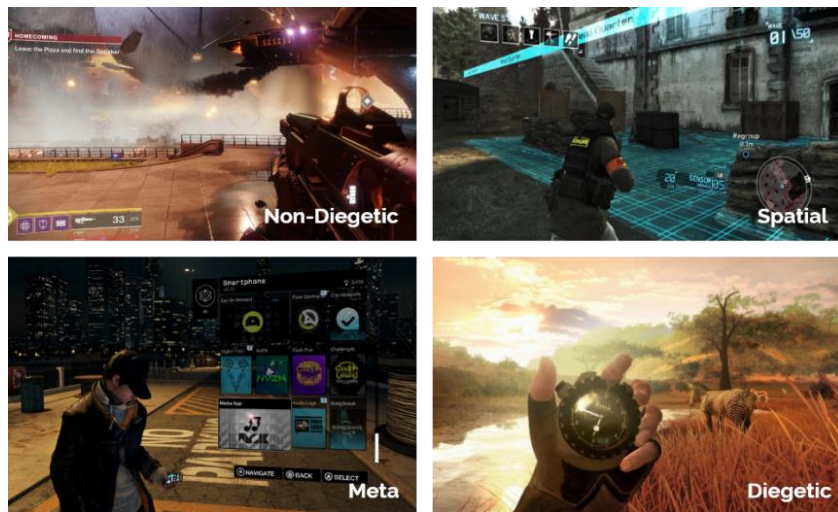


Figura 2.23: alcuni esempi di interfacce. Fonte: urly.it/3t4vj

Capitolo 3

Stato dell'arte

In questo capitolo viene presentato, da una parte, il rapporto tra la Realtà Virtuale e i processi di progettazione e prototipazione, con esempi provenienti sia dal mondo del design che da quello delle automobili. D'altra parte, vengono presentati degli studi condotti il cui oggetto è il confronto tra diversi sistemi di interazione per compiere azioni all'interno di mondi virtuali.

3.1 Realtà Virtuale e prototipazione

In un articolo pubblicato nel 2012 da Stefano Mottura e Marco Sacco [39] già si parlava di prototipazione attraverso l'utilizzo delle potenzialità di Realtà Virtuale e Aumentata. In particolare, viene portato come esempio del primo ambito il supporto che la VR può portare al processo di prototipazione di una moto. Viene spiegato, infatti, che questo processo prevede la realizzazione di un corrispondente fisico della moto che la rappresenta in ogni dettaglio per poterne valutare aspetto e proporzioni: nel momento in cui ci si dovesse accorgere di non essere soddisfatti del risultato o dovessero esserci ripensamenti si è costretti a produrre il componente corrispondente con le modifiche effettuate. Come si può immaginare, questo comporta una perdita di tempo da una parte con conseguente aumento delle tempistiche che vanno dall'ideazione della moto alla sua produzione e messa sul mercato, ad uno spreco di materiale.

Ecco, quindi, che emergono i due aspetti principali per cui una prototipazione all'interno di un ambiente virtuale risulta essere più flessibile e, in generale, più vantaggiosa rispetto al processo basato sulla prototipazione fisica.

3.1.1 IKEA VR Experience

Un primo esempio di applicazione di questo tipo è quella rilasciata nel 2016 da IKEA, chiamata IKEA VR Experience [40] (una schermata di esempio è mostrata nella figura 3.1) e disponibile gratuitamente sulla piattaforma Steam.

Si tratta di un'applicazione che propone un'esperienza in Realtà Virtuale Immersiva e che permette agli utenti di poter scegliere tra i prodotti disponibili dal

catalogo IKEA [41] e poter subito avere un riscontro di come questi si integrano tra di essi andando a definire l'arredamento finale della stanza in questione.

Indossando un visore per la VR come un HTC Vive [15], relativi controller e uno spazio sufficiente all'interno del quale potersi muovere, è possibile scegliere da un apposito menu diversi prodotti venduti dal brand, posizionarli all'interno dell'ambiente virtuale come meglio si crede, è possibile modificarne le caratteristiche quali dimensioni, colore, materiale e in alcuni casi tessuti. Una volta definiti gli elementi d'arredo è possibile anche modificare l'ora della giornata e settare le impostazioni delle luci in modo tale da avere un riscontro di come i materiali e i colori scelti rispondono a diverse condizioni di luminosità.

I complementi d'arredo inseriti all'interno della stanza in questione sono, inoltre, collegati direttamente all'e-commerce di IKEA [41] per un'esperienza d'acquisto diretta effettuabile da casa, senza la necessità di recarsi in negozio. Tuttavia, non è obbligatorio procedere all'eventuale acquisto non appena si è finito di arredare l'ambiente in quanto l'applicazione presenta una funzione per salvare il layout creato e, eventualmente, procedere all'acquisto in un secondo momento.



Figura 3.1: IKEA VR Experience. Fonte: urly.it/3t4vv

3.1.2 ShapesXR

Un'altra applicazione che permette di intraprendere percorsi di prototipazione è ShapesXR [42] (una schermata di esempio è mostrata nella figura 3.2), disponibile su MetaQuest. Si tratta di un'applicazione che permette, tramite l'utilizzo di un visore e relativi controller, di inserire qualunque elemento all'interno di un ambiente virtuale: sono disponibili, di default, forme geometriche primitive, elementi per progettare UI, modelli di avatar tramite i quali ottenere il prototipo per cui si sta lavorando che può riguardare gli ambiti più disparati e, quindi, anche quello industriale. Questo perché oltre a mettere a disposizione forme ed elementi di base, il sistema permette anche all'utente

di disegnare manualmente un oggetto tramite lo strumento pennello oppure di caricare modelli 3D di qualsiasi tipo all'interno di una libreria; una volta fatto l'upload del modello desiderato questo sarà subito disponibile nella sezione apposita all'interno del menu tramite il quale si istanziano i vari oggetti. Grazie a questa personalizzazione, quindi, l'applicazione può essere utilizzata per ottenere prototipi legati agli ambiti più disparati. Anche in questo caso, come nel precedente, è possibile andare a modificare posizione, rotazione, scala, colore e varie caratteristiche degli elementi per garantire il massimo livello di personalizzazione, potendo salvare il layout ottenuto e potendolo modificare in più sessioni. È, infine, possibile invitare altri utenti affinché collaborino all'interno dell'ambiente di lavoro.



Figura 3.2: Shapes XR. Fonte: urly.it/3t4vz

3.1.3 #NEXTGen (BMW Group Platform)

Restando nel campo di applicazioni simili a quelle descritte in precedenza ma spostandosi nel settore auto si trova il processo messo in atto da BMW [43] che ha deciso di iniziare a sfruttare le tecniche descritte negli esempi precedenti ed applicarle al processo di progettazione delle sue automobili. Sfruttando i vantaggi derivanti dal mondo della Realtà Virtuale, infatti, i processi di ideazione, prototipazione e sviluppo di nuovi prodotti e componenti si sono ridotti drasticamente, permettendo anche di risparmiare materiale durante il processo di prototipazione.

Quello che avviene ora, infatti, è che diverse figure possono lavorare su un determinato prodotto o su un suo aspetto in maniera anche collaborativa pur trovandosi in parti diverse del mondo, valutando vantaggi e svantaggi del percorso che è stato intrapreso ed eventualmente apportare in tempi brevissimi delle modifiche in modo da poter cambiare la strada che quel processo stava prendendo, il tutto senza aver utilizzato neanche un componente fisico.

Grazie alla VR, quindi, sia designer che progettisti possono già avere esperienza del prodotto finito in virtuale (esempio in figura 3.3), senza dover aspettare che questo passi effettivamente per tutto il processo di produzione, permettendo di individuare eventuali criticità con largo anticipo rispetto al passato.



Figura 3.3: #NEXTGen, BMW Group Platform. Fonte: urly.it/3t4w7

3.2 Hand Tracking in VR

Come detto nel capitolo precedente, nel momento in cui si ha un mondo virtuale è fondamentale la componente relativa all'interazione con oggetti che appartengono a questo mondo. In particolare, una delle possibilità esistenti prevede che si possa interagire con elementi virtuali sfruttando i gesti delle proprie mani. Di seguito sono riportati degli esempi di progetti che si basano sull'utilizzo delle pose delle mani per l'interazione con il mondo in VR.

3.2.1 Auto Hand – VR Physics Interaction

Auto Hand [44] (figura 3.4) è un asset a pagamento sviluppato per il game engine Unity3D [16] e disponibile sul corrispondente Asset Store. Supporta diversi sistemi per la realtà virtuale come OpenXR, SteamVR e Oculus. Il funzionamento si basa sull'associare pulsanti dei controller che si stanno utilizzando a pose delle mani per innescare determinati comportamenti. Ad esempio, se si decide di associare il pulsante Trigger alla posa di Grab, ogni volta che questo verrà premuto sarà possibile afferrare un oggetto. In questo caso, tuttavia, si tratta comunque di un'interazione basata sui gesti delle mani mediata, però, dai controller.

L'aspetto interessante di questo asset lo si ritrova nel momento in cui si utilizza un visore Oculus: in questo caso, infatti, oltre a poter mappare le pose delle mani sui

pulsanti del controller è anche possibile utilizzare il sistema di Hand Tracking messo a disposizione da Oculus. In questo caso, infatti, ci si può liberare del controller rendendo possibile il riconoscimento della posa delle mani da parte del sistema, andando a definire il valore di determinati parametri all'interno degli script preposti. L'interazione risulta, così, più immediata e realistica, aspetto favorito anche dal fatto che la posa che la mano assume risulta essere coerente con la forma dell'oggetto nel caso in cui questo dovesse essere afferrato.

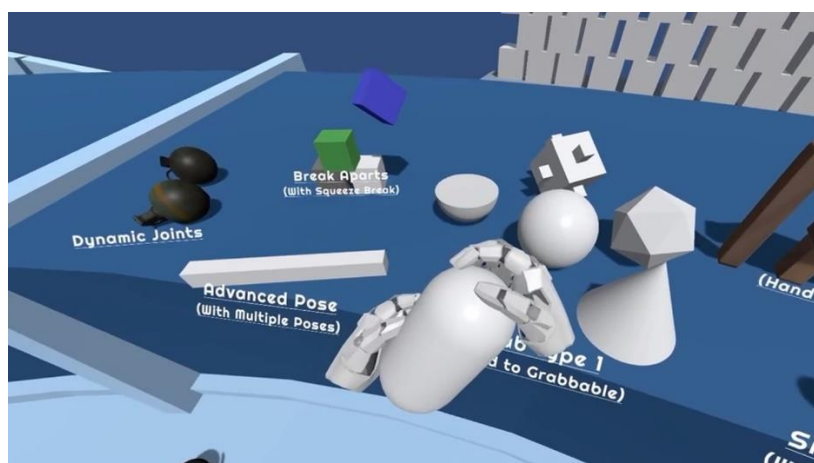


Figura 3.4: Auto Hand in Unity. Fonte: urly.it/3t4wr

3.2.2 Meta Presence Platform

Meta Presence Platform [45] (esempio in figura 3.5) è una suite rilasciata da Meta nel 2021 che permette di sviluppare applicazioni in Realtà Aumentata e Realtà Virtuale ancora più immersive. L'aspetto più interessante per questo studio riguarda l'interazione attraverso l'utilizzo delle mani, possibilità che viene offerta grazie all'Interaction SDK [46] presente al suo interno.

Si tratta di una libreria i cui componenti permettono di interagire con elementi del mondo virtuale o tramite controller o tramite le proprie mani. È supportata sia da visori appartenenti alla serie Meta Quest sia a visori non appartenenti ad essa che, però, richiedono delle configurazioni aggiuntive; per quanto riguarda Unity, invece, è supportata sulle versioni 2020 LTS (2020.3) e 2021 LTS (2021.3) che abbiano il relativo package per l'integrazione con Oculus installato.



Figura 3.5: Meta Presence Platform. Fonte: urly.it/3t4x5

Il suo funzionamento si basa su due concetti: Interactor e Interactable. Un Interactor è un componente che può agire sugli Interactable ad esso associati; questi ultimi, d'altra parte, sono componenti che reagiscono alle interazioni degli Interactor ad essi associati.

La libreria permette di definire le seguenti interazioni standard:

1. HandGrab [47]: basata su HandGrabInteractor e HandGrabInteractable, permette di afferrare gli oggetti predisposti a questo tipo di interazione utilizzando le informazioni provenienti dalle dita per definire quando l'azione inizia e quando finisce tramite la HandGrabAPI. Permette, inoltre, di far assumere alla mano delle pose definite precedentemente;
2. Poke [48]: basata su PokeInteractor e PokeInteractable, permette di interagire con superfici definite all'interno del mondo virtuale, superfici chiamate Pointable Surface per le quali viene abilitata la possibilità di riconoscere quando la mano si trova su di esse e quando le seleziona. È implementata anche la possibilità di interagire direttamente con Unity Canvas;
3. HandPose [49]: permette di riconoscere la posa assunta dalle mani sia attingendo da un set di pose predefinite sia sfruttando quelle custom definite dall'utente. Si basa sui componenti ShapeRecognition (permette di definire le dita che contribuiscono a determinare una posa), Transform Recognition (fa riferimento a posizione e orientamento) e Velocity Recognition;
4. Gesture [50]: si basa sul concetto di Sequences e permette di definire delle gestures complesse. Una sequenza contiene diversi Active State al suo interno, degli step intermedi che devono essere compiuti affinché la posa complessa venga riconosciuta; tutti gli Active State intermedi che compongono la sequenza devono essere verificati per un tempo minimo definito dall'utente;
5. Distance Grab [51]: permette all'utente di selezionare e spostare oggetti al di fuori della portata delle sue mani, ad esempio, attirandoli a sé (anche se possono essere definiti dei comportamenti personalizzati);

6. Touch Hand Grab [52]: permette di afferrare oggetti basandosi sulla configurazione dei loro collider, facendo in modo che le dita si adattino alla loro superficie in maniera dinamica;
7. Transformers: permette di definire i vari modi con cui un oggetto può essere manipolato (si può definire, ad esempio, come muoverlo, ruotarlo, scalarlo);
8. Ray Interactions [53]: permette di far partire un raggio dalla mano che, intersecando la superficie di un oggetto con cui poter interagire, permette di selezionarlo. Sfruttando una posa definita a tal proposito è possibile, ad esempio, scorrere un menu e, individuata la voce di interesse, selezionarla tramite la posa di Pinch.

3.2.3 Leap Motion

La Leap Motion Inc. era un'azienda americana nata nel 2010 (acquistata nel 2019 dalla società inglese Ultrahaptics che ne ha cambiato il nome in Ultraleap [54]), specializzata nella realizzazione e vendita di sensori tramite i quali è possibile tracciare il movimento di mani e dita; grazie al software rilasciato nel 2016 il tracciamento delle mani può essere utilizzato per applicazioni in Realtà Virtuale.

Il cuore di questo sistema di tracciamento si basa sull'utilizzo di due camere e LED ad infrarossi. Il prodotto consumer di Ultraleap è la Leap Motion Controller [37] (mostrato in figura 3.6).



Figura 3.6: Leap Motion Controller. Fonte: urly.it/3t4xq

Si tratta di una periferica USB da collegare al proprio pc oppure su un HMD. Attraverso tre LED ad infrarossi, viene generato un pattern di luce nel range degli infrarossi (IR) le cui riflessioni vengono catturate da due camere monocromatiche, anch'esse ad infrarossi. I dati catturati vengono inviati, tramite il cavo USB, al software sviluppato dalla company che, attraverso complessi calcoli matematici, riesce a risalire alla posizione della mano nello spazio. Il device è capace di rilevare la posizione della mano quando questa si trova entro un raggio di 1m. È questo l'aspetto che differenzia il Leap Motion Controller [37] dal Kinect [55], ad esempio: quest'ultimo, infatti, riesce a catturare movimenti entro un raggio più ampio rispetto al precedente ed è indicato nel

caso di movimenti all'interno di una stanza fornendo, però, risultati meno accurati di quanto non siano quelli del Leap Motion Controller [37]. Insieme alla componente hardware, al momento dell'acquisto, viene fornito anche un kit software basato su C# e due plugin i quali permettono di utilizzare i dati catturati all'interno di Unreal Engine [56] e Unity [16].

3.2.4 Confronto tra controller interaction e hand interaction

Di seguito sono riportati alcuni studi pubblicati il cui oggetto era quello di effettuare un confronto tra diverse interfacce di interazione all'interno di un mondo virtuale; i sistemi di interazione presi in considerazione sono i controller per la VR e l'interazione tramite le mani senza bisogno di altri dispositivi (tranne in un caso in cui si indossavano dei guanti).

Il primo è uno studio pubblicato a febbraio del 2023 ad opera di esponenti delle facoltà di Information Technology and Communication Sciences e del Dipartimento di radiologia del Tampere University Hospital di Tampere, in Finlandia [57]. Si tratta di un'esperienza in VR inerente il campo medico con lo scopo di mettere a confronto tre diversi sistemi di interazione: tramite mouse, tramite il tracking delle mani e tramite l'utilizzo in coppia di un controller e uno stilo per la VR; il mouse richiedeva l'utilizzo di una sola mano mentre gli altri due metodi hanno richiesto l'utilizzo di entrambe le mani. L'esperimento è stato sottoposto a 12 tester di cui dieci erano studenti universitari mentre gli altri due erano impiegati a tempo pieno, lontani dal mondo della medicina. Si trattava di sei uomini e sei donne con un'età media di 25 anni. Su 12 tester, due non avevano mai avuto un'esperienza in RV.

L'esperimento è stato sviluppato all'interno del software Unity [16] mentre, lato hardware, sono stati utilizzati i seguenti device (mostrati in figura 3.7): come HMD è stato scelto il Varjo VR2 Pro [58] il quale presenta un sistema integrato vision based per il tracciamento delle mani rilevate tramite il sensore [59] montato sul caschetto; i controller utilizzati sono stati i Valve Index Controller [60] mentre come stilo è stato usato il Logitech VR Ink [61]. Questi device venivano tracciati tramite le base stations SteamVR 2.0.



Figura 3.7: metodi di interazione utilizzati: mouse (a sinistra), hand tracking (al centro), controller e stylus (a destra). Fonte: urly.it/3t4xq

Il task sottoposto agli utenti si divideva in due parti: nella prima era richiesto di afferrare un oggetto posto nello spazio attraverso l'opportuna posa della mano oppure selezionarlo attraverso il mouse e, una volta afferrato, questo poteva essere spostato e/o ruotato per osservarlo da diversi punti di vista; la seconda parte consisteva nel segnare l'oggetto in un punto definito cliccando con il tasto sinistro del mouse (una volta selezionato l'oggetto), toccandolo e segnandolo con una penna virtuale nel caso si usassero solo le mani oppure toccandolo e segnandolo sfruttando lo stilo VR. Il task è stato ripetuto cinque volte per ogni modalità di interazione, ogni volta con un oggetto virtuale e in un ordine differenti.

Ai tester è stato sottoposto un questionario dopo ogni tentativo per ogni metodo di interazione e una volta effettuati tutti i tentativi con tutti i metodi di interazione. Ciò che è emerso è che la combinazione Valve Index controller – stilo Logitech VR Ink è risultata essere la migliore per il sistema proposto complice la sua alta accuratezza nel compiere le azioni che richiedono precisione; d'altra parte il sistema di interazione basato sulle mani è risultato essere il meno consigliato, risultato dovuto viceversa alla sua scarsa accuratezza. Questo è causato, principalmente, dal fatto che i sistemi di tracciamento vision based che vengono utilizzati per il tracking delle mani non sempre riconoscono le gestures con un livello di correttezza adeguato, il che porta l'utente a dover ripetere la posa diverse volte.

Il seguente studio è stato pubblicato nel dicembre 2022 ad opera di esponenti dell'Informatics Study Program e dell'Information Systems Study Program della Universitas Atma Jaya di Yogyakarta, in Indonesia [62]. In questo caso il confronto tra controller e interazione attraverso le pose delle mani riguarda l'ambito musicale; in particolare, lo scopo era quello di trasporre in virtuale un aspetto della cultura indonesiana, il Gamelan (ovvero un'orchestra di strumenti musicali che comprende

xilofoni, tamburi, metallofoni e gong) con il fine di attirare anche l'interesse delle nuove generazioni.

Gli utenti a cui è stato sottoposto il sistema erano studenti della Atma Jaya University di Yogyakarta con esperienze pregresse nell'utilizzo della RV e che conoscevano (direttamente o indirettamente) il mondo del Gamelan in modo da avere un metodo di confronto dell'esperienza virtuale con la realtà. Si tratta di 32 studenti che sono stati divisi in due gruppi, ad uno è stato fatto provare il sistema interagendo con i controller mentre all'altro è stata data la possibilità di interagire tramite l'Hand Tracking.

Le interazioni disponibili all'interno del sistema sono principalmente quattro: collision, pressing, grabbing e release. Collision viene usato nel caso di contatto tra l'oggetto e la mano virtuale; pressing viene usato per premere, colpire o toccare oggetti virtuali; grabbing è usato per afferrare elementi che poi si muovono in maniera solidale con la mano mentre release, viceversa, viene usato per rilasciarli.

A livello hardware, è stato usato un Oculus Quest poiché permette sia l'interazione attraverso i controller che quella attraverso l'utilizzo delle mani grazie all'Hand Tracking che implementa.

Al termine dell'esperienza, ai tester sono stati fatti compilare due questionari: il SUS (System Usability Scale) [63] per valutare l'usabilità del sistema e l'USEQ (USE Questionnaire) [64] per valutarne utilità, facilità d'uso, semplicità di apprendimento e soddisfazione. Infine, è stato preso un certo numero di rappresentanti di entrambi i gruppi di studenti per un'intervista di cinque minuti per valutare l'esperienza dal punto di vista emotivo e strumentale.

Dai risultati del questionario SUS è emerso che il sistema basato sull'interazione con i controller ha ottenuto un punteggio di 74.53/100 mentre quello basato sull'interazione delle mani ha ottenuto un punteggio di 64.38/100. Secondo il questionario SUS, un sistema può ritenersi soddisfacente nel momento in cui ottiene un punteggio superiore a 68/100 : da qui si deduce che l'utilizzo dei controller ha portato il sistema ad essere più usabile rispetto all'utilizzo delle gestures delle mani. Dalle interviste condotte con alcuni tester è anche emersa la causa principale che ha portato il sistema basato sull'Hand Tracking ad ottenere un punteggio al di sotto della sufficienza: la causa di ciò è da ricercarsi nel fatto che il tracciamento delle mani non è del tutto stabile, spesso gli utenti sono stati costretti a ripetere alcune azioni e hanno rilevato la presenza di lag.

Al contrario, invece, i risultati derivanti dal test USEQ hanno messo in risalto il fatto che tra i due sistemi di interazione, in termini di utilità, facilità d'uso e soddisfazione non è emersa una grande discrepanza, entrambe le varianti sono state considerate valide.

Il seguente studio è stato pubblicato nell'ottobre 2019 da parte di esponenti appartenenti a istituti portoghesi, in particolare al Polytechnic Institute of Tomar, alla

Universidade de Tras-os-Montes e Alto Douro e alla Univeristy of Coimbra [65]; l'occasione di pubblicazione è stata la IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2019) tenutasi a Bari.

Come nei casi precedenti, lo scopo è valutare se l'interazione sfruttando le gesture delle mani possa risultare più immersiva e usabile rispetto all'utilizzo dei classici controller; in questo caso, però, a differenza dei precedenti, non si usa un sistema di Hand Tracking ma l'utente indossa un guanto (in figura 3.8) capace di tracciare i movimenti del polso, l'orientamento della mano e i movimenti delle dita, liberandolo dall'obbligo di reggere un controller. L'ulteriore vantaggio dell'indossare un guanto rispetto all'interazione a mani "nude" sta nella possibilità di inviare dei feedback aptici all'utente, in aggiunta ai soli riscontri visivi. Il sistema è stato progettato utilizzando il game engine Unity [16], la posizione del polso viene derminata tramite HTC Vive Trackers (6 DoF) [66] mentre i movimenti di mani (3 DoF) e dita vengono rilevati dal prototipo di guanto in questione.

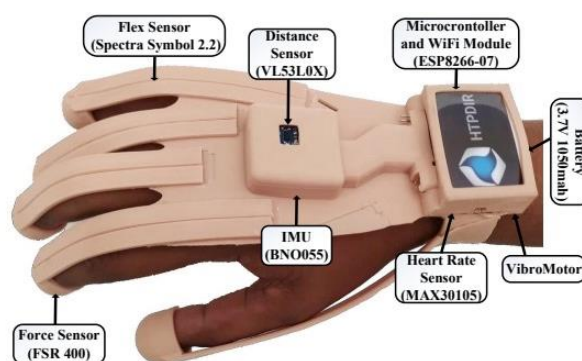


Figura 3.8: prototipo del guanto utilizzato nello studio presentato. Fonte: urly.it/3t4yw

Il task che gli utenti devono eseguire è molto semplice: si tratta di aprire una porta ruotandone il pomello con gli HTC Vive Controller [67] in un caso, premendo il pulsante corrispondente, e con il cyber glove nell'altro aprendo o chiudendo la mano.

I partecipanti ai test sono 22, sette uomini e quindici donne tra studenti e ricercatori appartenenti a corsi di ingegneria del Polytechnic Institute of Tomar. I partecipanti avevano un'età compresa tra i 20 e i 46 anni; tra tutti i tester, quattro non hanno mai avuto esperienze con tecnologie inerenti il mondo dei videogiochi mentre solo tre avevano già avuto esperienze con un dispositivo simile al cyber glove proposto.

Per poter visualizzare l'ambiente virtuale è stato usato un HMD, appartenente alla casa di produzione VIVE, dotato di due schermi AMOLED con una risoluzione ciascuno di 1080x1200 pixel (fondendoli insieme si ottiene una risoluzione di 2160x1200 pixel), un refresh rate di 90Hz e un FOV di 110°. All'interno è dotato di giroscopi e accelerometri

che, insieme ad un sistema di tracciamento laser outside-looking in, permette di ricavare la posizione della testa dell'utente mentre, come detto prima, la posizione della mano viene ricavata tramite gli HTC Vive Controller [67] in un sistema di interazione mentre in quello basato sulle gestures delle mani tramite gli HTC Vive Trackers [66].

Il task è stato fatto eseguire ai tester due volte, una per ogni sistema di interazione proposto e in un ordine random. Al termine dei task, agli utenti è stato somministrato un questionario per ottenere informazioni riguardo l'efficienza del sistema (misurando, ad esempio, il tempo impiegato dall'utente per aprire la porta) e il livello di immersione e presenza all'interno dell'ambiente.

Analizzando i risultati oggettivi per entrambi i sistemi di interazione non sono emerse particolari differenze che facciano intuire la preferenza dei controller sul cyber glove o viceversa. Guardando i risultati soggettivi inerenti il senso di presenza e immersione nell'ambiente, invece, a prevalere è stata l'interazione basata sul cyber glove: questa, infatti, è stata valutata come più naturale, migliorando il senso di presenza dell'utente all'interno del mondo virtuale. Un altro aspetto segnalato, inoltre, riguarda la libertà di movimento percepita dai tester: indossando il cyber glove si sono sentiti più liberi e senza la preoccupazione che avevano, nell'altro caso, di far cadere il controller.

Ci sono, invece, due aspetti limitanti segnalati riguardanti l'utilizzo del guanto: da una parte vi è la limitazione inerente la sua misura che è standard nonostante le diverse dimensioni della mano dei tester, mentre dall'altra è stata segnalata una certa latenza nell'adattamento del movimento delle dita alla reale posa della mano.

Un ultimo studio, infine, si differenzia dai precedenti in quanto non ha come oggetto il confronto tra due sistemi di interazione in quanto si parla sempre di interazione tramite pose delle mani ma sfruttando un HMD in un caso (quindi le mani reali non sono visibili) e implementando lo stesso sistema in un CAVE nell'altro caso (qui è possibile vedere le proprie mani). Il sistema di Hand Tracking utilizzato è basato sul Leap Motion Controller [37]. Lo studio è stato pubblicato nel marzo 2022 da un professore e tre dottorandi dell'Institute of Robotics di Valencia, in Spagna [68].

Nel primo caso è stato utilizzato come visore un HTC Vive (mostrato in figura 3.9) su cui è stato posto il Leap Motion Controller [37]. Insieme a questo HMD sono state utilizzate due base station che permettono di tracciare il caschetto nello spazio per ricavare la posizione dell'utente; l'area dello spazio virtuale è stata settata in modo da essere la stessa disponibile all'interno del CAVE (ovvero 6.25 m²).



Figura 3.9: HTC Vive utilizzato nella versione HMD dello studio descritto. Fonte: urly.it/3t4z3

Per quanto riguarda il CAVE, invece, si tratta di una struttura composta da quattro schermi (tre verticali e uno orizzontale come pavimento) di dimensione pari a 2.5x2.5m con una risoluzione di 1050x1050 pixel ciascuno. La visione stereoscopica di questi schermi viene resa possibile grazie all'utilizzo di occhiali 3D (in figura 3.10) su cui, anche in questo caso, è stato posto il Leap Motion Controller [37].



Figura 3.10: occhiali 3D utilizzati nella versione per il CAVE dello studio descritto. Fonte: urly.it/3t4z3

Entrambe le versioni del sistema sono state sviluppate su Unity [16].

L'esperimento è stato condotto su 17 adulti reclutati tramite social, di cui nove donne e otto uomini di età compresa tra i 19 e i 58 anni. La maggior parte di essi non aveva mai avuto esperienze né con un HMD né con un CAVE. Gli utenti sono stati divisi in due gruppi: ai membri del primo gruppo è stata fatta provare prima la versione con l'HMD mentre al secondo gruppo prima quella nel CAVE. Al termine di ogni prova è stato somministrato un questionario prima di provare l'altra versione del sistema a seguito

della quale è stato somministrato un altro questionario; infine, ne è stato somministrato uno comparativo dei due sistemi.

Per ognuno dei due sistemi implementati, ogni utente deve completare due task: il primo consiste nel prendere delle sfere colorate di diverse dimensioni da un tavolo e metterle all'interno di contenitori di dimensioni diverse, e del colore corrispondente della sfera, posti su un tavolo vicino al primo.

Il secondo task, invece, prevede che l'utente scriva una frase di 20 caratteri digitandoli su una tastiera presente nell'ambiente virtuale; nell'ambiente erano presenti in tutti tre tastiere con dimensioni dei tasti differenti.

Dai risultati provenienti dal test SUS è emerso che, considerando come 68/100 il punteggio minimo per considerare un sistema accettabile, la versione basata sull'HMD è risultata più usabile rispetto a quella basata sul CAVE con punteggi rispettivamente di 77.79/100 e 67.35/100; come si può vedere non solo il sistema basato sul CAVE ha ricevuto un punteggio più basso rispetto a quello basato sull'HMD, ma il punteggio risulta essere, anche se di poco, più basso anche della soglia minima definita dal SUS [63].

In generale, quindi, il sistema basato sull'HMD è risultato migliore in tutti gli aspetti che sono stati chiesti di valutare agli utenti tranne per uno su cui il campione di tester si è diviso a metà: ovvero sul fatto che vedere ed interagire con le proprie mani nel CAVE possa fornire un maggior senso di presenza all'interno dell'ambiente virtuale. Per quanto riguarda, invece, la prevalenza del sistema che fa uso dell'HMD questa si può rintracciare, anche leggendo le risposte aperte date dai tester nel questionario, nel fatto che nel CAVE è più facile notare degli errori nell'interazione, spesso non sono riusciti a capire quale fosse la posizione delle sfere rispetto alle proprie mani, nonostante il fatto che vederle fisicamente possa portare a pensare che sia un vantaggio.

Il report si chiude con il suggerimento da parte degli autori di uno sviluppo futuro dei sistemi qui riportati: viene proposta la possibilità di utilizzare diversi sistemi di Hand Tracking per valutarne le prestazioni e confrontarle; tra questi troviamo l'Hand Tracking integrato nei visori Oculus Quest ma, soprattutto, viene presentata la possibilità di utilizzare guanti per la RV come è stato fatto nel progetto descritto in questa tesi per il quale sono stati usati i MANUS Prime II [14].

Capitolo 4

Tecnologie utilizzate

4.1 Hype cycle di Gartner

Il modello dell'Hype Cycle (figura 4.1) definisce il ciclo di vita di utilizzo di una specifica tecnologia.

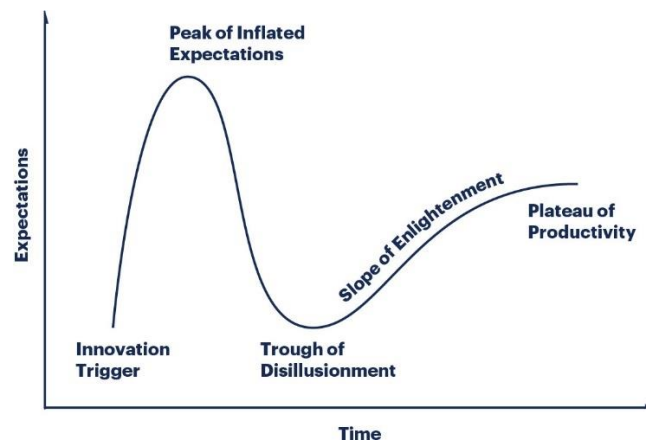


Figura 4.1: Hype Cycle di Gartner. Fonte: urly.it/3t4tz

Il modello suddivide il ciclo di vita in 5 fasi differenti:

1. **Technology Trigger:** è quella che definisce il lancio di un nuovo prodotto tecnologico, il quale si presenta come la soluzione a un problema. Dal lancio scaturisce pubblicità e interesse nel pubblico;
2. **Peak of Inflated Expectations:** la seconda fase mette in evidenza sia i successi derivanti dall'utilizzo della nuova tecnologia ma anche insuccessi e fallimenti;
3. **Trough of Disillusionment:** nel momento in cui le attese non vengono rispettate scatta la terza fase, quella della disillusione verso la nuova tecnologia che continua ad essere usata perlopiù dagli early adopter;
4. **Slope of Enlightenment:** la quarta fase, quella di risalita nell'utilizzo della tecnologia, viene messa in moto nel momento in cui la tecnologia stessa viene migliorata, producendo una seconda e terza generazione del prodotto;
5. **Plateau of Productivity:** la tecnologia in questione viene usata sempre di più grazie ai suoi miglioramenti e alla risoluzione di errori da cui era affetta.

4.2 Architetture hardware e software

Per quanto riguarda l'architettura hardware e software inerente il progetto sviluppato per HMD, questa si configura come mostrato nella tabella 4.1.


Hardware		Software	
Vive	MANUS	Vive	MANUS
 Workstation	 Workstation	 Unity	 Unity
 Base station	 Base station	 Visual Studio	 Visual Studio
 HTC Vive Pro	 HTC Vive Pro	 SteamVR	 SteamVR
 HTC Vive Controller	 HTC Vive Tracker		MANUS™ MANUS Dashboard + Manus Core
	 MANUS Prime II		




Tabella 4.1: Architettura del sistema sviluppato per HMD

In particolare, nel primo caso a livello di hardware è stata utilizzata una workstation con una CPU Intel Core i7, una scheda grafica NVIDIA GeForce Quadro 4000 e 128 GB di RAM e Windows 10 come sistema operativo. Per utilizzare l'applicazione, permettere l'invio di input al sistema e di output all'utente è stato

utilizzato un HTC Vive Pro [27] come HMD, i relativi Vive Controller [67] e quattro base station che ne permettono il tracciamento all'interno dell'area di gioco definita. Per quanto riguarda, invece, la versione che implementa i MANUS, è stata sviluppata sulla stessa workstation, utilizzando i guanti MANUS Prime II [14] per il tracciamento delle pose riprodotte dall'utente e sono state sfruttate le stesse base station per il tracciamento dei guanti nello spazio; in questo caso, i dispositivi che ne permettono di tracciare posizione e rotazione sono gli HTC Vive Tracker 2.0 [71] che vengono agganciati uno per ciascun polso e che sfruttano la stessa tecnica di tracciamento utilizzata per l'HMD e per i controller.

Lato software, invece, in entrambi i casi sono stati utilizzati SteamVR [69] per il tracciamento degli elementi nell'area di gioco, il game engine Unity [16] per lo sviluppo del sistema e lo strumento Visual Studio [87] per la scrittura del codice. Nel caso di utilizzo dei guanti sono presenti due ulteriori componenti: Manus Core [89] e la MANUS Dashboard che permettono di rilevare i dati provenienti dai sensori posti all'interno dei guanti e di inviarli a Unity [16] tramite opportuno plugin.

Per quanto riguarda la versione del sistema sviluppata per il CAVE, la struttura è mostrata nella tabella 4.2.

Hardware	Software
 Workstation	 Unity
 Barco UDM 4k15	 Visual Studio
 OptiTrack PrimeX 22	MANUS™ MANUS Dashboard + Manus Core
 Volfoni Edge VR	BARCO Software Barco

 <p>Marker</p>	 <p>Motive</p>
 <p>MANUS Prime II</p>	

Tabella 4.2: Architettura del sistema sviluppato per il CAVE

Il CAVE funziona utilizzando quattro proiettori Barco UDM 4k15 [79], otto camere OptiTrack PrimeX 22 [82] per il tracciamento degli oggetti all'interno dell'area di cattura e relativi marker posti su di essi ai fini del loro rilevamento da parte delle camere. La visione stereoscopica, invece, è resa possibile grazie all'utilizzo di appositi occhiali 3D, in particolare sono stati utilizzati dei Volfoni Edge VR [80]. Anche in questo caso sono stati utilizzati i guanti MANUS Prime II [14] per l'interazione con gli oggetti virtuali e la stessa workstation del caso precedente.

Lato software, invece, per lo sviluppo dell'applicativo in sé sono stati utilizzati gli stessi componenti del caso precedente, quindi Unity [16], Visual Studio [87], Manus Core [89] e la MANUS Dashboard. Sono stati utilizzati, in aggiunta, i software Barco [79] per la gestione dei proiettori e Motive [90] per il tracciamento dei marker nello spazio e l'invio dei relativi dati all'interno di Unity [16].

4.3 Hardware

Prima di analizzare nel dettaglio il progetto svolto, viene di seguito riportata una panoramica delle componenti hardware e software utilizzate per il suo sviluppo. La fase preliminare riguardante la loro scelta, infatti, assume una certa rilevanza per la progettazione di un'applicazione poiché, in base a quali componenti hardware e software si decide di utilizzare, le prestazioni finali possono essere anche molto differenti. È importante, quindi, innanzitutto individuare quali prestazioni si desidera avere per il proprio progetto per poi passare alla scelta delle relative componenti sia hardware sia software.

Per quanto riguarda l'HMD, e relativi controller, la scelta è ricaduta su un visore di casa HTC [15] poiché risulta avere delle prestazioni e una resa migliori rispetto, ad esempio, ai visori di casa Meta. I visori HTC, inoltre, possono essere facilmente integrati in un progetto sviluppato con Unity [16] grazie all'utilizzo del plugin SteamVR [69] che si occupa di tutta la gestione del rilevamento dei dati di tracciamento di posizione e rotazione del caschetto oltre che a permettere di mappare i pulsanti dei controller utilizzati (di cui viene tracciata posizione e rotazione) con funzioni definite all'interno del progetto che si intende sviluppare.

4.3.1 HTC Vive Pro

Tra le possibilità disponibili in casa HTC, la scelta è ricaduta sull'HTC Vive Pro [27] (figura 4.2).



Figura 4.2: HTC Vive Pro. Fonte: [urly.it/3t4ts](https://www.urly.it/3t4ts)

Si tratta di un visore messo in commercio nel 2018. Rispetto ai precedenti visori sviluppati da HTC, un notevole miglioramento lo si nota per quanto riguarda i due display utilizzati per la visualizzazione del mondo virtuale: si tratta di due schermi AMOLED da 3.5 pollici, aventi ciascuno una risoluzione di 1440x1600 pixel che, combinando i due occhi, permette di avere una risoluzione complessiva di 2880x1600 pixel. Si tratta di un aumento del 78% rispetto al modello precedente, garantendo una qualità delle immagini superiore di altri visori come l'Oculus Quest e il Windows Mixed Reality.

Il FOV garantito è di 110° mentre il refresh rate arriva ai 90Hz, con una latenza inferiore ai 20 millisecondi. È dotato, inoltre, di una fotocamera che, tramite il sistema Chaperone, avverte l'utente nel momento in cui dovesse avvicinarsi troppo ad ostacoli nel mondo reale. Questo, infatti, è sempre stato uno degli aspetti a cui prestare attenzione quando si parla di VR poiché l'utente, non avendo una visione diretta dell'ambiente reale in cui si trova ma potendosi muovere fisicamente al suo interno, rischia di urtare o

inciampare su oggetti in esso presenti. Tramite il sistema Chaperone, nel momento in cui l'utente si trova nelle vicinanze di un ostacolo, gli viene mostrata sui display del visore una griglia in modo tale da comunicare quale sia il limite oltre cui non andare per restare in un'area sicura.

A proposito dell'area di gioco disponibile per l'utente, la compatibilità con il sistema di tracking SteamVR [69], sia nella sua versione 1.0 che in quella 2.0, permette di realizzare delle esperienze VR room-scale andando a definire l'area di gioco che può arrivare ad avere un'ampiezza di 36m². In alternativa, si possono creare ambienti virtuali molto più ampi e sfruttare la tecnica del teletrasporto per muoversi al loro interno.

Anche dal punto di vista ergonomico presenta delle migliorie rispetto alla sua versione precedente. Il suo peso, infatti, risulta distribuito in maniera uniforme e l'aderenza alla testa è regolabile tramite apposita rotella posta sul retro del visore. È inoltre, presente, un sistema di cinghie regolabili che, essendo in tessuto, limitano il senso di pressione che il visore potrebbe causare sul cranio. È, poi, possibile regolare la distanza reciproca dei due display in base alla propria IPD (Inter Pupillar Distance).

Dal punto di vista della sensoristica di cui è provvisto è possibile ritrovare sensori per il tracciamento tramite SteamVR [69], sensori di prossimità, accelerometri e giroscopi.

Per il progetto sviluppato, infine, non è stato usato il visore collegato al pc ma è stata sfruttata la possibilità di utilizzarlo in modalità wireless tramite il Wireless Adapter [70]; questo sfrutta la tecnologia Intel WiGig con una banda di 60Hz.

Di seguito sono riportati i requisiti di sistema minimi per l'utilizzo del visore:

1. Processore: Intel Core i5-4590 o AMD FX 8350, equivalente o superiore
2. Grafica: NVIDIA GeForce GTX 970 o AMD Radeon R9 290, equivalente o superiore
3. Memoria: 4 GB di RAM o più
4. Uscita video: 1 porta HDMI 1.4 o DisplayPort 1.2 o superiore
5. USB: 1 porta USB 3.0 o superiore
6. Sistema operativo: Windows 7, Windows 8.1 o successivi, Windows 10

4.3.2 HTC Vive Controller

Uno dei due sistemi di interazione utilizzati all'interno del progetto oggetto di questa tesi è quello basato sugli HTC Vive Controller [67] (figura 4.3).



Figura 4.3: HTC Vive Controller. Fonte: urly.it/3t4z-

Si tratta di controller wireless che hanno una parte tramite cui possono essere impugnati e una parte circolare che presenta i 24 sensori utilizzati dalle base station per il tracciamento. Sono, inoltre, dotati di un laccetto all'interno del quale è possibile infilare il proprio polso per fissare meglio il controller ed evitare che cada durante una sessione di gioco. Ha una batteria di 960mAh ricaricabile tramite il cavo microUSB fornito; la loro autonomia va dalle 3 alle 5 ore. Offre, infine, la possibilità di fornire all'utente dei feedback aptici.

I pulsanti (mostrati in figura 4.4) presenti sono i seguenti:

1. Pulsante Menù tramite il quale è possibile interagire con i menù;
2. Trackpad che può essere sia premuto sia usato per rilevare le coordinate x e y del pollice nel momento in cui questo è a contatto con il pulsante, senza necessità di premerlo;
3. Pulsante System tramite cui è possibile accendere e spegnere il controller;
4. Status Light utilizzata per dare informazioni sullo stato del controller (indica è acceso, in carica, connesso o in modalità connessione);
7. Pulsante Trigger posizionato nella parte posteriore del controller;
8. Pulsante Grip posizionato lateralmente al controller.

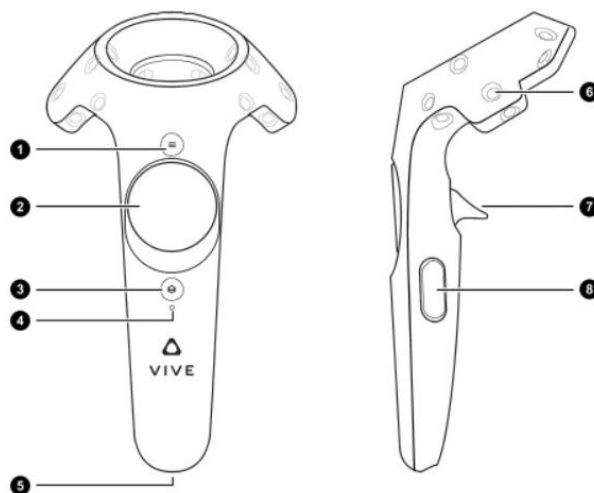


Figura 4.4: pulsanti presenti su un HTC Vive Controller. Fonte: urly.it/3t4-2

4.3.3 Vive Trackers 2.0

Nel caso in cui il sistema di interazione utilizzato fosse quello basato sui MANUS Gloves [14], c'è un altro prodotto di casa HTC necessario per poter tracciare la posizione delle mani all'interno dell'area di gioco, ovvero i Vive Trackers 2.0 [71] (figura 4.5), anch'essi compatibili con la versione 2.0 di SteamVR [69] e, quindi, facilmente utilizzabili in Unity [16].



Figura 4.5: HTC Vive Trackers 2.0. Fonte: urly.it/3t4-8

Si tratta di tracker che, tramite opportuno sistema di mounting, possono essere posti sul corpo dell'utente (in corrispondenza dei polsi, del bacino e dei piedi principalmente) o su un qualunque oggetto per poterlo tracciare nello spazio e spostare, di conseguenza, la sua controparte virtuale.

Possono essere utilizzati in modalità wireless connettendoli direttamente al visore (nel caso in cui non venissero usati i controller poiché l'HMD supporta al massimo due dispositivi wireless ad esso connessi) oppure inserendo il dongle all'interno del relativo adattatore e collegando questo, tramite cavo USB, al proprio PC. Tutti questi componenti vengono forniti insieme al tracker al momento dell'acquisto.

Il Vive Tracker [71] presenta i seguenti elementi (mostrati in figura 4.6):

1. Sensori in maniera analoga ai Vive Controller;
2. Porta USB per il caricamento della batteria;
3. Connettore Pogo Pin;
4. Superficie di attrito sottostante per evitare scivolamenti del dispositivo;
5. Sistema di mounting standard per posizionare il tracker su una camera;
6. Perno stabilizzatore;
7. Status Light che indica se il tracker è acceso, connesso o in modalità connessione;
8. Pulsante di accensione/spegnimento.

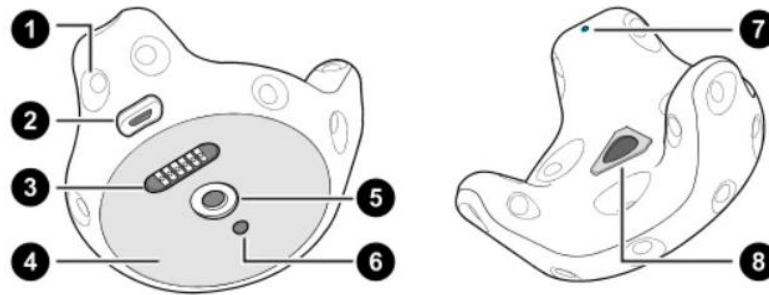


Figura 4.6: elementi presenti sugli HTC Vive Tracker 2.0. Fonte: urly.it/3t4-b

4.3.4 MANUS Prime II – Xsens Edition

MANUS [14] è un'azienda, con sede nei Paesi Bassi, leader nel settore dello sviluppo di tecnologie per la VR. In particolare, i suoi prodotti permettono il tracciamento di parti del corpo e delle dita per la Motion Capture in Realtà Virtuale. È stata fondata nel 2016 con il nome MANUS Machinae; in quell'anno è stato sviluppato il primo prototipo di guanto capace di ottenere informazioni sulla posizione di mani e dita.

Il guanto scelto per lo sviluppo di questo progetto appartiene alla serie Prime; in particolare si tratta di un prodotto frutto della collaborazione tra MANUS [14] e Xsens [72] da cui è nato il MANUS Prime II – Xsens Edition [73] (figura 4.7) che ha un costo di circa €3500.



Figura 4.7: MANUS Prime II - Xsens Edition. Fonte: urly.it/3t4-n

La scelta è ricaduta su questo modello in quanto, rispetto alla serie Prime I, i Prime II presentano dei sensori che non misurano più solo la posizione delle dita per definire se siano aperte o chiuse ma anche di quanto sono distanziate tra di loro, tracciandone i movimenti di avvicinamento e allontanamento. I sensori di flessione raccolgono dati in tre punti delle dita ricavandone i corrispondenti 11 DoF.

È presente, inoltre, uno slot per inserire la batteria che si può cambiare quando è scarica anche a guanto acceso; nel momento in cui viene inserita quella carica, il sistema si ricollega in automatico senza bisogno di dover ricalibrare il guanto; basterà, infatti, aspettare alcuni secondi e i dati di calibrazione verranno recuperati. La batteria, inoltre, ha una durata di circa 5 ore, superiore rispetto a quella dei modelli Prime I e può essere ricaricata in circa 1 ora e mezza.

Sul dorso del guanto, poi, c'è anche un nuovo sistema che permette di agganciarci diversi tipi di tracker per la posizione della mano così come viene data la possibilità di montare degli adattatori per i sensori Xsens.

La calibrazione è un processo molto rapido completabile in 45 secondi riproducendo le 3 pose indicate. Supporta, inoltre, la calibrazione multiutente.

Il tessuto, infine, è lavabile grazie alla semplice rimozione dei sensori dal suo interno. Al momento dell'acquisto, oltre al paio di guanti, vengono dati in dotazione due cavi USB per il loro caricamento e relativo adattatore, quattro batterie ricaricabili, due slot da montare sul dorso del guanto per inserirci i sensori Xsens, due per poterci agganciare tracker di posizione come i sopra citati HTC Vive Trackers [71] e, infine, un dongle che deve essere inserito nel PC in modo tale che vengano raccolti i dati di tracciamento.

Le specifiche del dispositivo sono le seguenti:

1. Latenza del segnale: <5ms;
2. Sample rate dei sensori: 90Hz;
3. Durata della batteria: 5h;
4. Caricamento: USB – C (5V);
5. Peso: 60g;
6. Comunicazione via cavo: USB – C;
7. Comunicazione wireless: protocollo proprietario HP 2.4GHz (certificato);
8. Tipologia sensori per le dita: 5x2 DoF Flexible Sensors e 6x9 DoF sensori IMU;
9. Accuratezza orientamento dei sensori: $\pm 2.5^\circ$.

I guanti Xsens di MANUS [14] sono incorporati in modo nativo nel software MVN Animate di Xsens per fornire una facile integrazione nella pipeline del software. I dati di rilevamento delle dita sono compatibili con tutti i principali software 3D come: Unreal Engine [56], Unity3D [16], Autodesk (3DS Max [74], Maya [75], MotionBuilder [76]), Houdini [77], Cinema 4D [78] e altri ancora.

4.3.5 CAVE (Cave Automatic Virtual Reality)

La seconda parte di questo lavoro di tesi consiste nell'implementare l'applicativo sviluppato con il sistema di interazione basato sui MANUS Gloves [14] all'interno di un CAVE. Si tratta di una struttura composta da quattro schermi retroproiettati, tre verticali (uno frontale e due laterali) e uno orizzontale (usato come pavimento). Di questi quattro schermi, quello centrale funziona come master mentre tutti gli altri sono slave, ricevono informazioni da quello centrale in modo che ci sia sempre coerenza di informazioni all'interno di tutto il sistema.

I proiettori utilizzati sono dei Barco UDM 4k15 [79] (figura 4.8), proiettori digitali DLP 3 in 4K adatti ad essere utilizzati in ambienti di grandi dimensioni.



Figura 4.8: proiettore Barco UDM 4k15 presente nel sistema CAVE utilizzato. Fonte: urly.it/3t4-x

Per poter visualizzare correttamente, in maniera stereoscopica, le immagini proiettate sugli schermi sono utilizzati gli occhiali 3D Volfoni Edge VR [80] (figura 4.9).



Figura 4.9: Volfoni Edge VR utilizzati per la visione 3D all'interno del CAVE. Fonte: urly.it/3t4-z

Si tratta di occhiali compatibili con la tecnologia DLP dei proiettori utilizzati, hanno un FOV orizzontale di 170° e di 115° in verticale. Al momento dell'acquisto, oltre agli occhiali, viene fornito un cavo microUSB per ricaricare la batteria (che ha

un'autonomia di 40h), aste ricambiabili di diverse dimensioni, una guida utente e dei panni per pulirne le lenti. Sulle aste laterali è presente un pulsante per accendere/spengere gli occhiali e un ingresso microUSB per poterli collegare ad un PC in caso di ricarica o di aggiornamento del software.

Per tracciare degli elementi all'interno del CAVE viene utilizzato il sistema OptiTrack [81]: in particolare sulla struttura sono montate otto camere (una per ogni angolo e una in una posizione intermedia per ogni lato della struttura) a infrarossi. I raggi emessi da queste camere vengono riflessi da marker passivi posti sugli elementi che si vogliono tracciare che diventano una sorta di spot luminosi; tramite delle tecniche matematiche e sfruttando le informazioni provenienti dalle diverse camere, si riesce a calcolare la collocazione e l'orientamento nello spazio dei vari oggetti tracciati.

In particolare, le camere utilizzate per questo sistema sono delle PrimeX 22 [82] (figura 4.10).



Figura 4.10: OptiTrack PrimeX 22 utilizzate per il tracking di oggetti nel CAVE. Fonte: urly.it/3t4_3

Si tratta di camere con una risoluzione di 2048x1088 pixel, con un'accuratezza pari a $\pm 0.15\text{mm}$ per quanto riguarda la posizione e pari a $\pm 0.5^\circ$ per quanto riguarda la rotazione (valori calcolati considerando un'area di tracking di 9x9m, marker di 1.4cm con un'esposizione delle camere pari a 800 e il minimo valore di f-stop possibile). Il framerate nativo è pari a 360fps ma è possibile arrivare a valori superiori ai 500fps in base alla risoluzione con cui le si vuole utilizzare. Le immagini che si ottengono hanno elevati valori di nitidezza grazie sia alla bassa distorsione che caratterizza gli obiettivi montati sulle camere, sia grazie alla scala di grigi a 10bit con cui vengono catturate le immagini. Per ottenere un ulteriore miglioramento delle prestazioni di tracciamento è opportuno eliminare tutte le interferenze che ci potrebbero essere e che potrebbero far sì che una camera consideri qualcosa che non è un marker come se lo fosse. Le due principali cause di interferenze sono la luce solare (per questo è consigliato bloccare tutte le finestre presenti nella zona della struttura) e altre eventuali sorgenti di luce infrarossa come luci a incandescenza, alogene e al sodio ad alta pressione).

Per quanto riguarda luci al di fuori del campo delle infrarosse non ci dovrebbero essere problemi poiché tutte le camere sono dotate di filtri appositi; eventualmente si può anche interagire con il software utilizzato per il tracciamento andando a mascherare gli elementi disturbanti. Mascherarli implica che le camere ignoreranno qualunque informazione in quelle zone e questo può influire sul tracciamento; pertanto, è sempre consigliato eliminare l'oggetto di disturbo, quando possibile, piuttosto che mascherarlo via software.

Per quanto riguarda il posizionamento delle camere, inoltre, è importante che queste vengano disposte in modo da catturare una porzione unica dello spazio e che i loro FOV si intersechino nelle zone di cattura principali. Per queste ragioni è consigliabile posizionare le camere ad altezze diverse.

4.4 Software e Plugin

Il progetto all'interno del quale sono stati integrati i due sistemi di interazione citati, ovvero quello basato sugli HTC Vive Controller [67] e quello sui MANUS Gloves [14], è stato sviluppato nel game engine Unity3D (versione 2020.3.44f1) [16] integrando il plugin SteamVR (versione 2.0) [69] per utilizzare l'HTC Vive Pro [27] come sistema di visualizzazione, e il plugin di MANUS (versione 1.6) [83] per poter integrare i guanti e poterli utilizzare per interagire con gli elementi all'interno dell'ambiente virtuale.

Il progetto per il CAVE, invece, è stato sviluppato in Unity3D (versione 2019.4.28f1) [16] utilizzando anche in questo caso il plugin MANUS [14] per l'integrazione dei guanti, e un template sviluppato da Reply [84] per permettere la corretta visualizzazione del progetto all'interno del CAVE.

4.4.1 Unity

Lanciato nel 2005 come un game engine esclusivo per Mac OS X, Unity [16] è diventato uno dei software più utilizzati per lo sviluppo di diverse tipologie di applicazioni: è possibile, infatti, sviluppare progetti in 2D e 3D, mobile e desktop, in realtà virtuale e aumentata non necessariamente appartenenti al campo dell'entertainment. Ha iniziato ad essere utilizzato anche in ambiti quali la cinematografia, l'industria, l'architettura e molti altri grazie alla sua versatilità. Permette, infatti, di rilasciare progetti per più di 20 piattaforme differenti tra mobile e desktop, realtà virtuale e console.

A partire dalla sua versione 5, rilasciata nel 2015, il linguaggio di programmazione principale utilizzato per lo sviluppo di applicazioni su Unity [16] è diventato C# (C Sharp), un linguaggio orientato agli oggetti.

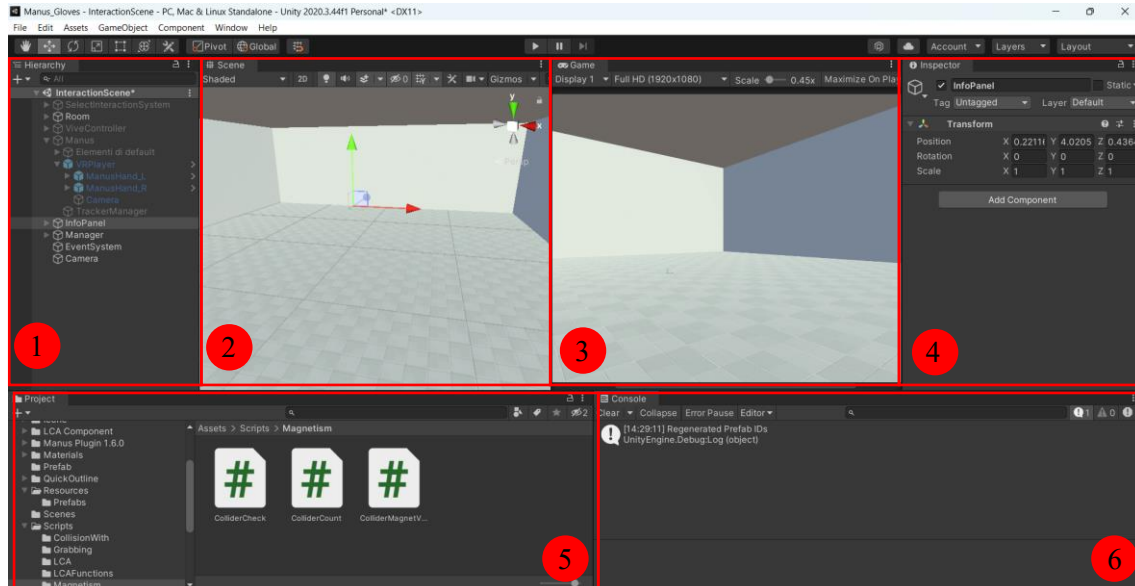


Figura 4.11: interfaccia di Unity versione 2020.3.44f1

La figura 4.11 mostra l'interfaccia di Unity la quale è composta dai seguenti elementi:

1. Hierarchy: è la finestra nella quale è presente l'elenco di tutti gli oggetti, chiamati GameObject, presenti all'interno della scena, che siano essi visibili o non visibili. In questo pannello è possibile creare un nuovo GameObject all'interno della scena, andando a cliccare sul pulsante "+" presente nella parte alta sinistra della finestra, ma anche andare a definire rapporti di parentela tra oggetti. È possibile, infatti, semplicemente trascinando un elemento sotto un altro far sì che il primo diventi figlio del secondo ereditandone le caratteristiche; nel momento in cui cambia la posizione, ad esempio, del padre viene modificata di conseguenza anche quella dei suoi figli;
2. Scene View: questa finestra mostra il mondo che si sta sviluppando. Contiene tutti i GameObject inseriti nella scena e che sono resi visibili. È possibile navigare al suo interno in modo da esplorare l'ambiente, permette di selezionare gli oggetti presenti e modificarne proprietà come posizione, rotazione e scala; consente, inoltre, di eliminarli;
3. Game View: in questa finestra viene mostrato il mondo creato, renderizzato dal punto di vista della camera (o delle camere) inserita. Qui è possibile vedere come apparirà il tutto all'interno della build finale;
4. Inspector: all'interno di questa sezione sono presenti tutte le proprietà del GameObject selezionato e da qui è possibile modificarle. Fornisce informazioni base sull'oggetto come nome, tag, layer, materiale dell'oggetto e sua transform

(contenente i valori di posizione, rotazione e scala), ma anche tutti i componenti che gli sono stati assegnati come Collider, Rigidbody, MeshRenderer e Script;

5. **Project:** in questo pannello vengono mostrati tutti i file presenti all'interno del progetto che si sta sviluppando. È possibile trovare i modelli degli elementi inseriti, la cartella con i relativi materiali, gli script creati per definire dei comportamenti all'interno dell'ambiente, eventuali altre risorse esterne importate (come immagini, video) ed eventuali package e/o asset importati tramite l'Asset Store di Unity [85].

È possibile cercare gli elementi manualmente oppure filtrarli scrivendone il nome all'interno della barra di ricerca;

6. **Console:** è la finestra all'interno della quale vengono mostrati errori, warning e messaggi di vario tipo generati dall'Editor, come errori di compilazione. Si tratta di uno strumento molto utile allo sviluppatore poiché permette di capire quale aspetto del sistema non funziona o non si comporta come è stato programmato per poter intervenire di conseguenza.

È possibile, inoltre, definire dei messaggi personalizzati da stampare all'interno della Console sfruttando la classe Debug.

Per poter definire i comportamenti desiderati all'interno del proprio progetto sono fondamentali gli script, associabili ad uno qualunque dei GameObject presenti all'interno dell'ambiente; ogni oggetto può avere nessuno, uno o più script. Come detto prima, il linguaggio di programmazione utilizzato è C# e ogni script deve derivare in maniera esplicita dalla classe MonoBehaviour.

Ci sono due principali metodi che caratterizzano script di questo tipo e sono:

1. **Start():** viene chiamato all'avvio dell'applicazione, al primo frame dal momento in cui lo script è abilitato. Viene usato generalmente per inizializzare le variabili che verranno utilizzate nel codice contenuto in quello specifico script;
2. **Update():** non è detto che uno script necessiti di questo metodo. Viene utilizzato per definire comportamenti, azioni, definire il contenuto di variabili e così via per aggiornare costantemente lo stato dell'applicazione. A differenza del metodo Start(), infatti, questo viene chiamato ad ogni frame.

Presenta due varianti: FixedUpdate(), chiamato ad ogni frame impostato, però, all'interno del sistema fisico presente in Unity, e LateUpdate(), chiamato ad ogni frame ma dopo che è stato completato il metodo Update.

Esistono, tuttavia, diverse funzioni di libreria messe a disposizione da Unity; in particolare per questo progetto ne sono state usate due per rilevare le collisioni tra GameObject:

1. **OnTriggerEnter(Collider col):** viene chiamato nel momento in cui un oggetto collide con un altro. Per far sì che questa collisione venga rilevata e che venga aggiornato lo stato del sistema di conseguenza, gli oggetti che collidono devono

- avere un Collider (almeno uno deve avere la spunta in corrispondenza della proprietà `IsTrigger`) e un `RigidBody`;
2. `OnTriggerExit(Collider col)`: viene chiamato, a differenza del precedente, quando termina la collisione tra i due oggetti.

4.4.2 Visual Studio

Si tratta di un ambiente di sviluppo integrato di Microsoft [86] che permette la creazione di progetti utilizzabili su diverse piattaforme tra cui desktop, mobile e console. Si tratta di uno strumento versatile grazie al supporto di un certo numero di linguaggi di programmazione quali Visual Basic .Net, Java, Java Script e C# che è quello utilizzato in questo progetto. Il supporto per questa serie di linguaggi è stato reso possibile dal momento in cui è nata la piattaforma .NET, anch'essa sviluppata da Microsoft [86].

Visual Studio [87] presenta al suo interno la tecnologia IntelliSense grazie alla quale è possibile intercettare errori nel codice, sia di tipo sintattico che logico; possiede, inoltre, un debugger grazie al quale, una volta individuato l'errore, vengono proposte al programmatore anche possibili soluzioni.

Il biennio 2015 – 2017 è stato un periodo di espansione per Visual Studio [87] durante il quale è stata sviluppata la possibilità di integrarlo con Unity [16]. La versione utilizzata per questo progetto è quella distribuita nell'aprile 2019 disponibile in tre versioni: quella Community, gratuita, mentre quelle Professional ed Enterprise sono a pagamento. Quella usata per lo sviluppo del progetto di questa tesi è la versione Community.

4.4.3 SteamVR

Per permettere l'integrazione e l'utilizzo dell'HTC Vive Pro [27] all'interno del progetto Unity [16] è stato utilizzato il plugin di SteamVR [69], parte di Steam e specifico per la Realtà Virtuale, scaricabile gratuitamente dall'Asset Store di Unity [88].

Avendo il campo visivo completamente oscurato a causa del caschetto indossato, risulta sicuramente vantaggioso poter vedere le proprie mani virtuali all'interno dell'ambiente. Il plugin permette, quindi, di avere una rappresentazione dei controller e delle mani, aspetto che porta dei vantaggi sia dal punto di vista funzionale che da quello immersivo.

La parte principale di SteamVR [69] consiste nella possibilità di definire in maniera semplice delle Actions all'interno di Unity che possono poi essere mappate sui

pulsanti del controller che si sta utilizzando in modo che quando questi vengono premuti venga attivato il comportamento corrispondente. Il pannello in cui definire le Actions lo si trova in Unity sotto la voce Window → SteamVR Input (figura 4.12).

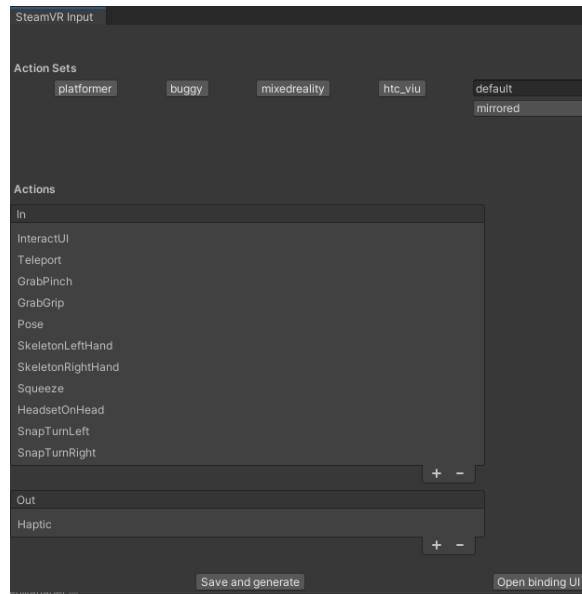


Figura 4.12: pannello SteamVR Input in cui poter definire delle Action da mappare sui pulsanti del controller

SteamVR [69] offre sei tipi di input:

1. Boolean: le Boolean Actions possono essere true o false. A questa categoria appartengono tutte quelle azioni in cui ci sono solo due stati possibili;
2. Single: le Single Actions sono azioni in cui si ha un valore analogico, compreso tra 0 e 1; si tratta di azioni utilizzate nel caso in cui gli stati possibili non siano solo due come nel caso precedente;
3. Vector2: le Vector2 Actions sono caratterizzate dall'avere una combinazione di due valori analogici, una X e una Y;
4. Vector3: le Vector3 Actions sono meno comuni. Sono simili alle precedenti ma presentano tre valori analogici, una X, una Y e una Z;
5. Pose: le Pose Actions sono caratterizzate da valori analogici riferiti a posizione e rotazione 3D; vengono utilizzate per tracciare i controller nello spazio;
6. Skeleton: le Skeleton Actions sfruttano lo SteamVR Skeleton Input, ottenendo una miglior stima circa l'orientamento delle dita delle mani nel momento in cui si impugnano i controller.

SteamVR [69] offre, inoltre, non solo la possibilità di avere un modello per i controller ma anche uno per le mani in modo da rendere l'esperienza ancora più realistica. Attraverso SteamVR Skeleton Input viene fornita una stima per la posizione e rotazione

dei giunti delle mani personalizzata in base al controller che si sta utilizzando. Le informazioni si possono ottenere in due modalità:

1. WithController: permette di sfruttare al meglio le informazioni che il controller mette a disposizione;
2. WithoutController: offre un range di posizioni che va dalla mano completamente aperta al pugno chiuso.

In alcuni casi può tornare utile avere un'informazione generale di quanto il dito sia o meno piegato, il che viene mappato su un range di valori che va da 0 a 1; stesso discorso lo si può applicare per quanto riguarda la distanza tra due dita. Ci sono, inoltre, diversi livelli di accuratezza con cui i controller tracciano i giunti:

1. Estimated: le parti del corpo non sono determinate direttamente ma attraverso una stima basata sulle informazioni derivanti da pulsanti o joystick. In questo caso rientrano i Vive Controller;
2. Partial: in questo caso si effettua una misura diretta delle parti del corpo ma non con tutti i DoF che si avrebbero nella realtà; in questo campo rientrano i guanti che misurano solo, ad esempio, di quanto sono piegate le dita;
3. Full: in questo caso ci si riferisce a sistemi di Motion Capture o guanti che rilevano dati in corrispondenza di ogni giunto delle dita; è la situazione in cui si ottengono i risultati migliori.

L'oggetto principale che permette di simulare la presenza dell'utente all'interno del mondo virtuale e consente di interagire con gli oggetti ad esso appartenenti è il Prefab Player. Contiene al suo interno sia elementi che simulano il punto di vista dell'utente sull'ambiente digitale, in base alla posizione che assume in quello reale, sia oggetti che simulano le mani dell'utente con alcuni script che permettono l'interazione.

Di oggetti Player non dovrebbero essercene più di uno all'interno di una scena Unity [16]. L'interazione con elementi virtuali avviene tramite il componente Hand che presenta alcune funzioni che vengono chiamate nel momento in cui viene identificato un oggetto interagibile. Queste funzioni sono:

1. OnHandHoverBegin(): la mano si posiziona su un oggetto interagibile;
2. HandHoverUpdate(): viene invocata ogni frame mentre la mano è sull'oggetto interagibile;
3. OnHandHoverEnd(): invocata quando la mano smette di essere sull'oggetto interagibile;
4. OnAttachedToHand(): chiamata quando l'oggetto viene attaccato alla mano;
5. HandAttachedUpdate(): invocata ogni frame in cui l'oggetto è attaccato alla mano;
6. OnDetachedFromHand(): chiamata quando l'oggetto viene staccato dalla mano;
7. OnHandFocusLost(): chiamata se l'oggetto attaccato perde il focus a causa di un altro oggetto che viene attaccato alla mano;
8. OnHandFocusAcquired(): chiamata quando un oggetto ottiene il focus dopo che l'oggetto che lo aveva in precedenza viene staccato dalla mano.

Il componente fondamentale affinché un oggetto venga identificato come interagibile è *Interactable*. A questo possono affiancarsi altri component, non obbligatori, per poter sfruttare altri comportamenti come *Throwable* (che permette di lanciare un oggetto nel momento in cui questo viene staccato dalla mano), *Velocity Estimator* (stima velocità e accelerazione di un oggetto ricavandole dai cambiamenti di posizione dell'oggetto stesso), *InteractableHoverEvents* a cui si possono associare eventi Unity chiamati nel momento in cui si ricevono determinati messaggi dalla mano.

Il componente *Steam VR_Skeleton_Poser*, infine, fa sì che la mano assuma una posa coerente con la forma dell'oggetto nel momento in cui questo viene afferrato. Sono già presenti delle pose ma se ne possono definire anche di personalizzate tramite il *Pose Editor* (figura 4.13).

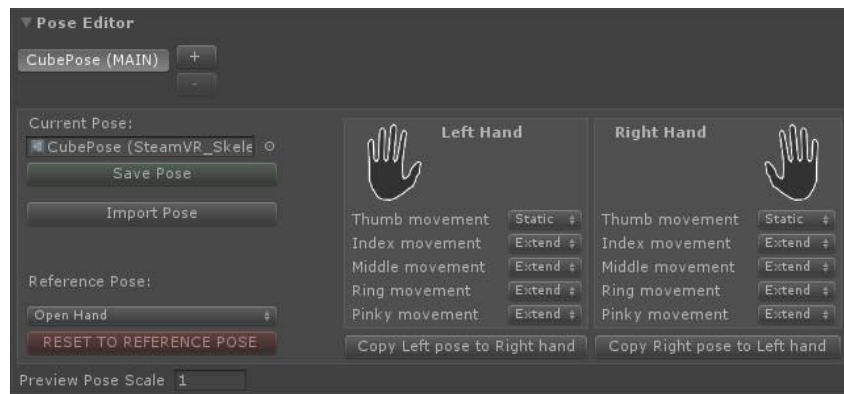


Figura 4.13: Pose Editor del componente *SteamVR_Skeleton_Poser*

Alle dita, inoltre, può essere data una libertà di movimento definita da uno dei seguenti parametri:

1. *Static*: non c'è libertà di movimento, si utilizza esclusivamente la posa;
2. *Free*: c'è piena libertà di movimento, si ignora la posa;
3. *Extend*: c'è libertà di movimento delle dita in estensione mentre non possono contrarsi oltre la posizione della posa;
4. *Contract*: opposta alla precedente, libertà di movimento in contrazione ma non in estensione per la quale si fa riferimento alla posa.

4.4.4 MANUS Core e MANUS Dashboard

In caso di utilizzo dei MANUS Prime II [14] risulta fondamentale l'utilizzo di Manus Core [89], la Messenger Application che si occupa di raccogliere i dati catturati dai sensori posti all'interno dei guanti e li rende disponibili per essere trasmessi ed

utilizzati all'interno di software esterni come Unity [16]. Manus Core [89] viene eseguita in background mentre l'applicazione front-end corrispondente è la MANUS Dashboard.

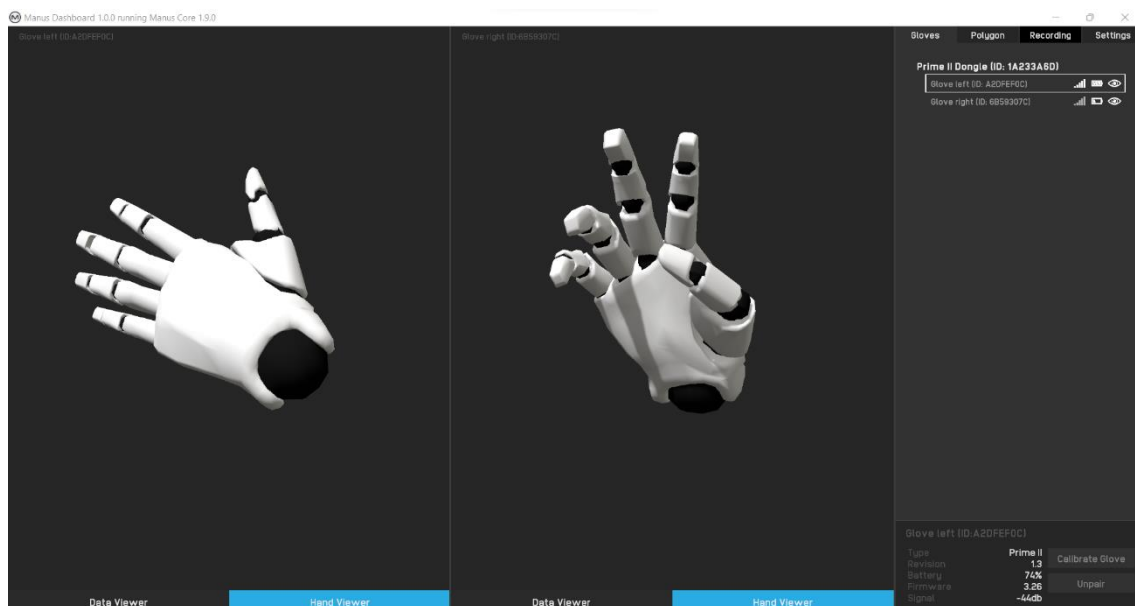


Figura 4.14: interfaccia della MANUS Dashboard. Fonte: urly.it/3t51s

Aperto la MANUS Dashboard (figura 4.14), se non è ancora stato inserito il dongle fornito con i guanti al momento dell'acquisto, comparirà la scritta “No dongles connected”. Inserito il dongle, se i guanti sono ancora spenti verrà visualizzata la voce “No gloves connected”; accesi i guanti sarà necessario accoppiarli con il dongle mettendoli in Pairing Mode. Una volta accoppiati verranno visualizzati i modelli 3D delle mani che riproducono la posa assunta dall'utente nella realtà.

Per migliorare il risultato del tracciamento è opportuno effettuare una fase di calibrazione disponibile direttamente all'interno della MANUS Dashboard poiché i guanti sanno come le dita si stanno muovendo ma non conoscono la loro dimensione. Per ogni guanto, infine, vengono fornite indicazioni su stato della sua batteria, qualità della connessione e versione del firmware con il quale il dongle sta lavorando.

La MANUS Dashboard presenta diversi tab:

1. **Gloves:** si tratta dell'interfaccia appena descritta, mostra un'anteprima della riproduzione della posa delle mani assunta dall'utente e fornisce informazioni riguardo lo stato dei guanti, il tipo di guanti e il firmware;
2. **Polygon:** in questa sezione è possibile gestire eventuali tracker utilizzati per il tracciamento di alcune parti del corpo nel caso si vogliano avviare delle sessioni di Motion Capture. Nel caso del progetto sviluppato per questa tesi, in questo tab venivano mostrati tre tracker: l'HMD indossato e i due Vive Trackers [71] in corrispondenza dei polsi in modo da tracciare la posizione dei guanti nello spazio.

Per ottenere informazioni corrette, i tracker disponibili devono essere associati alla corrispondente parte che si vuole tracciare (i Vive Tracker [71] sono stati associati alle mani mentre l'HTC Vive Pro [26] alla testa);

3. Settings: in questo tab è possibile modificare le impostazioni dell'applicazione. È possibile decidere di disabilitare il tracciamento tramite cui ricavare la distanza tra due dita, è possibile ruotare il modello 3D delle mani per cambiare il punto di vista. Nel caso in cui si utilizzi un sistema di tracciamento questo deve essere selezionato tra quelli disponibili in modo che la Dashboard mostri i tracker in utilizzo: in questo caso, utilizzando il sistema basato su SteamVR [69], bisogna selezionare la voce OpenVR dall'elenco dei sistemi di tracking disponibili;
4. Recording: è il tab che permette la registrazione di animazioni sfruttando i dati provenienti sia dai tracker sia dai guanti. È possibile esportare in un file formato FBX tutte le informazioni registrate.

4.4.5 MANUS Unity Plugin

Attraverso il plugin fornito da MANUS [14], è possibile gestire in maniera semplice la trasmissione in Unity dei dati raccolti da Manus Core [89].

Gli script ManusManager e CommunicationHub sono quelli che assicurano che i dati raccolti da Manus Core [89] vengano inviati a Unity [16]; se non sono presenti all'interno della scena vengono istanziati in automatico nel momento in cui si fa Play.

Il componente assegnato alla root del GameObject che si vuole utilizzare come mano è chiamato Hand e riceve i dati provenienti dai sensori dei guanti. Gli oggetti in Unity [16] che presentano lo script Hand vengono registrati nel CommunicationHub in modo tale da ricevere i dati catturati da Manus Core.

Ricevere solo i dati non basta però, per poter effettivamente vedere il modello della mano che si muove in accordo con la posa assunta nella realtà è necessario associare all'oggetto mano lo script Hand Animator. Anche questo deve essere assegnato al GameObject usato come mano o ad un suo parent. Necessita di un breve setup:

1. Innanzitutto, bisogna settare l'Hand Model Type, ovvero specificare se quel GameObject sarà una mano destra o sinistra;
2. Assegnare il Wrist Bone, ovvero il polso che dovrebbe essere l'osso parent di tutte le ossa delle dita;
3. È possibile popolare i campi inerenti le ossa delle dita manualmente oppure in maniera automatica cliccando sulle voci Index o Names;
4. Premendo su Calculate Axes, verranno generate correttamente le informazioni sulla rotazione delle ossa.

Per quanto riguarda il riconoscimento delle pose esiste una classe base, *GestureBase*, che contiene il metodo *Evaluate()*: questo non fa altro che, quando viene invocato, verificare che vengano soddisfatte le condizioni affinché si possa dire che una determinata posa è stata riprodotta e innescare il comportamento definito. Qualunque script si voglia creare per il riconoscimento di una posa, questo deve ereditare da *GestureBase* per poter implementare il metodo *Evaluate()* secondo le proprie esigenze. Per determinare se viene riprodotta la posa di *Grab*, ad esempio, è stato creato uno script chiamato *HandGrabInteraction* che verrà descritto più nel dettaglio nel capitolo successivo.

Per quanto riguarda gli oggetti afferrabili, invece, questi devono avere lo script chiamato *GrabbableObject*, il quale eredita l'interfaccia *IGrabbable* incaricata di assegnare automaticamente lo script *GrabbedObject* quando un oggetto viene afferrato e di eliminarlo quando viene rilasciato.

Il componente *HandCollision* è incaricato di verificare che sia avvenuta una collisione tra un oggetto del mondo virtuale e quello a cui lo script è assegnato (o uno dei suoi figli), in questo caso la mano.

4.4.6 Motive

Motive [90] (schermata di esempio mostrata in figura 4.15) è il software di motion capture che, affiancato all'utilizzo di camere *OptiTrack*, permette di avere ottimi risultati sia per quanto riguarda il tracciamento di oggetti sia per quello di figure umane. Fornisce dati, infatti, con un'accuratezza di $\pm 0.2\text{mm}$ in merito alla posizione, di $\pm 0.1^\circ$ per la rotazione ed ha una latenza inferiore ai 9ms.

I dati catturati dalle camere e mostrati in *Motive* [90] possono essere utilizzati in software esterni: nel caso di questo progetto, i dati sul tracciamento di determinati elementi sono stati utilizzati all'interno di *Unity* [16] per adattare l'applicativo basato sui *MANUS Gloves* [14] a funzionare all'interno del *CAVE*; in particolare gli oggetti tracciati di interesse sono gli occhiali 3D, utilizzati per la visione stereoscopica delle immagini proiettate sugli schermi, e i due guanti per ricavare la posizione delle mani all'interno dell'area di cattura.

Analizzando i dati provenienti dalle camere, il software permette di calcolare la posizione dei marker da tracciare. Di marker se ne possono avere di due tipi:

1. Passivi (o retroriflettenti): hanno forme di sfere, semisfere e dischi piatti che riflettono i raggi infrarossi diffusi nello spazio dalle camere; nel momento in cui i marker vengono colpiti da questi raggi diventano una sorta di spot luminoso per le camere e, sfruttando le informazioni di un certo numero di camere, è possibile ricostruire la posizione del marker nello spazio;

2. Attivi: in questo caso sono i marker stessi che generano raggi infrarossi che vengono riconosciuti dalle camere e grazie ai quali riescono a ricavarne la posizione. Consentono di ottenere risultati migliori rispetto a quelli passivi, con tempi di acquisizione decisamente minori (inferiori a 0.1s).

Il software presenta i seguenti pannelli:

1. Devices Pane: in questo pannello vengono mostrate le camere presenti nel sistema; qui è possibile settarne le impostazioni (come fps, esposizione) e decidere per ciascuna camera se utilizzarla per il tracking o per l'acquisizione di video di riferimento in scala di grigi;
2. Properties Pane: qui, nel momento in cui viene selezionato un oggetto all'interno del software, vengono mostrate tutte le sue proprietà. Questo vale sia per asset, come Skeleton e RigidBody, sia per le camere e qualunque altro dispositivo sia connesso;
3. View Pane: lo si può dividere in due parti. Quella superiore viene chiamata Perspective Viewport, al suo interno viene mostrato il volume di acquisizione ricavato dopo il processo di calibrazione e tutti gli oggetti presenti al suo interno visibili alle camere; in questa parte è possibile selezionare un oggetto e settarne le sue proprietà.
La parte inferiore, invece, è chiamata Camera Viewport e al suo interno è possibile tenere sotto controllo la vista di ciascuna camera presente nel sistema. Viene utilizzato per esaminare i marker, le luci a infrarossi presenti nell'area ed applicare eventuali maschere per eliminare fonti di disturbo per il processo di tracciamento;
4. Calibration Pane: è il pannello tramite cui è possibile calibrare le camere così da ricostruire l'ambiente all'interno del quale è possibile tracciare degli oggetti o delle figure umane;
5. Graph View Pane: fornisce una rappresentazione grafica delle coordinate di posizione e orientamento degli oggetti tracciati all'interno dell'ambiente di cattura;
6. Control Deck: da qui è possibile monitorare la registrazione (nel caso in cui si lavori in Live Mode) o la riproduzione di dati (nel caso in cui si lavori in Edit Mode).

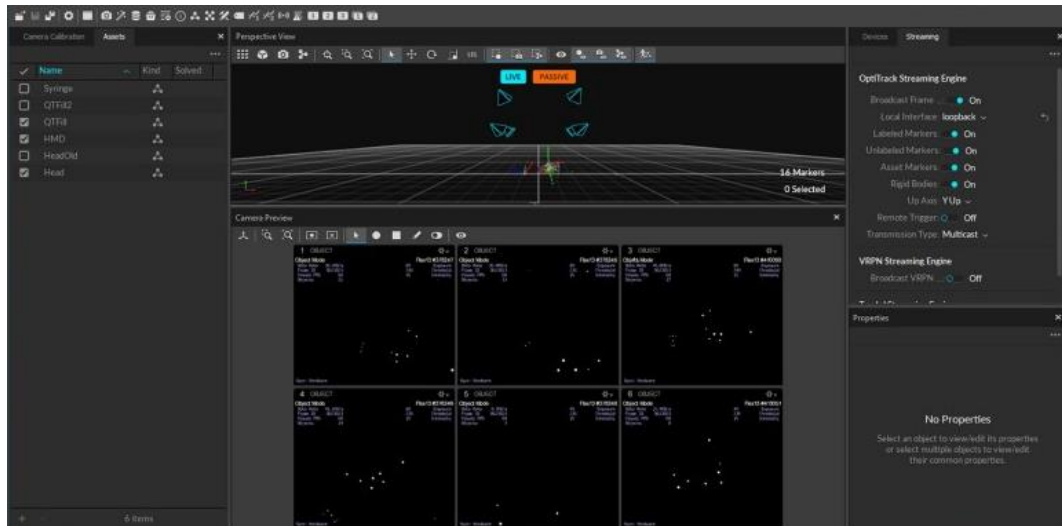


Figura 4.15: interfaccia del software Motive. Fonte: urly.it/3t52j

Per poter ottenere dei dati coerenti tali da poter essere utilizzati in Unity [16] è importante la calibrazione dello spazio di cattura (ovvero il CAVE, in questo caso), processo che può essere svolto direttamente all'interno di Motive [90], grazie al Calibration Pane, seguendo i seguenti step:

1. Masking: prima di iniziare la calibrazione è importante andare a rimuovere tutte quelle fonti che potrebbero interferire con il tracciamento; se non è possibile rimuoverle fisicamente allora si procede mascherandole in modo che le camere li ignorino;
2. Wanding: si utilizza uno strumento prodotto da OptiTrack [81], la Calibration Wand [91] (figura 4.16), che presenta dei marker a distanze reciproche note. Questo step consiste nel muoversi all'interno del volume di acquisizione muovendo la Calibration Wand e disegnando una figura di otto in modo tale da coprire quanta più superficie possibile;



Figura 4.16: Calibration Wand. Fonte: urly.it/3t52q

3. Floor Calibration: l'ultimo step consiste nella calibrazione del pavimento tramite il Calibration Square [92] (figura 4.17). Si tratta di un oggetto, anch'esso di casa

OptiTrack [81], con dei marker che va posizionato in modo che il marker al vertice sia esattamente nello origine del sistema di riferimento (del CAVE in questo caso). L'asse Z verrà orientato lungo il braccio più lungo del Calibration Square, l'asse X verso quello più corto e l'asse Y verrà ricavato con la regola della mano destra.



Figura 4.17: Calibration Square. Fonte: urly.it/3t52w

Motive [90] permette di creare due tipologie di asset: Skeleton e Rigidbody. I primi vengono utilizzati per il tracciamento del movimento di figure umane mentre i secondi per tracciare qualunque tipo di oggetto. In questo progetto sono stati usati tre Rigidbody: uno per gli occhiali 3D e uno per ciascuna mano. Basta posizionare i marker sull'oggetto in questione e questi compariranno nella Perspective View del software; da qui è possibile selezionarli per creare un nuovo Rigidbody a cui assegnare un nome (importante perché sarà grazie a questo che i dati verranno inviati al GameObject corretto in Unity [16], come spiegato nel capitolo successivo).

Lo streaming dei dati da Motive [90] a Unity [16] avviene tramite VRPN, interfaccia di rete indipendente dal dispositivo, molto diffusa nel campo della VR che permette di codificare e inviare dati riguardanti il tracciamento di oggetti. Si basa su una connessione tra un VRPN Client e un VRPN Server e la comunicazione può avvenire sfruttando come protocolli il TCP o l'UDP.

4.4.7 Mirror Unity Library

Per lo sviluppo del template realizzato da Reply [84] tramite il quale è possibile far funzionare e visualizzare correttamente un progetto Unity [16] all'interno del CAVE è stata utilizzata una libreria messa a disposizione da Unity, la libreria Mirror [93] che viene impiegata per implementare il networking. Si tratta di una libreria che permette di gestire delle funzionalità utili per lo sviluppo di applicazioni multiplayer come gestione

dei messaggi in rete, serializzazione, gestione degli oggetti presenti nell'applicazione, classi per la gestione di server e client e dei componenti che ne permettono la sincronizzazione.

Come detto prima, ci sono due tipologie principali di attori all'interno di sistemi multiplayer:

1. Server: si tratta di un'istanza dell'applicazione alla quale si connettono tutti gli altri utenti che intendono usufruirne; si occupa, inoltre, della gestione generale del sistema e comunica lo stato ai client in modo che il tutto sia sincronizzato.
Il server può essere un server dedicato o un host: nel primo caso, si tratta di avere un'istanza del sistema che svolge unicamente la funzione di server; nel secondo caso, invece, si ha una singola istanza del sistema che prende il nome di host e che svolge sia la funzione di server che quella di client;
2. Client: sono istanze del sistema che si possono connettere al server da diversi pc, sia su una rete online che su una locale (nel caso del progetto sviluppato in questa tesi la connessione avviene su una rete locale).

Una delle funzionalità di Mirror [93] è il fatto di gestire la creazione dinamica di oggetti all'interno dell'applicazione, funzionalità tra le centrali del progetto sviluppato. La creazione dinamica di oggetti, infatti, non avviene semplicemente chiamando il metodo `Instantiate()` che si utilizzerebbe nel caso di sistemi single player; per sistemi multiplayer si utilizza il sistema di spawning messo a disposizione da Mirror [93] il quale si occupa non solo di creare degli oggetti ma anche di renderli disponibili in rete in modo che siano visibili non solo al server (che è colui che li crea) ma anche ai client ad esso connessi.

Capitolo 5

Design e sviluppo

5.1 Progetto per HMD

Come anticipato nei capitoli precedenti, con il progetto in questione si intende simulare il processo di prototipazione di un LCA, indipendentemente dal sistema di interazione che si utilizza. Per fare questo si può attingere ad una libreria di oggetti selezionabili per andare a comporre la struttura finale.

Il funzionamento del sistema, infatti, si basa su tre tipologie di oggetti:

1. Oggetti singoli: sono quei componenti direttamente selezionabili dal menu principale che permette di istanziarli all'interno dell'ambiente virtuale;
2. Oggetto composti: si tratta di due o più oggetti singoli agganciati tra loro in base alle regole definite all'interno del sistema. In particolare, gli oggetti singoli possono avere dei Collider Points che rappresentano punti di aggancio ai quali è possibile unire altri oggetti;
3. Strutture complesse: si tratta di due o più oggetti composti agganciati tra loro seguendo le regole definite all'interno del sistema. Una struttura è identificata attraverso un valore numerico che viene assegnato a tutti gli elementi che ne fanno parte in modo che, nel momento in cui uno di essi viene afferrato per essere spostato, tutti gli altri che hanno il suo stesso codice struttura vengono spostati in maniera solidale.

Di seguito sono riportate le proprietà che caratterizzano gli oggetti singoli, indicando anche a quali altri oggetti singoli possono legarsi per formarne di composti:

1. Tubo: posizione, rotazione, dimensione. Può unirsi ad una Giunzione o Ruota;
2. Giunzione: posizione, rotazione. Può unirsi ad un Tubo;
3. Giunzione a 2: posizione, rotazione;
4. Ruota: posizione, rotazione. Può unirsi ad un Tubo;
5. Base Rulliera: posizione, rotazione, dimensione. Può unirsi a Gancio Rulliera;
6. Freno: posizione, rotazione. Può unirsi all'oggetto composto LCABraccio;
7. Ruote Rulliera: posizione, rotazione. Può unirsi all'oggetto composto LCARulliera;
8. Gancio Rulliera: posizione, rotazione. Può unirsi a Base Rulliera;
9. Odette: posizione, rotazione, dimensione.

Il sistema prevede, inoltre, i seguenti oggetti composti:

1. LCABraccioIncompleto: è formato da un Tubo e una Giunzione;

2. LCABraccio: è formato da un Tubo e due Giunzioni. Può unirsi a LCARuota per formare una struttura complessa;
3. LCARullieraIncompleta: formato da Base Rulliera e un Gancio Rulliera;
4. LCARulliera: formato da Base Rulliera e due Ganci Rulliera;
5. LCARullieraConRuote: formato da Base Rulliera, due Ganci Rulliera con l'aggiunta delle Ruote Rulliera. Può unirsi a LCABraccio per formare una struttura complessa;
6. LCARuota: formato da Tubo e Ruota. Può unirsi a LCABraccio per formare una struttura complessa.

La figura 5.1 riporta gli elementi presenti all'interno della scena in Unity [16], i quali verranno analizzati in dettaglio nei paragrafi successivi.

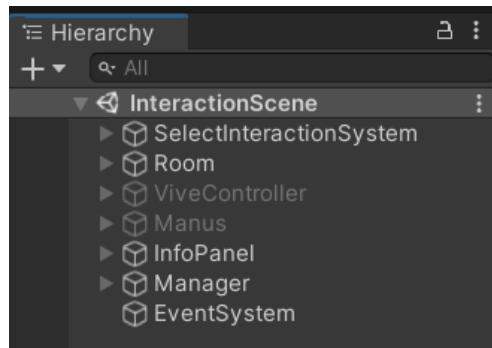


Figura 5.1: gerarchia oggetti presenti nella scena Unity

5.1.1 Controllo del sistema

HTC Vive Controller

In base al sistema di interazione che si decide di utilizzare, le azioni all'interno del sistema vengono innescate in maniera diversa.

Nel caso degli HTC Vive Controller [67], sono state definite delle azioni all'interno del pannello SteamVR Input, visualizzabile in Window → SteamVR Input (figura 5.2). Al suo interno è presente l'elenco con tutte le azioni che sono state definite con la possibilità di modificarne le caratteristiche o di crearne di nuove tramite il pulsante “+” nella sezione “In” della tabella. Creata una nuova azione, è necessario definirne un nome, la tipologia (Boolean, Vector1, Vector2, Vector3, Pose o Skeleton) e indicare se si tratta di un'azione che all'interno del sistema è obbligatoria, suggerita o opzionale.

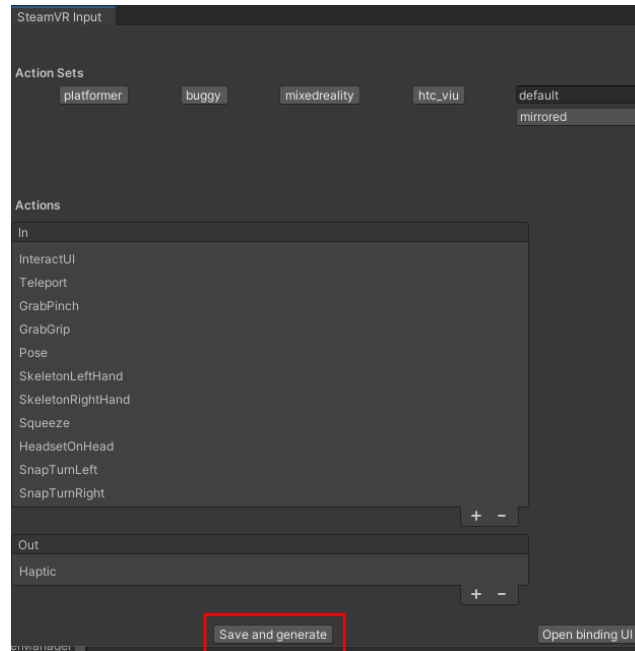


Figura 5.2: pannello per la creazione e il salvataggio di azioni da mappare sui pulsanti dei controller

Create tutte le azioni necessarie, attraverso il pulsante Open Binding UI, si aprirà una pagina nel browser (mostrata in figura 5.3) all'interno della quale è possibile scegliere il tipo di controller che si sta utilizzando e associare ciascuna delle azioni che sono state definite in Unity [16] ai pulsanti corrispondenti sul controller stesso. A questo punto basterà cliccare su Save and Generate e la mappatura azioni – pulsanti è terminata.

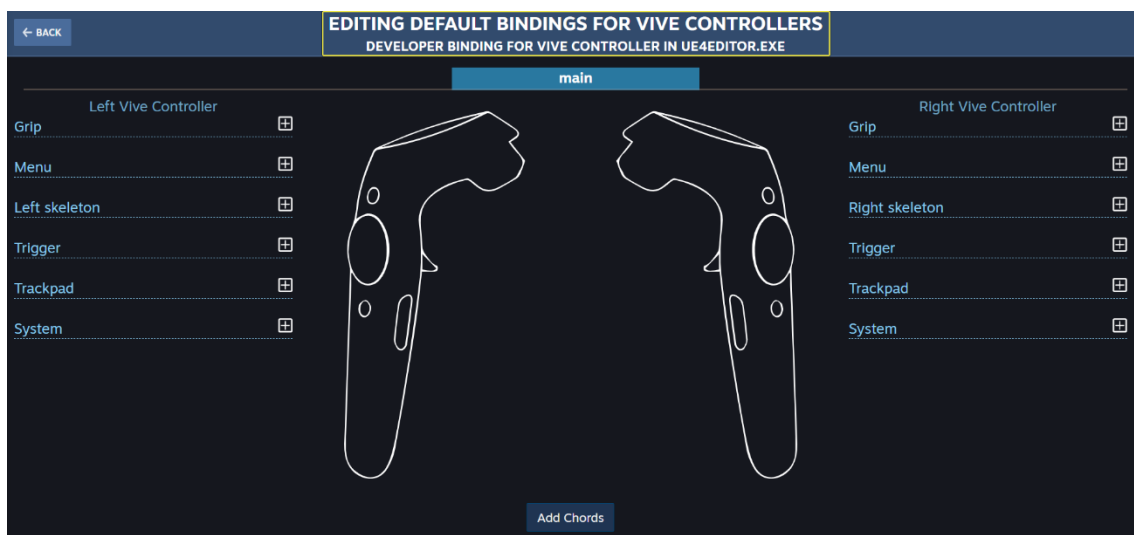


Figura 5.3: interfaccia che permette di associare ai pulsanti dei controller le Action definite in Unity

All'interno del progetto è presente uno script chiamato `LCAInputManager` che si occupa di innescare le funzioni corrispondenti al tasto del controller che si è premuto. Nel caso di questo sistema le azioni sono così mappate:

1. **Menu:** utilizzato per la gestione dei menu. Nel momento in cui viene premuto, i menu disponibili nell'applicazione vengono aperti o chiusi in base allo stato in cui il sistema si trova;
2. **Trigger:** è il tasto presente sul retro del controller e, quando premuto, permette di iniziare ad afferrare un oggetto mentre quando viene rilasciato l'oggetto viene staccato dalla mano;
3. **Grip:** è il tasto presente sui lati del controller e, quando premuto mentre si punta un oggetto composto parte di una struttura complessa, permette di staccarlo da tale struttura;
4. **Touch:** è il tasto centrale presente nella parte anteriore del controller e, quando premuto insieme al Trigger mentre si punta un oggetto composto all'interno di una struttura complessa, permette di trascinare verso l'alto o verso il basso tale oggetto composto facendo restare invariata la posizione della restante struttura.

Per quanto riguarda la selezione delle voci appartenenti ai vari pannelli presenti nel sistema, queste possono essere selezionate andando semplicemente a far collidere la mano virtuale con la voce desiderata.

MANUS Prime II

Nel caso si utilizzino i guanti MANUS [14] come sistema di interazione, i vari comportamenti all'interno del sistema vengono innescati in base alla posa della mano che l'utente riproduce. In particolare sono tre le pose definite:

1. **Open Hand Pose** (figura 5.4): viene riconosciuta quando l'utente apre la mano completamente. È utilizzata per aprire il MainMenu nella scena;

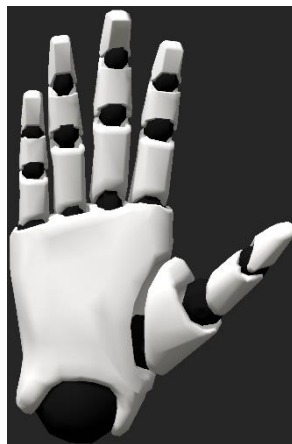


Figura 5.4: Open Hand Pose

2. Grab Pose (figura 5.5): viene riconosciuta quando l'utente chiude la mano a pugno ed è utilizzata per iniziare ad afferrare l'oggetto nelle immediate vicinanze della mano. Nel momento in cui viene aperta, l'oggetto viene rilasciato;



Figura 5.5: Grab Pose

3. Pointing Pose (figura 5.6): viene riconosciuta quando l'utente ha la mano chiusa ad eccezione del dito indice. Viene utilizzata per la selezione di voci all'interno dei menu oppure per aprire i pannelli con le informazioni specifiche di un oggetto nel momento in cui la posa viene riprodotta mentre il dito indice collide con tale oggetto. Per far sì che la posa venga considerata corretta solo quando è il dito indice ad essere disteso e non uno qualunque, alle parti della mesh della mano che costituiscono il dito è stato assegnato un tag differente rispetto al resto della mano: questa, infatti, ha un generico tag "Player", mentre l'indice della mano destra e sinistra hanno, rispettivamente, il tag "Hand_R" e "Hand_L".

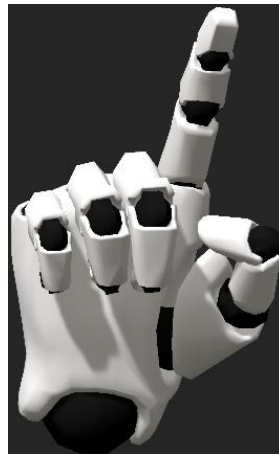


Figura 5.6: Pointing Pose

Indipendentemente dal sistema di interazione, invece, le collisioni tra il modello 3D della mano dell'utente e i vari oggetti presenti all'interno della scena vengono

riconosciute tramite la funzione `OnTriggerEnter()`, nel momento in cui la collisione ha inizio, e la funzione `OnTriggerExit()` nel momento in cui termina.

5.1.2 Select Interaction System

Si tratta di un pannello UI (mostrato in figura 5.7) tramite il quale è possibile, all'avvio dell'applicazione, selezionare il sistema di interazione che si vuole utilizzare, quindi gli HTC Vive Controller [67] o i MANUS Prime II [14].

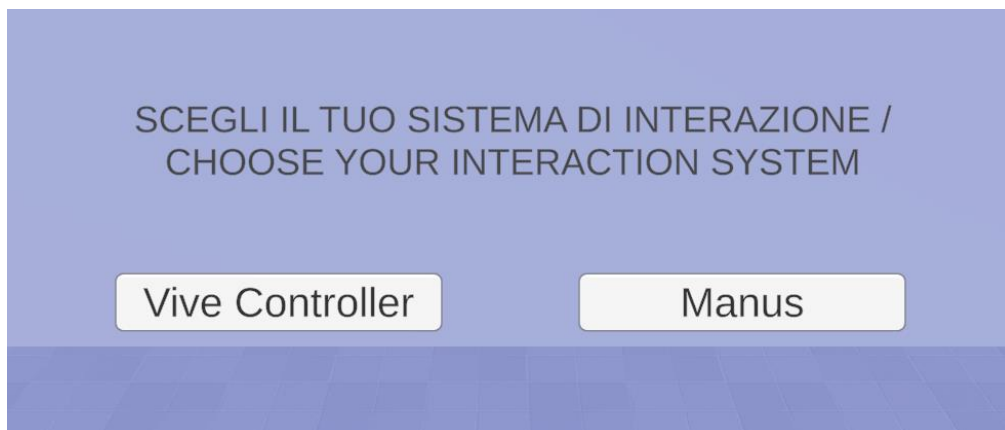


Figura 5.7: pannello per scegliere il sistema di interazione

Presenta due pulsanti al suo interno ai quali è stato associato uno script chiamato `InteractionModeSelector`: il metodo in esso contenuto, `ChooseInteractionMode()`, riconosce quale dei due pulsanti è stato cliccato, istanzia nella scena il Player corrispondente e i relativi pannelli del tutorial.

Il riconoscimento del pulsante cliccato avviene nel seguente modo: all'interno della scena è presente un `EventSystem` incaricato di gestire gli eventi in Unity [16]. I pulsanti hanno un componente, `Button`, nel quale è stato definito un evento che consiste nel chiamare il metodo `ChooseInteractionMode()` nel momento in cui si clicca su uno dei due bottoni. È possibile risalire a quale si è cliccato grazie alla proprietà `currentSelectedGameObject` della classe `EventSystem`.

5.1.3 Info Panel

Si tratta di un `GameObject` i cui figli sono tutti i menu che è possibile utilizzare all'interno del sistema (figura 5.8).

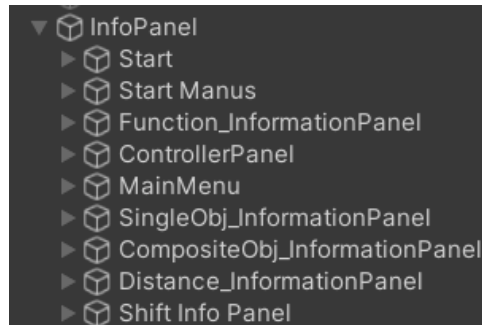


Figura 5.8: gerarchia degli elementi che compongono il GameObject Info Panel

I primi due presenti, Start e StartManus (rispettivamente figure 5.9 e 5.10), sono i pannelli che costituiscono il tutorial con la spiegazione dell'applicativo, personalizzata in base a quale sistema di interazione si utilizza. Per procedere, tornare indietro, saltare il tutorial o aprire il menu contenente le varie funzioni (e loro spiegazione) effettuabili sui singoli oggetti della scena sono stati inseriti rispettivamente, i tasti Next, Back, Skip Tutorial e Function List con i corrispondenti script NextButton, BackButton, SkipButton e FunctionListButton. La selezione di uno di questi elementi avviene andando a farli collidere con la mano virtuale per entrambi i sistemi di interazione.



Figura 5.9: introduzione tutorial per sistema basato sugli HTC Vive Controller



Figura 5.10: introduzione tutorial per sistema basato sui MANUS Prime II

Gli elementi della Function List, mostrata nella figura 5.11, hanno uno script chiamato InfoFunctionButtonSelection il quale si occupa di mostrare all'utente una breve spiegazione della funzione di cui si è deciso di avere maggiori informazioni; anche in questo caso la selezione avviene facendo collidere la mano virtuale con il nome della funzione.

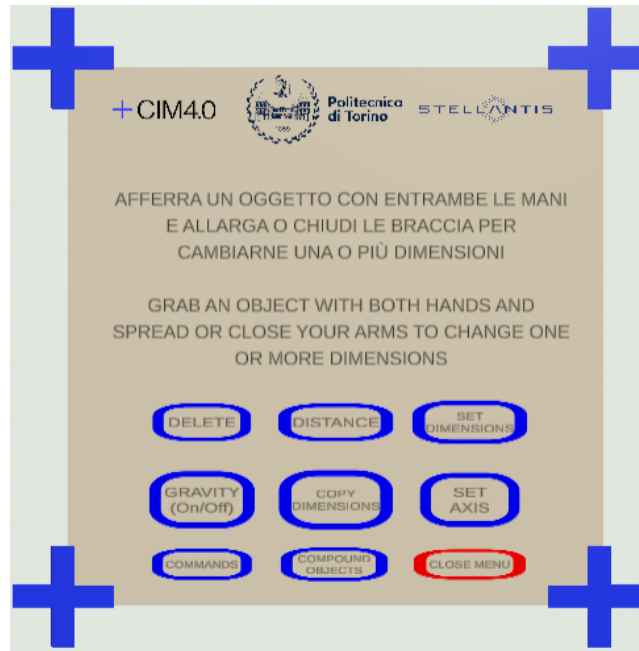


Figura 5.11: lista delle funzioni applicabili su oggetti singoli e/o composti

Nel caso in cui si stiano utilizzando gli HTC Vive Controller [67] come sistema di interazione, nel momento in cui si termina il tutorial viene mostrato un pannello (figure 5.12 e 5.14) che mostra i controller e il nome dei vari pulsanti che possiedono; attraverso di esso viene fornito all'utente un feedback in tempo reale su quale tasto stia premendo poiché lo sfondo del nome corrispondente viene colorato di giallo (figure 5.13 e 5.15).



Figura 5.12: pannello con riproduzione dei controller e relativi pulsanti, visione dall'editor di Unity



Figura 5.13: pannello dei controller mentre viene premuto il pulsante Touch, visione dall'editor di Unity



Figura 5.14: pannello con riproduzione dei controller e relativi pulsanti, visione dall'applicativo in esecuzione



Figura 5.15: pannello dei controller mentre viene premuto il pulsante Touch, visione dall'applicativo in esecuzione

Uno dei pannelli più importanti del sistema è il MainMenu (figure 5.16 e 5.17) attraverso il quale l'utente può istanziare oggetti all'interno del mondo virtuale. Al suo interno sono presenti gli oggetti singoli che è possibile creare richiamando, lato codice, il metodo Instantiate() ovvero il Tubo, la Giunzione, la Ruota, la Base Rulliera, il Freno, le Ruote Rulliera, il Gancio Rulliera e Odette.

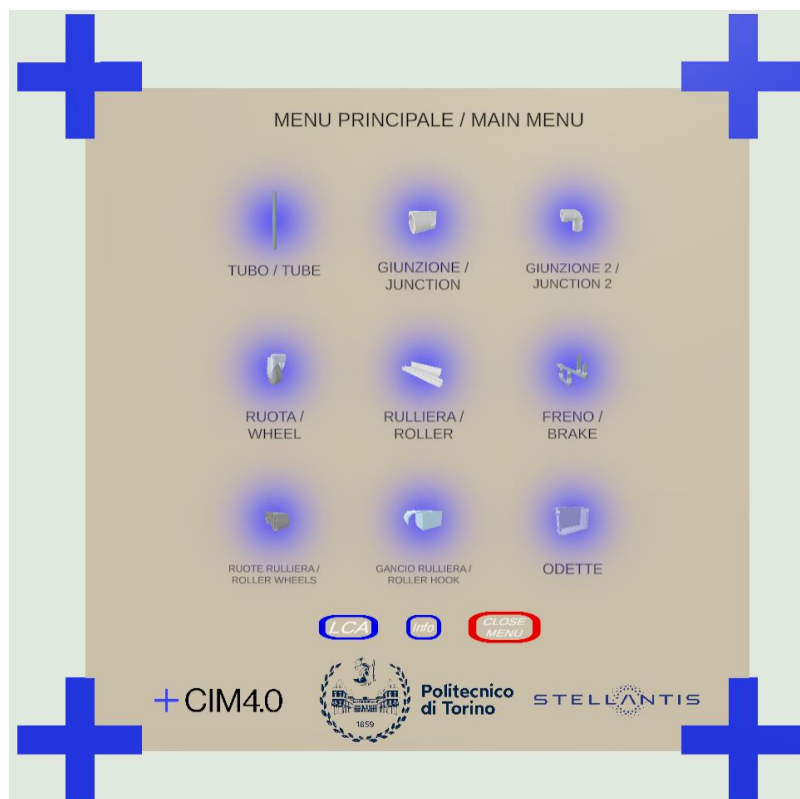


Figura 5.16: menu principale contenente gli oggetti istanziabili, visione dall'editor di Unity

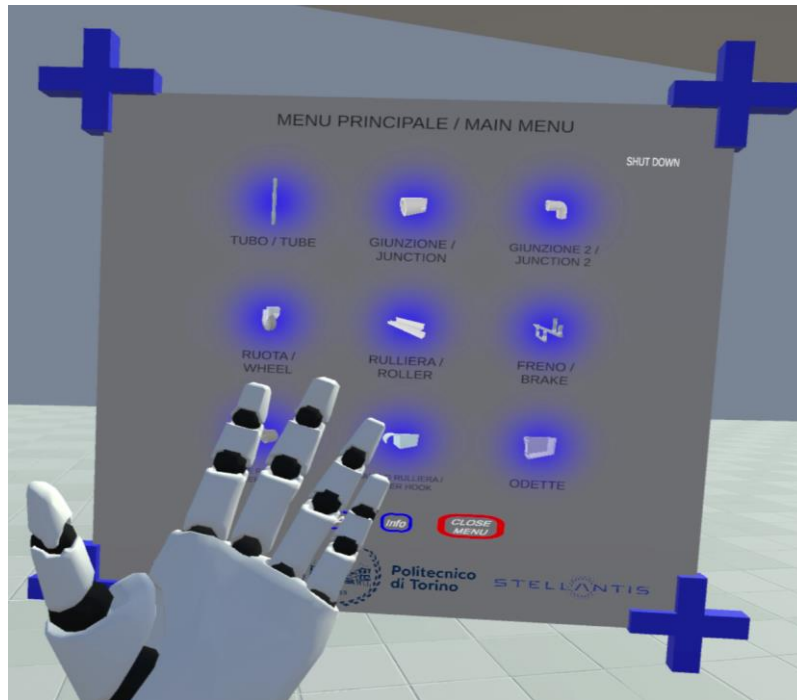


Figura 5.17: apertura del menu principale, visione dall'applicativo in esecuzione

Per inserirli all'interno della scena è sufficiente far collidere la mano con l'immagine dell'oggetto che si vuole creare e questo comparirà davanti all'utente.

Il menu principale presenta, inoltre, tre pulsanti:

1. LCA: selezionando questa voce comparirà all'interno dell'ambiente la struttura di un LCA di esempio che l'utente può provare a riprodurre;
2. Info: quando viene selezionato, l'utente vedrà nuovamente il pannello Function List di cui si è parlato in precedenza (figura 5.11);
3. Close Menu: presente solo se si utilizzano i MANUS Prime II [14] e nel momento in cui viene selezionato il menu si chiude. Nel caso in cui si utilizzino i controller Vive, la chiusura avviene premendo il tasto Menu, come spiegato nel paragrafo Controllo del Sistema.

Come detto in precedenza, il sistema non si basa solo sulla creazione di oggetti ma anche sulla loro manipolazione. Le funzioni applicabili sugli oggetti vengono mostrate in appositi pannelli i cui elementi mostrati cambiano in base al fatto che si tratti di un oggetto singolo o di uno composto.

Nel primo caso, il pannello di riferimento è il SingleObjInformationPanel, mostrato nelle figure 5.18 e 5.19.

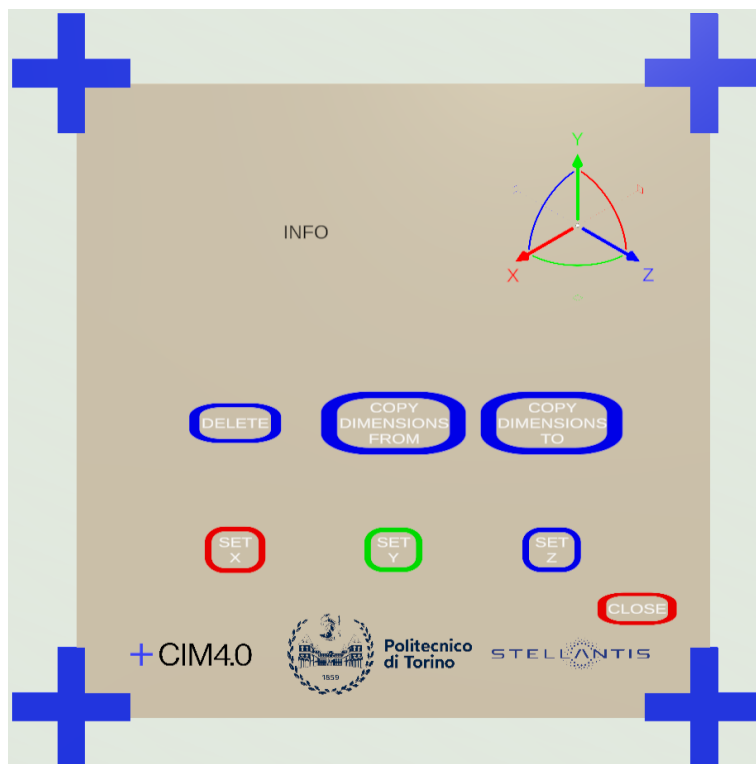


Figura 5.18: pannello con le informazioni riguardanti un oggetto singolo e le funzioni su di esso applicabili, visione dall'editor di Unity

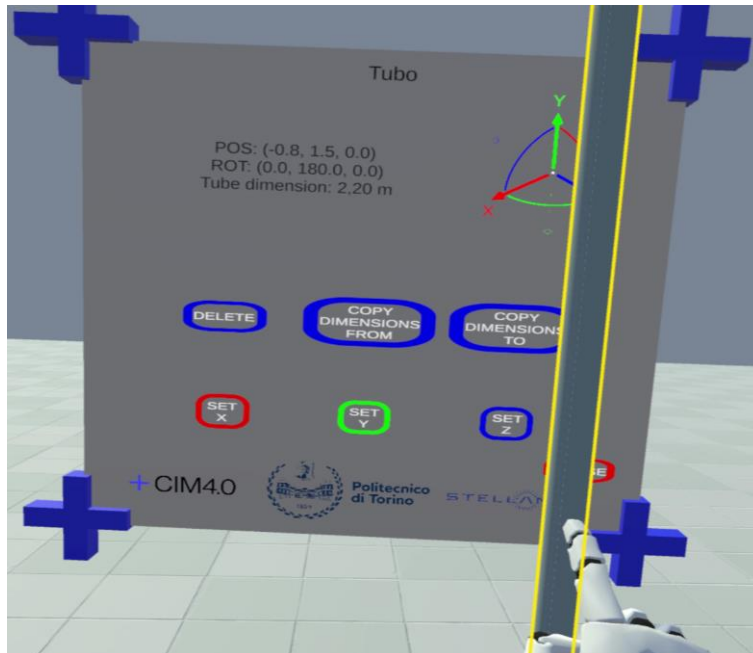


Figura 5.19: pannello con le informazioni riguardanti un tubo singolo e le funzioni su di esso applicabili, visione dall'applicativo in esecuzione

Nel momento in cui viene aperto in riferimento ad un oggetto singolo, al posto della voce INFO vengono mostrate informazioni riguardo posizione e rotazione dell'oggetto nello spazio e, in alcuni casi, il peso dell'oggetto e/o la sua dimensione. Le funzioni messe a disposizione per gli oggetti singoli sono:

1. Delete: permette di eliminare l'oggetto dall'ambiente;
2. Copy dimension from/to: permettono di copiare la dimensione di un oggetto su un altro dello stesso tipo;
3. Set X/Y/Z: permettono di copiare le coordinate x, y o z di un oggetto su un altro dello stesso tipo.

Nel caso in cui si consideri un oggetto composto, invece, il pannello di riferimento è chiamato CompositeObjInformationPanel, mostrato nelle figure 5.20 e 5.21.

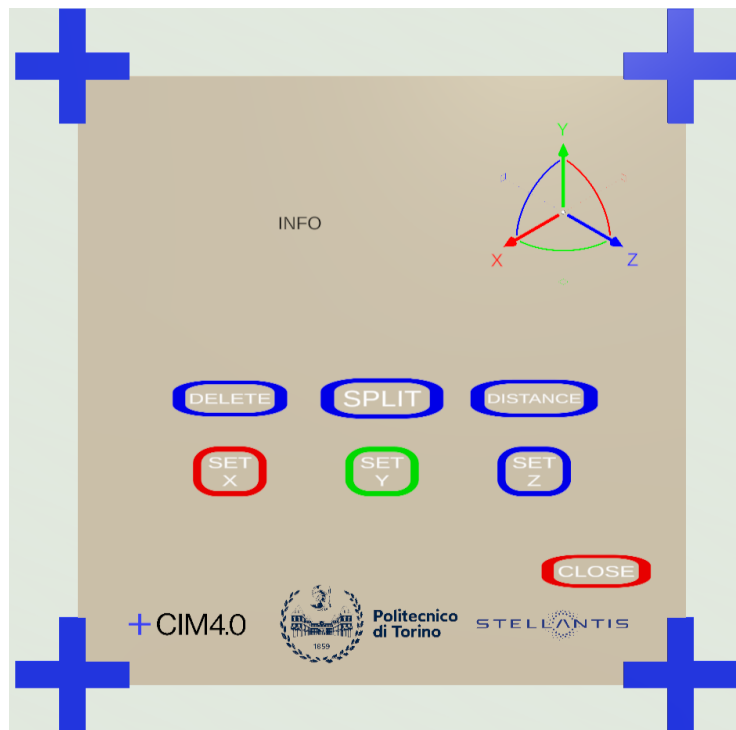


Figura 5.20: pannello con le informazioni riguardanti un oggetto composto e le funzioni su di esso applicabili, visione dall'editor di Unity

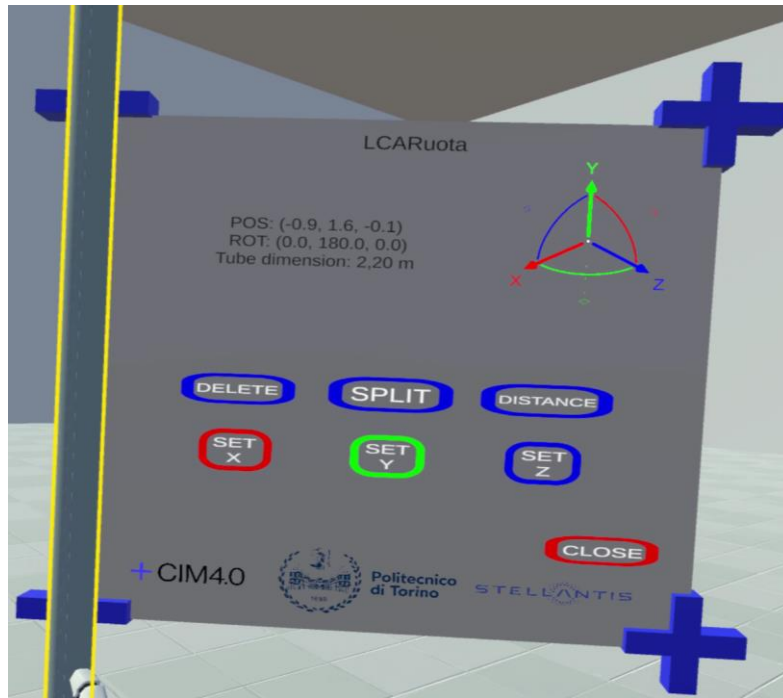


Figura 5.21: pannello con le informazioni riguardanti un LCARuota e le funzioni su di esso applicabili, visione dall'applicativo in esecuzione

Anche in questo caso, al posto della parola INFO vengono mostrate le stesse informazioni del pannello inerente gli oggetti singoli, quindi loro posizione e rotazione e, in alcuni casi, peso e/o dimensione. Anche le funzioni Delete e Set X/Y/Z svolgono gli stessi compiti illustrati precedentemente. Nel caso di oggetti composti, però, ne sono presenti altre:

1. Split: questa funzione permette di separare un oggetto composto nei vari oggetti singoli che lo compongono. In particolare, quello composto viene distrutto tramite il metodo Destroy() e vengono reistanziati i vari elementi che lo costituivano tramite il metodo Instantiate();
2. Distance: permette di ottenere informazioni riguardo la distanza (in metri) tra un oggetto A e un oggetto B.

Il meccanismo di tutte le funzioni, fatta eccezione per Delete e Split, è il seguente: nel momento in cui viene selezionata una funzione da un pannello, che sia quello relativo ad oggetti singoli o quello relativo ad oggetti composti, ne compare uno nuovo contenente il nome della funzione selezionata e una sfera gialla, anch'essa afferrabile, chiamata Highlighter (figure 5.22 e 5.23).

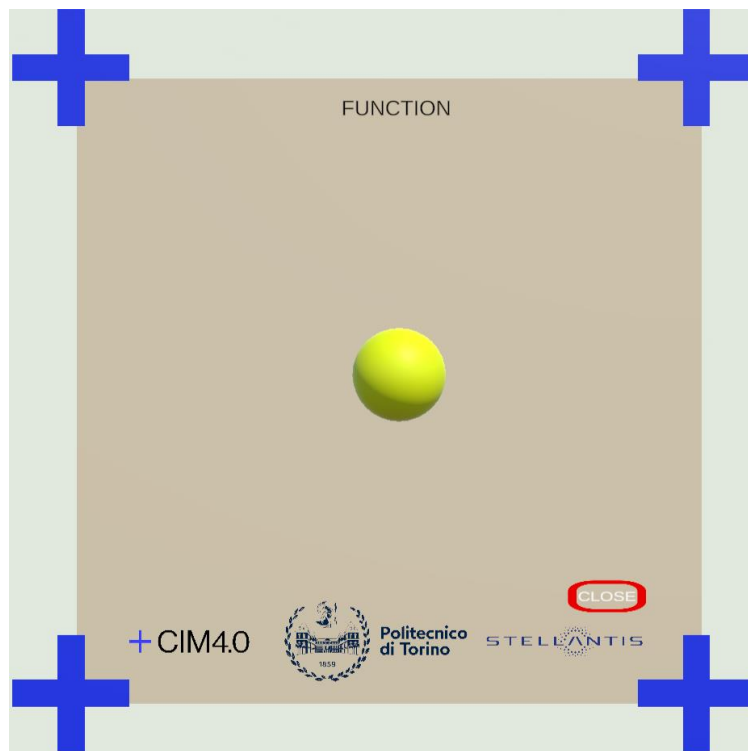


Figura 5.22: pannello contenente l'Highlighter con cui innescare il comportamento di determinate funzioni, visione dall'editor di Unity

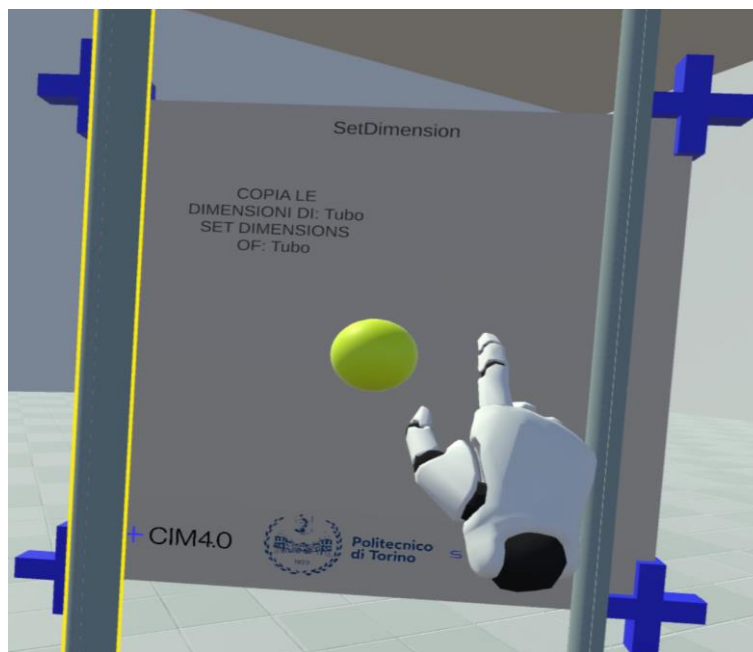


Figura 5.23: pannello contenente l'Highlighter con cui innescare il comportamento della funzione SetDimension, visione dall'applicativo in esecuzione

Nel caso del Copy dimension to/from e del Set X/Y/Z, l'Highlighter viene usato per copiare le informazioni relative a dimensione o posizione dell'oggetto con cui l'Highlighter viene fatto collidere sull'oggetto attivo. Per oggetto attivo si intende quello a cui fa riferimento il menu aperto.

5.1.4 Differenze nel Player

In base al fatto che all'interno del sistema si decida di interagire con gli HTC Vive Controller [67] o i MANUS Prime II [14] vengono utilizzati due Player differenti. Nel caso dei Vive Controller, il Player ha la struttura mostrata nella figura 5.24.

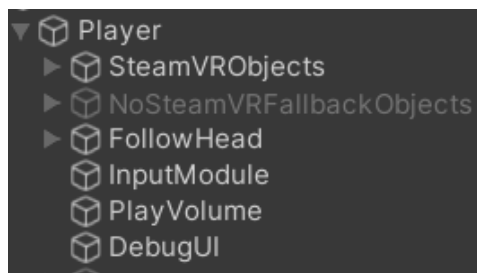


Figura 5.24: struttura del Player utilizzato per gli HTC Vive Controller

Gli elementi che gestiscono l'interazione con gli oggetti all'interno della scena sono contenuti all'interno del GameObject chiamato SteamVRObjects, il quale si presenta come mostrato nella figura 5.25.



Figura 5.25: struttura del GameObject SteamVRObjects

Tra questi sono da menzionare:

1. LeftHand: è il GameObject che gestisce il comportamento della mano e della sua interazione con gli oggetti della scena. Possiede i seguenti script:
 - a. Steam VR_Behaviour_Pose: gestisce le pose che il modello della mano virtuale deve assumere in base alle azioni che si stanno compiendo;
 - b. Hand Physics: gestisce le collisioni della mano;

- c. Hand: gestisce l'interazione vera e propria, riconosce quando il modello della mano si sovrappone ad un oggetto interagibile della scena, lo evidenzia e gestisce lo spostamento di tale oggetto nel momento in cui su di esso venga effettuata un'azione di Grab;
- 2. RightHand: discorso analogo a quello fatto per LeftHand con la differenza che in questo caso il modello mostrato è quello della mano destra mentre nel caso precedente era quello della mano sinistra;
- 3. VRCamera: è la camera che rappresenta il punto di vista dell'utente all'interno del mondo virtuale e si muove seguendo i movimenti della testa della persona che indossa l'HMD;
- 4. [SteamVR]: a questo GameObject è stato assegnato uno script, il quale permette di caricare, nel momento in cui si avvia l'applicazione, il set di azioni definito per il controller tramite l'interfaccia di SteamVR (figura 5.3).

Queste azioni sono mappate sui pulsanti del controller e, attraverso lo script LCAInputManager (di cui si parlerà in seguito) all'interno di Unity, vengono associate delle funzioni per definire i comportamenti da innescare nel momento in cui un determinato pulsante del controller viene premuto.

Nel caso dei MANUS Prime II [14] il Player è fatto come mostrato in figura 5.26.



Figura 5.26: struttura del Player utilizzato nel caso di impiego dei MANUS Prime II

Al suo interno si trovano:

- 1. una Camera che, in realtà, è stata aggiunta in quanto non contenuta nel Player originale presente nel plugin di MANUS [83] per Unity [16] e il cui scopo è quello di rappresentare il punto di vista dell'utente all'interno del mondo virtuale;
- 2. ManusHand_L e ManusHand_R che rappresentano i GameObject grazie ai quali si può gestire l'interazione con gli oggetti all'interno dell'ambiente virtuale. A questi GameObject è assegnato lo script Hand, il quale ha il compito di registrare le mani al Manus Communication Hub in modo da renderle disponibili per interagire con gli oggetti all'interno della scena in Unity. Nel momento in cui una mano viene inserita all'interno del Manus Communication Hub è possibile recuperarne le informazioni derivanti dai sensori presenti all'interno dei guanti e in base a questi dati il modello della mano all'interno di Unity [16] assumerà le pose corrispondenti.

I GameObject `ManusHand_L` e `ManusHand_R` hanno la seguente struttura (figura 5.27).



Figura 5.27: struttura del GameObject `ManusHand_L` (analoga a quella di `ManusHand_R`)

- a. L'`SK_Hand` è un GameObject a cui sono assegnati due script, `HandCollision` che si occupa di gestire le collisioni tra la mano e gli oggetti in scena mentre `HandAnimator` si occupa di animare la mano in modo tale che assuma una posa coerente con le informazioni provenienti dai sensori presenti all'interno del guanto e con la posa che l'utente sta assumendo nella realtà. Ha due elementi figli, ossia il modello della mano e lo scheletro che permette di animarlo;
- b. `Interaction`, invece, presenta tanti script quante sono le posizioni chiave che si vuole vengano riconosciute per poter far assumere al sistema determinati comportamenti. Nel caso specifico, gli script sono tre in quanto le pose della mano da riconoscere sono la `Grab Pose` quando si afferra un oggetto, la `Open Hand Pose` e la `Pointing Pose` (mostrate nel pannello corrispondente del tutorial iniziale, figura 5.10);
3. Tutto questo, per poter funzionare correttamente, ha anche bisogno di un altro elemento, il `Tracker Manager` a cui sono associati diversi script:
 - a. il `CommunicationHub`, che se non presente nella scena viene istanziato in automatico all'avvio dell'applicazione, il quale permette lo scambio di informazioni tra i sensori dei guanti e le `Hand` che ad esso sono state registrate tramite lo script `Hand`;
 - b. il `Tracker Manager` che inizializza il sistema di tracking che viene rilevato (se disponibile, di default assegna `SteamVR`);
 - c. il `TrackingSystemHermes` che gestisce l'aggiornamento dei tracker che vengono individuati e associati al sistema di tracciamento che è stato inizializzato.

In questo caso il sistema di tracciamento è quello di `SteamVR` [69] e i tracker sono quello relativo al caschetto indossato dall'utente e i due `Vive Tracker` [71] che vengono montati uno per guanto per poter ottenere le informazioni relative a posizione e rotazione dell'intera mano all'interno dello spazio.

5.1.5 Differenze di script per rendere gli oggetti interagibili

Che si utilizzi il sistema attraverso i Vive Controller [67] o i MANUS Prime II [14] ci sono degli script che devono essere assegnati agli oggetti affinché questi possano essere riconosciuti come interagibili.

Nel caso in cui si decida di utilizzare il sistema attraverso i Vive Controller [67] gli script che devono essere assegnati ad un oggetto interagibile sono i seguenti:

1. **Interactable**: è lo script base che rende un oggetto interagibile. Grazie ad esso, un GameObject può essere soggetto ad eventi quali il posizionamento del controller sull'oggetto stesso per far sì che esso possa essere legato al controller nel momento in cui si decide di compiere l'azione di Grab;
2. **Velocity Estimator**: come suggerisce il nome, questo script viene utilizzato per stimare la velocità (sia lineare che angolare) con cui viene spostato e/o ruotato l'oggetto, basandosi sul suo cambiamento di posizione. Le funzioni che sono in esso presenti vengono richiamate in un altro script che viene assegnato ad un oggetto interagibile che è il Throwable;
3. **Interactable Hover Events**: questo script permette di definire (nel caso in cui dovessero servire) delle funzioni personalizzate che si vuole vengano chiamate nel momento in cui il controller inizia (e smette) di posizionarsi sull'oggetto a cui questo script è associato ma anche nel momento in cui l'oggetto viene legato (e slegato) dal controller stesso;
4. **Throwable**: questo script viene utilizzato nel caso in cui un oggetto dovesse essere lanciato via. Sfruttando, infatti, il Rigidbody dell'oggetto lanciato (che è obbligatorio) viene settata la sua velocità lineare ed angolare andando a sfruttare le funzioni preposte definite nello script Velocity Estimator;
5. **Steam VR_Skeleton_Poser**: questo script permette di definire quale deve essere la posa del modello della mano simulata all'interno di Unity [16] nel momento in cui l'oggetto viene afferrato in modo tale che si possa rendere in maniera più realistica possibile l'azione di Grab. È possibile scegliere tra pose già definite all'interno del package di SteamVR [69] oppure definirne di personalizzate all'interno di Unity [16].

Nel caso in cui si decida di usare i MANUS Prime II [14] per interagire con gli oggetti nel mondo virtuale, invece, la situazione si semplifica in quanto gli script necessari affinché questo avvenga sono due:

1. **Grabbable Object**: questo script deve essere assegnato all'oggetto dall'editor di Unity. Possiede al suo interno dei metodi chiamati OnGrabbedStart(), OnGrabbedEnd() e OnGrabbedFixedUpdate() i quali vengono chiamati rispettivamente nel momento in cui si inizia ad afferrare un oggetto, quando lo si rilascia e mentre l'oggetto è afferrato per aggiornarne la posizione e la rotazione in maniera solidale con il modello 3D della mano. Possiede altri due metodi, OnAddedInteractingInfo() e OnRemovedInteractingInfo(), che vengono chiamati

quando un nuovo elemento inizia (o smette) di afferrare l'oggetto che possiede questo script (grazie alla funzione `OnAddedInteractingInfo`, ad esempio, è possibile gestire il Grab con entrambe le mani).

I metodi sopra citati sono metodi che vengono ereditati dall'interfaccia `IGrabbable`;

2. **Grabbed Object:** questo script viene assegnato all'oggetto in maniera automatica nel momento in cui questo inizia ad essere afferrato da una o due mani e viene eliminato nel momento in cui l'oggetto viene rilasciato.

Le funzioni dell'interfaccia `IGrabbable` che vengono definite all'interno dello script `GrabbableObject` vengono richiamate all'interno di `GrabbedObject`.

5.1.6 Gestione delle pose della mano con i MANUS Prime II

Utilizzando i MANUS Prime II [14] è possibile, attraverso apposito script, andare a definire dei parametri che permettono di valutare se si sta riproducendo, nella realtà, una determinata posa della mano (sfruttando i dati provenienti dai sensori posti all'interno dei guanti) in modo tale da poter definire dei comportamenti differenti.

Nel caso specifico si hanno tre script che valutano, rispettivamente, tre pose:

1. La **Grab Pose**: corrisponde a chiudere la mano a pugno e viene utilizzata per iniziare ad afferrare un oggetto interagibile (per rilasciarlo basta aprire la mano);
2. La **Open Hand Pose**: utilizzata per l'apertura del menu principale;
3. La **Pointing Pose**: utilizzata per puntare l'oggetto che si desidera istanziare nel mondo virtuale, a partire dal menu principale; viene, inoltre, utilizzata per puntare un oggetto interagibile per poter svolgere delle funzioni su di esso e per operare delle scelte previste dai menu presenti all'interno del sistema.

Si prenda il caso della posa di Grab (ma il discorso si può estendere a tutte e tre le pose definite sopra). La sua valutazione e gestione avviene attraverso i seguenti script:

1. il primo script, chiamato **Gesture Grab** eredita da una classe astratta chiamata **Gesture Base**, la quale al suo interno contiene il metodo `Evaluate()`.

Questo metodo viene implementato all'interno dello script **Gesture Grab** e il suo compito è quello di utilizzare i valori provenienti dai sensori posti all'interno dei guanti e, in base a dei valori float definiti all'interno dello script stesso (e, quindi, customizzabili in base alla posa che si vuole valutare), valuta per ogni dito se questo è completamente o parzialmente contratto o, viceversa, parzialmente o completamente disteso.

In base al numero di dita che risultano contratte o distese (nel caso del Grab è stato definito che debbano essere tutte e cinque almeno parzialmente contratte) viene settato un booleano a `true` se viene ritenuto che la posa assunta dall'utente sia un Grab e a `false` viceversa;

2. il secondo script è chiamato `HandGrabInteraction` e contiene al suo interno due funzioni principali, ovvero quella di `Grab` e quella di `Release`. Questo script, e gli analoghi per la posa della mano aperta e dell'indice puntato, sono assegnati al `GameObject` chiamato `Interaction` presente all'interno del `Player` di `Manus` mostrato nella figura 5.27.

All'interno dell'`Update()` viene chiamato il metodo `Evaluate()` definito nello script `GestureGrab`, ottenendo il valore del booleano che quello script restituisce. Se questo risulta essere `true` viene chiamato il metodo `Grab()`, mentre se risulta essere `false` viene chiamato il metodo `Release()`.

Il compito principale del metodo `Grab()` è quello di assegnare lo script `GrabbedObject` all'oggetto che si sta afferrando (recuperato dall'`OnTriggerEnter` definito in `HandGrabInteraction`) in modo tale che questo richiami le funzioni definite in `GrabbableObject` che permettono all'oggetto di seguire in maniera solidale il modello 3D della mano.

Viceversa, il metodo `Release()` elimina lo script `GrabbedObject` dall'oggetto che si stava afferrando in modo che questo venga rilasciato e smetta, quindi, di seguire la mano virtuale.

Oltre a chiamare i metodi `Grab()` e `Release()`, all'interno dell'`Update()` vengono anche settati due booleani, uno per la mano destra e uno per la mano sinistra, che tengono traccia sia del fatto che la mano stia riproducendo la posa di `Grab` (in quel caso il booleano è settato a `true`) o meno (in questo caso viene settato a `false`) ma anche del fatto che si tratti della mano destra o di quella sinistra.

Nel caso delle altre due pose (mano aperta e dito indice puntato) non sono presenti le funzioni di `Grab()` e `Release()` ma solo i due booleani di cui si è parlato sopra che permettono di tenere traccia del fatto che la mano stia assumendo una di quelle pose e quale mano sia;

3. l'ultimo script è il `GlobalPoseManager` il quale, al suo interno, recupera il riferimento allo script `HandGrabInteraction` (nel caso in cui si parli della posa di `Grab` mentre per le altre pose recupera lo script ad esse relativo) e tiene traccia, tramite due booleani, del fatto che una o entrambe le mani stiano assumendo la posa relativa. Questo viene fatto assegnando a questi due booleani il valore di quelli definiti all'interno di `HandGrabInteraction` (ed analoghi).

In questo modo questi due valori vengono messi a disposizione di tutti gli script definiti all'interno dell'applicativo e, tramite il loro valore, si possono andare a definire delle modalità secondo le quali il sistema deve comportarsi.

È possibile creare una nuova posa partendo dal template di quella di `Grab` e modificare direttamente dall'editor di `Unity` [16] i parametri in modo da personalizzarli. In particolare, i parametri messi a disposizione sono:

1. `allFingersMustBeAtLeastPartiallyBentForGesture` (booleano): se è settato a `true` implica che tutte le dita debbano essere almeno parzialmente contratte per far sì che la posa in questione venga riconosciuta, viceversa se è `false`;

2. `includeThumbInBendCount` (booleano): se è settato a `true` implica che anche il valore di contrazione del pollice debba essere tenuto in conto per determinare se una posa è assunta o meno, viceversa se è `false`;
3. `numberOfFullyBentFingersRequiredForGesture` (numero intero): permette di definire il numero di dita che devono essere completamente contratte affinché la posa in questione venga riconosciuta;
4. `valueAboveWhichFingerIsConsideredPartiallyBent` (valore float): indica il valore, tra 0 e 1, oltre il quale un dito viene considerato come parzialmente contratto;
5. `valueAboveWhichFingerIsConsideredFullyBent` (valore float): indica il valore, tra 0 e 1, al di sopra del quale un dito viene considerato completamente contratto.

5.1.7 Gestione menu

Anche la gestione dei menu utilizzati per istanziare elementi all'interno della scena e per eseguire delle azioni su di essi vengono gestiti in maniera differente a seconda del sistema di interazione che si decide di utilizzare.

Nel caso in cui si decida di utilizzare i Vive Controller [71] la gestione dei menu viene affidata allo script chiamato `LCAInputManager` che, in generale, raccoglie i diversi input provenienti dai controller tra cui, quindi, anche l'evento che ne consegue nel momento in cui si decide di premere il tasto Menu. All'interno dello script è definito un metodo, `Touch()`, che viene chiamato nel momento in cui il pulsante Menu del controller viene premuto.

A seconda dello stato attuale dell'applicazione, alla pressione del tasto Menu si susseguono diversi comportamenti:

1. se quando viene premuto non è aperto nessun menu e la mano non sta né puntando un oggetto interagibile né lo sta afferrando allora viene mostrato quello principale, ovvero quello tramite il quale è possibile istanziare oggetti all'interno del mondo virtuale. Per interagire con gli elementi di questo menu ma anche con tutti quelli contenuti negli altri, è sufficiente intersecare il modello 3D della mano con il riquadro che mostra l'oggetto che si vuole istanziare nel caso del menu principale mentre nel caso degli altri menu verrà intersecato con i pulsanti corrispondenti alla funzione che si vuole compiere sull'oggetto a cui il menu si riferisce;
2. se quando viene premuto il tasto Menu non ce n'è nessuno aperto ma si sta puntando un oggetto semplice viene mostrato il pannello con le funzioni che su di esso si possono compiere (Delete, Copy dimension from, Copy dimension to, Set X, Set Y e Set Z);
3. se quando viene premuto il tasto Menu non ce n'è nessuno aperto e si sta puntando un oggetto composto allora viene mostrato il pannello con le funzioni che su di esso si possono compiere (Delete, Split, Distance, Set X, Set Y e Set Z);

4. nel caso in cui venga premuto il tasto Menu sul controller e c'è già un pannello aperto che sia il Main Menu, il pannello specifico di un oggetto semplice, di uno composto o che si tratti di un pannello riferito ad una delle funzioni messe a disposizione per gli oggetti semplici e composti, questo viene chiuso.

Lo script `LCAInputManager` possiede anche un attributo di tipo stringa a cui si associa il nome del menu che in quel momento viene mostrato in modo da renderlo disponibile ad altri script qualora ce ne fosse bisogno.

Nel caso in cui si decida di utilizzare i guanti MANUS [14] per interagire all'interno dell'applicazione, il sistema di gestione dei menu cambia. Ci sono due script preposti a questo compito e sono:

1. `GlobalMenuManager`: con questo script se l'utente assume la posa della mano completamente aperta, che sia la destra o la sinistra, e non c'è nessun menu aperto viene mostrato quello principale; in questo caso, quindi, non avendo a disposizione dei pulsanti da premere per far comparire i pannelli vengono sfruttate le pose che le mani possono assumere.

Stesso discorso vale per l'interazione con gli elementi contenuti all'interno dei menu: se nel caso dei Vive Controller [71] era sufficiente far intersecare il modello 3D della mano con un elemento per poter istanziare un oggetto nel caso del menu principale o eseguire delle azioni sull'oggetto nel caso di altri, con i MANUS Prime II [14] è necessario intersecare una delle mani con l'elemento del menu desiderato ma questa deve anche trovarsi nella Pointing Pose.

Anche in questo caso è presente un parametro di tipo stringa al quale viene associato il nome del menu che in quel momento è aperto (stringa vuota se non ce n'è nessuno) in modo tale che sia a disposizione di altri script qualora alcuni comportamenti debbano essere attivati solo con un determinato pannello aperto;

2. `ActionOnObject`: questo script viene assegnato a tutti gli elementi interagibili. Regola l'apertura del pannello con le funzioni specifiche degli oggetti singoli o composti nel momento in cui il modello della mano interseca l'oggetto interagibile stesso mentre assume la Pointing Pose.

Stessa posa risulta necessaria per poter compiere sull'oggetto una delle azioni messe a disposizione dal menu andando ad intersecare la mano con l'elemento corrispondente all'azione che si vuole compiere.

Nel caso dei MANUS Gloves [14] ci sono delle differenze per quanto riguarda due funzioni che si possono compiere su un oggetto composto, ovvero la possibilità di staccarlo da una struttura complessa e quella di poterlo spostare rigidamente verso l'alto o verso il basso, anche in questo caso mentre è legato ad una struttura complessa. Queste due funzioni, infatti, possono essere attivate tramite gli appositi riquadri interagibili, rispettivamente "Split From Structure" e "Shift Up/Down", mostrati nel pannello degli oggetti composti (figure 5.28 e 5.29).

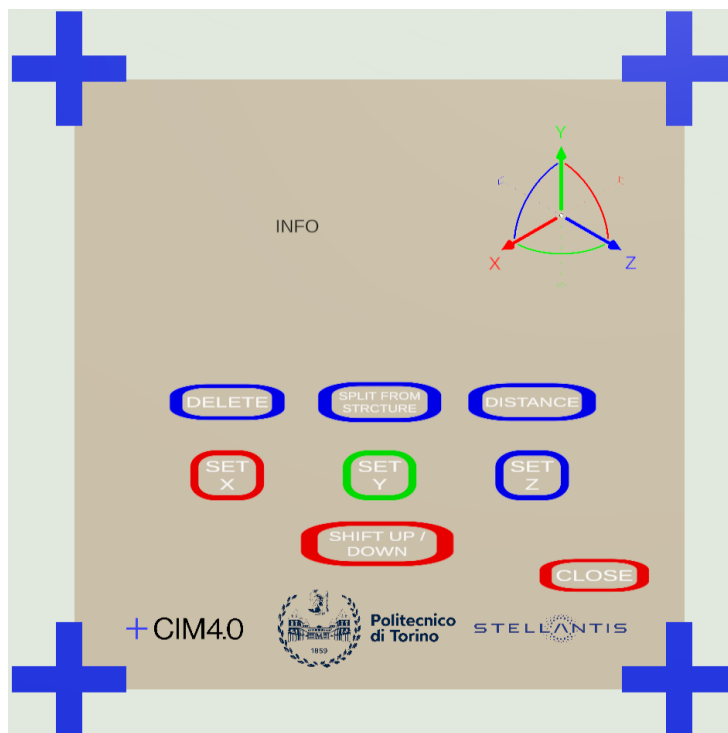


Figura 5.28: pannello per oggetti composti legati a strutture nel caso di utilizzo dei guanti MANUS, visione dall'editor di Unity. La versione con i Vive Controller è mostrata in fig. 5.16

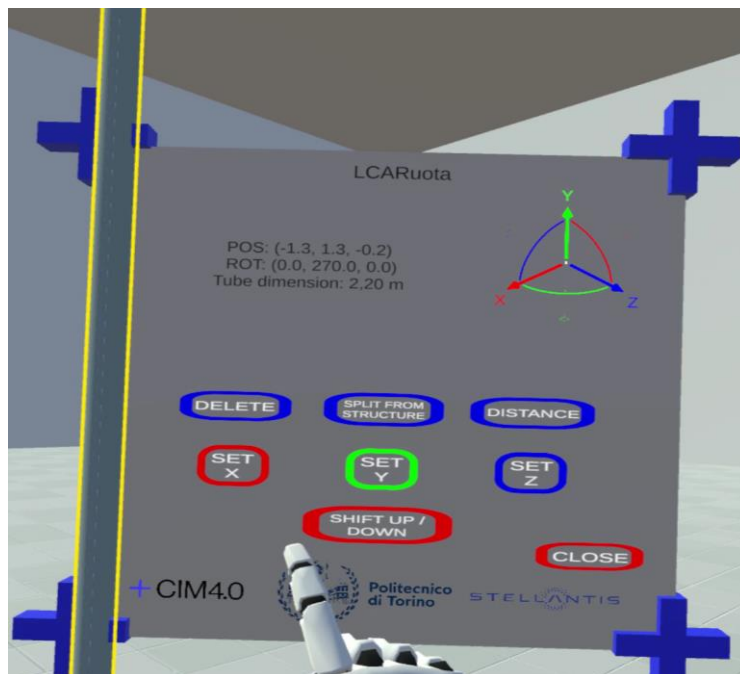


Figura 5.29: pannello di un LCARuota legato ad una struttura nel caso di utilizzo dei guanti MANUS, visione dall'applicativo in esecuzione

Nel caso dei Vive Controller [67], invece, queste funzioni vengono azionate nel momento in cui si preme il tasto Grip (se si vuole staccare l'oggetto composto dalla struttura complessa) o la combinazione dei pulsanti Touch + Trigger (nel caso in cui lo si voglia traslare rigidamente verso l'alto o verso il basso).

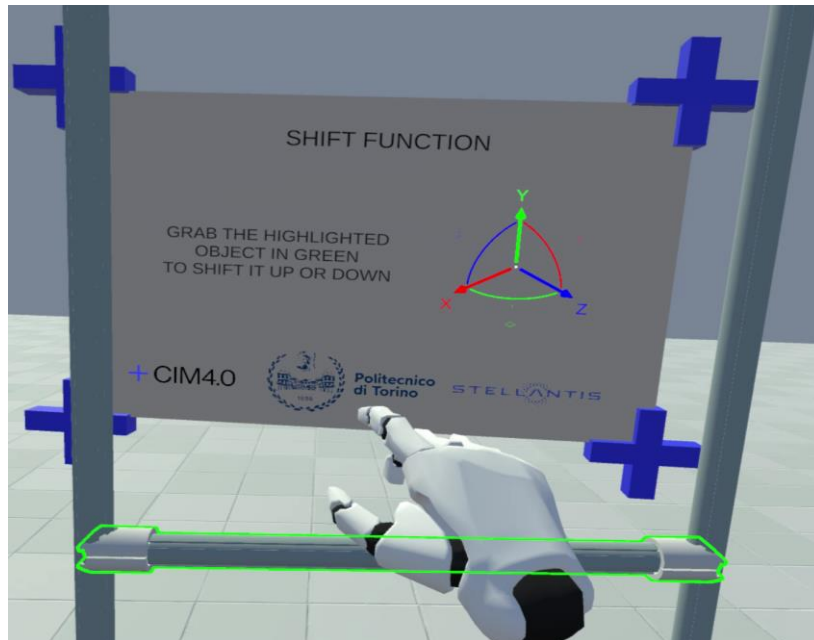


Figura 5.30: pannello mostrato quando si vuole spostare verticalmente un elemento della struttura complessa utilizzando i guanti MANUS

Un'altra differenza tra i due sistemi di interazione riguarda il come viene gestita la chiusura dei menu. Nel caso del Vive Controller [67], infatti, è sufficiente che ci sia un menu aperto e che venga premuto il tasto corrispondente sul controller per far sì che il pannello smetta di essere mostrato all'utente. Nel caso dei MANUS Gloves [14], invece, tutti i menu sono dotati di un pulsante Close che permette di chiuderli. Anche in questo caso, come per altre funzioni, per far sì che il menu venga chiuso è sufficiente far intersecare con il pulsante Close la mano mentre assume la Pointing Pose.

I pannelli specifici che vengono mostrati nel momento in cui si deve compiere su un oggetto, semplice o composto, una delle funzioni ad esso relative vengono, invece, chiusi in automatico nel momento in cui l'azione a loro associata è stata completata.

5.2 Progetto per il CAVE

Come accennato, lo stesso progetto sviluppato per l'HMD, basato solo sul sistema di interazione dei MANUS Prime II [14], è stato riadattato per funzionare all'interno di un CAVE. La logica del funzionamento del sistema non cambia rispetto al progetto descritto

in precedenza. Gli oggetti semplici sono gli stessi e possono unirsi per formare gli stessi oggetti composti con le medesime regole così come invariate sono le funzioni messe a disposizione per manipolarli. Anche il sistema di pose implementato per interagire con i vari elementi del mondo virtuale è rimasto invariato. La parte che cambia è legata al fatto che:

1. non si utilizza SteamVR [69] per integrare l'HMD, in quanto nel CAVE non viene usato come sistema di tracciamento;
2. risulta necessario adottare alcuni accorgimenti legati all'ambito del networking considerando che il sistema è composto dallo schermo centrale, il master, e dai due laterali e il pavimento, i client, che ricevono tutti i dati dal master.

Prima di passare al progetto in Unity [16], dato che all'interno del CAVE il tracciamento degli oggetti avviene tramite dei marker (passivi, in questo caso), è necessario andarli a configurare all'interno del software Motive [90].

Di base è sufficiente prendere un marker, posizionarlo all'interno dello spazio di cattura (il CAVE nel caso specifico) e sarà subito visibile all'interno del software. Dovendo tracciare sia posizione che rotazione delle mani, però, un solo marker non è sufficiente, ne sono necessari almeno tre per ciascuna mano.

Per questo è stata utilizzata la struttura mostrata in figura 5.31, una per mano, sulla quale sono stati posti tre marker con due configurazioni differenti in modo che il software riesca a distinguerle.



Figura 5.31: struttura su cui sono stati inseriti i marker per configurarli all'interno di Motive con il fine di tracciare la posizione dei guanti MANUS

A questo punto, all'interno di Motive [90], la struttura mostrata sopra sarà visibile come un triangolo dove i vertici sono i tre marker che si stanno utilizzando. L'ultimo passo prima di passare a Unity [16] è selezionare i marker in Motive e con questi creare un asset Rigidbody, assegnandogli un nome che sarà utilizzato nello script VRPN_Tracker in Unity [16], di cui si parlerà in seguito in questo capitolo. A questo punto è possibile passare al progetto all'interno di Unity [16].

Il template che permette di visualizzare l'ambiente virtuale sugli schermi del CAVE, visualizzabile in 3D tramite occhiali appositi, di impostare la comunicazione tra server e client e di inviare i dati di tracciamento da Motive [90] al progetto Unity [16] è stato sviluppato da Reply [84] ed è organizzato attorno al prefab chiamato Cave System (figura 5.32).



Figura 5.32: gerarchia degli elementi che costituiscono il progetto Unity per il CAVE

Prima di passare a descriverlo è opportuno soffermarsi, però, su due componenti messi a disposizione dalla libreria Mirror [93], i quali devono essere assegnati ad un elemento del mondo virtuale affinché non sia visibile solo al server ma abbiano consapevolezza della sua esistenza anche i client. I due componenti in questione sono i seguenti:

1. **NetworkIdentity:** è un componente che deve essere assegnato a qualunque oggetto si voglia gestire all'interno dell'ambiente in modo tale che sia visibile sia al server che ai client ad esso connessi. Alla versione attuale di Mirror, questo componente può essere assegnato ad un GameObject qualsiasi se si tratta di un oggetto singolo, mentre se un oggetto è figlio di un altro, il NetworkIdentity deve essere assegnato al padre;
2. **NetworkTransform:** è il componente con il compito di sincronizzare posizione, rotazione e scala di un oggetto, con un NetworkIdentity, tra il server e i client ad esso connessi. Nel momento in cui si vuole assegnare un NetworkTransform ad un GameObject, questo deve necessariamente avere un NetworkIdentity che, se non presente, verrà aggiunto automaticamente.

Sono presenti due versioni di questo component: una più affidabile che è, però, caratterizzata da una larghezza di banda limitata e può avere tempi di latenza elevati, e una meno affidabile che però garantisce una larghezza di banda alta e tempi di latenza molto bassi (è la versione consigliata).

La struttura del prefab Cave System è mostrata nella figura 5.33.

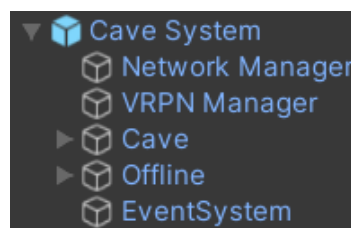


Figura 5.33: struttura del prefab Cave System

Il primo elemento che si incontra, il Network Manager, è anche quello che si occupa di instaurare e di gestire la connessione tra il server e i client. Presenta i seguenti script:

1. Cave Network Manager: si tratta di uno script che eredita dal Network Manager definito da Mirror [93]. Contiene al suo interno il metodo `OnServerAddPlayer()` tramite il quale viene istanziato il Player all'interno dell'ambiente virtuale. Presenta un attributo, `role`, che viene inizializzato a `Host` se la macchina su cui gira l'applicazione è stata impostata come server, viceversa viene impostato a `Client`. Ci sono i metodi, ereditati dal NetworkManager di Mirror, `StartHost()` e `StartClient()` per avviare server e client in base al valore con cui è stato inizializzato l'attributo `role`.

Presenta, inoltre, una lista nella quale è possibile inserire i prefab che l'utente può istanziare dinamicamente mentre utilizza l'applicazione;

2. Lite Net Lib 4 Mirror Transport e Lite Net Lib 4 Mirror Discovery: vengono usati in coppia e sono incaricati dell'effettivo scambio di dati tra il server e i client; nel momento in cui viene assegnato ad un GameObject lo script Lite Net Lib 4 Mirror Discovery, se la versione Transport non è presente nello stesso GameObject, viene associata in automatico.

Il primo script ha, tra i parametri, il numero massimo di connessioni accettate dal server, l'indirizzo IP dei client (in questo caso è un indirizzo IP locale) e la porta da utilizzare per lo scambio di messaggi. Come controparte, per permettere la comunicazione, anche il Lite Net Lib 4 Mirror Discovery ha il parametro che fa riferimento alla porta utilizzata per lo scambio dei messaggi che deve essere uguale al valore inserito nell'altro script.

Il GameObject successivo è il VRPN Manager al quale è assegnato lo script omonimo, `VRPN_Manager`. Questo script ha un parametro pubblico che rappresenta l'indirizzo di riferimento del server e permette anche in questo caso il corretto scambio di informazioni; come per il caso precedente, anche qui l'indirizzo è impostato a `localhost`.

Lo script presenta due metodi principali:

1. `GetPosition()`: accetta come parametri una stringa che è il nome del tracker di cui si vogliono ottenere le informazioni e il rispettivo canale, un numero intero. Questo metodo, in particolare, restituisce la posizione del tracker specificato chiamando il metodo `TrackerPos()` della classe `VRPN_NativeBridge`, la quale rappresenta l'interfaccia vera e propria con la libreria VRPN;
2. `GetRotation()`: accetta anch'esso come parametri il nome del tracker e il relativo canale ma, a differenza del caso precedente, restituisce informazioni riguardo la rotazione del tracker specificato. Anche in questo caso viene chiamato un metodo della classe `VRPN_NativeBridge`, `TrackerQuat()`, per andare a recuperare i dati a cui si è interessati.

Il GameObject tramite il quale viene riprodotta la struttura del CAVE all'interno del progetto Unity [16] è il successivo, chiamato Cave (struttura in figura 5.34) a cui sono stati assegnati i componenti NetworkIdentity e NetworkTransform.



Figura 5.34: struttura dell'oggetto Cave

Tra i diversi oggetti figli che possiede, quelli utili ai fini di questo progetto sono:

1. [Layout] PoliTo: rappresenta in scala 1:1 la struttura del CAVE; ha quattro oggetti figli, uno per ogni schermo del CAVE, a cui è associato uno script chiamato CaveWall. Questo viene richiamato in un altro script, OffAxisCamera, il quale è assegnato alle camere, di cui si parlerà al punto successivo, attraverso le quali è possibile visualizzare l'ambiente virtuale;
2. Head: rappresenta il punto di vista dell'utente. Possiede il NetworkIdentity e il NetworkTransform, in modo che sia visibile sia al server che ai client, e lo script tramite il quale viene tracciata la posizione e la rotazione della testa dell'utente, VRPN_Tracker.

Presenta due metodi principali, Position() e Rotation(). Il primo restituisce la posizione del tracker mentre il secondo ne restituisce la rotazione e le applica alla Transform dell'oggetto Head.

Importante è l'attributo pubblico Tracker presente all'interno di VRPN_Tracker. Si ricordi che nel software Motive [90] nel momento in cui viene creato un asset Rigidbody gli si associa un nome; questo deve essere lo stesso assegnato all'attributo Tracker affinché posizione e rotazione di Head siano effettivamente quelle del Rigidbody associato agli occhiali 3D indossati dall'utente.

Per recuperare i dati di posizione e rotazione restituiti da questi metodi, vengono richiamati i due presenti nel VRPN_Manager, GetPosition() e GetRotation().

L'oggetto Head ha due oggetti figli, Right Eye Origin e Left Eye Origin, che a loro volta ne hanno quattro ciascuno: si tratta di quattro camere orientate verso i quattro schermi del CAVE e tramite le quali si ha la visione dell'ambiente virtuale. Ciascuna di esse presenta uno script, OffAxisCamera, il quale permette di recuperare le dimensioni reali dei vari schermi verso cui una determinata camera è orientata e di modificare, in accordo con i dati provenienti dal tracking, il punto di vista dell'utente in modo che si abbia la sensazione di prospettiva all'interno del mondo virtuale;

3. Manus: contiene al suo interno il Player fornito dal plugin MANUS [14] per Unity [16] che è stato utilizzato anche per il progetto con l'HMD. Rispetto a quello utilizzato in quel caso, ci sono delle piccole differenze.

Come per altri oggetti all'interno dell'ambiente virtuale, anche in questo caso ai prefab che rappresentano le mani dell'utente sono stati assegnati i componenti `NetworkIdentity` e `NetworkTransform` mentre per tener traccia della loro posizione è stato assegnato lo script `VRPN_Tracker`, inserendo come nome del tracker quello definito nel software Motive [90].

Rispetto al progetto sviluppato per l'HMD ci sono delle differenze a livello di codice in modo tale che il sistema abbia delle informazioni coerenti tra il server e i client.

La prima differenza riguarda l'inserimento di un oggetto all'interno del mondo virtuale: in questo caso non avviene più semplicemente chiamando il metodo `Instantiate()` della classe `GameObject` ma è necessario ricorrere al metodo `Spawn()` della classe `NetworkServer`, il quale si occupa di rendere visibile l'oggetto sia al server ma anche ai client ad esso connessi. Tutti i prefab istanziabili in maniera dinamica devono avere il componente `NetworkIdentity` ed essere inseriti nell'apposita lista definita all'interno dello script `CaveNetworkManager`.

Un'altra differenza riguarda la chiamata ai vari metodi definiti all'interno degli script: per far sì che questi vengano chiamati anche dai client in modo che le azioni vengano eseguite anche su di essi è necessario, innanzitutto, che ogni script non derivi più dalla classe `MonoBehaviour` ma dalla `NetworkBehaviour` e prima di ogni metodo che si vuole chiamare anche sui client deve essere scritto il comando `[ClientRpc]`.

Per far sì che determinate variabili poi vengano sincronizzate all'interno del sistema, `Mirror` [93] offre la possibilità di definirle come `[SyncVar]` nel momento in cui vengono definite all'interno di uno script. È possibile, eventualmente, definire dei cosiddetti hook, ovvero delle funzioni da eseguire solo sui client nel momento in cui la variabile definita come `[SyncVar]` subisce delle modifiche.

Capitolo 6

Test e risultati

Per poter valutare gli effettivi vantaggi di un sistema di interazione sull'altro, quindi degli HTC Vive Controller [67] sui MANUS Prime II [14] o viceversa, sono stati eseguiti dei test. Questi consistono in due parti principali:

1. Una prima in cui si fa provare l'applicativo sia sfruttando i controller che i guanti MANUS [14] a ciascun utente;
2. Una seconda parte che prevede, al termine dell'esperienza di ciascun utente con ciascun sistema di interazione, la compilazione di un questionario.

La prima parte, inoltre, si divide in due sotto parti:

1. la prima in cui all'utente vengono mostrati dei pannelli facenti parte del tutorial all'interno dei quali viene descritto il sistema, il suo funzionamento e i comandi per poter compiere delle azioni al suo interno;
2. nella seconda viene assegnato all'utente un task che consiste nel riprodurre la struttura di un LCA di esempio (mostrato all'interno dell'ambiente virtuale).

Ad ogni utente sono state fatte provare entrambe le versioni del sistema, utilizzando entrambi i sistemi di interazione. Per ognuna delle due variazioni del progetto, durante l'esecuzione dei test è stato seguito l'ordine descritto sopra.

Il tutorial, in entrambe le versioni sottoposte all'utente, consiste nei seguenti step:

1. viene spiegato all'utente che potrà interagire con degli oggetti all'interno dell'ambiente virtuale attraverso i pulsanti dei controller in un caso e attraverso le pose delle mani nell'altro;
2. viene mostrato come aprire il menu principale, istanziare un oggetto e afferrarlo;
3. viene presentata la possibilità di scalare alcuni oggetti e come farlo;
4. viene spiegato come aprire il menu contenente le funzioni applicabili ad un oggetto singolo e si invita a provarle;
5. viene, poi, presentata la possibilità di creare degli oggetti composti partendo da quelli singoli;
6. viene spiegato che alcuni oggetti composti possono essere uniti per formare una struttura complessa;
7. viene illustrato come spostare l'intera struttura complessa;
8. viene mostrato come spostare verso l'alto o il basso un solo elemento della struttura complessa lasciando invariati gli altri che la compongono;
9. viene spiegato, infine, che è possibile staccare un oggetto composto dalla struttura complessa.

Il secondo step consiste nel far riprodurre all'utente la struttura di un LCA visualizzabile all'interno del mondo virtuale (figura 6.1) tramite apposito pulsante del menu principale.

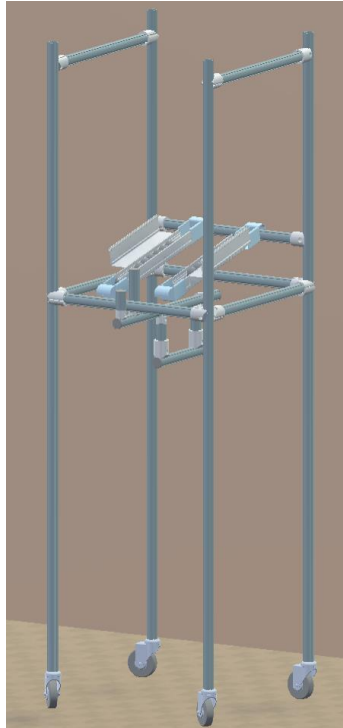


Figura 6.1: LCA che l'utente deve riprodurre in fase di testing

L'utente avrà sempre a disposizione un pannello contenente tutte le funzioni applicabili su un oggetto e relativa spiegazione e, nel caso di utilizzo dei controller, anche un pannello con una loro rappresentazione contenente il nome dei relativi pulsanti; questi avranno uno sfondo che si colorerà di giallo nel momento in cui vengono premuti dall'utente in modo da fornire un feedback immediato.

Durante questa fase, per entrambe le versioni del sistema, si è tenuta traccia degli errori commessi dagli utenti, delle domande poste per poter procedere nel task assegnato e del tempo totale che è stato impiegato per portarlo a termine.

L'ultimo step della fase di testing consiste nella compilazione di un questionario in lingua italiana tramite il quale poter valutare l'usabilità e il carico di lavoro percepito in entrambi i sistemi in modo da poterli confrontare. Il questionario si suddivide nelle seguenti sezioni:

1. una prima parte generale in cui si chiede genere, età dell'utente e se ha mai avuto esperienze in realtà virtuale;
2. una seconda parte in cui viene valutata l'usabilità del sistema rispondendo alle domande del questionario SUS [94];

3. una terza parte in cui viene valutato il carico di lavoro percepito dagli utenti durante la realizzazione dell'LCA di esempio attraverso il questionario NASA Task Load Index (NASA-TLX) [95];
4. un'ultima parte contenente domande più generiche per fornire una valutazione soggettiva del sistema e dei suggerimenti per un suo miglioramento.

6.1 Questionario SUS (System Usability Scale)

Si tratta del questionario che viene più frequentemente utilizzato per la valutazione dell'usabilità di un sistema. Creato nel 1986 da John Brooke, viene scelto nella maggior parte dei casi in quanto presenta un template già predefinito senza il bisogno di pensare a ulteriori domande, può essere somministrato facilmente e ha un'alta attendibilità.

È composto da dieci affermazioni alle quali è possibile rispondere assegnando un valore su una scala tra 1 e 5 punti i cui valori sono così mappati:

1. Fortemente in disaccordo: 1 punto;
2. In disaccordo: 2 punti;
3. Neutro: 3 punti;
4. D'accordo: 4 punti;
5. Fortemente d'accordo: 5 punti.

I valori assegnati a ciascuna domanda contribuiscono al calcolo di un punteggio finale, compreso tra 0 e 100, tramite cui viene valutata l'usabilità generale del sistema. Il punteggio finale si ottiene nel seguente modo:

1. Sottrarre 5 ai punteggi delle domande dispari e sommare i valori ottenuti;
2. Sottrarre a 25 la somma dei punteggi delle domande pari;
3. Per ottenere il punteggio finale, sommare i valori ottenuti dai punti precedenti e moltiplicare il totale per 2.5.

Basandosi su questo punteggio, il sistema viene valutato seguendo la suddivisione in fasce riportata nella tabella 6.1.

Punteggio SUS	Valutazione sistema
> 80.3	Eccellente
68 – 80.3	Buono
68	Accettabile
51 - 68	Scarso
< 51	Pessimo

Tabella 6.1: Suddivisione dei punteggi test SUS

Il valore minimo per considerare un sistema accettabile dal punto di vista dell'usabilità è 68: punteggi inferiori mostrano un sistema che presenta problemi nella sua progettazione che deve essere ripensata in più parti, mentre valori superiori a 68 fanno riferimento ad un sistema che ha bisogno di piccoli aggiustamenti senza, però, criticità dal punto di vista progettuale.

6.2 Questionario NASA – TLX

Il NASA Task Load Index (chiamato anche NASA – TLX) [95] è un test, sviluppato presso l'Ames Research Center della NASA, utilizzato per valutare il carico di lavoro percepito dagli utenti durante l'utilizzo di un sistema. È diviso in due parti. Nella prima parte sono presenti sei affermazioni inerenti:

1. Richiesta mentale: attività mentale e percettiva richieste per lo svolgimento del task proposto;
2. Domanda fisica: attività fisica richiesta durante lo svolgimento del task;
3. Domanda temporale: riferita a quanta pressione in termini di tempo l'utente ha sentito durante lo svolgimento del compito proposto;
4. Prestazioni complessive: riferito a quanto l'utente ha avuto successo e si sente soddisfatto della prestazione ottenuta;
5. Sforzo: quanto l'utente ha dovuto sforzarsi, sia mentalmente che fisicamente, per raggiungere la prestazione ottenuta;
6. Livello di frustrazione: quanto l'utente si è sentito frustrato, stressato durante lo svolgimento il compito che gli è stato assegnato.

Per ognuna di queste domande, l'utente può assegnare un punteggio da 0 a 100 con incrementi di 5 punti (nel caso di questo studio la scala è stata rimappata con punteggi compresi tra 1 e 10).

La seconda parte del test, invece, presenta le stesse sei affermazioni precedentemente elencate ma in coppia, per un totale di 15 coppie (in modo da avere tutte le possibili combinazioni). Per ognuna di esse, l'utente deve indicare quale dei due aspetti ha influito maggiormente sul carico di lavoro: se, ad esempio, la coppia è formata da domanda temporale e fisica, deve scegliere se nello svolgimento del task ha avuto un peso maggiore la pressione in termini di tempo o di attività fisica. Questa seconda parte, combinata con la prima, permette di ottenere una media ponderata per ciascuna delle categorie.

6.3 Risultati ottenuti

I test sono stati condotti presso il CIM 4.0 [7] su un campione di 12 utenti di cui 3 donne e 9 uomini, di età compresa tra i 19 e i 25 anni, sia con esperienze pregresse nel campo della VR che non.

Nella tabella 6.2 sono riportati i tempi impiegati dall'utente per completare il processo di prototipazione dell'LCA proposto.

Utente	HTC Vive Controller (durata in minuti)	MANUS Prime II (durata in minuti)
1 (V - M)	22:58	12:52
2 (M - V)	13:48	12:34
3 (V - M)	08:42	06:40
4 (M - V)	08:50	06:00
5 (V - M)	14:26	12:16
6 (M - V)	06:40	11:42
7 (V - M)	27:54	14:43
8 (M - V)	11:47	12:23
9 (V - M)	08:06	09:05
10 (M - V)	13:51	10:01
11 (V - M)	06:06	05:07
12 (M - V)	10:09	09:31
MEDIA	12:46	10:15

Tabella 6.2: Tempi di prototipazione dell'LCA per entrambi i sistemi di interazione

Insieme al numero dell'utente sono presenti, tra parentesi, le lettere V e M per identificare, rispettivamente, il sistema basato sui Vive Controller [67] e sui MANUS Prime II [14]; l'ordine con cui sono scritti indica quale dei due sistemi è stato fatto provare per primo e quale per secondo. Quest'ordine viene invertito tra un utente e l'altro per evitare che il secondo sistema provato risulti sempre migliore del primo in quanto se ne sono già apprese le dinamiche di funzionamento. Come si vede, anche se di poco, il sistema basato sui MANUS Prime II [14] ha richiesto in media meno tempo per il completamento del task rispetto alla versione basata sui controller.

Dal questionario SUS [63] è emerso che entrambi i sistemi oggetto di test hanno ottenuto un punteggio superiore alla soglia minima per considerarli accettabili e non soggetti ad errori profondi di design. Tuttavia, il sistema basato sui Vive Controller [67] si colloca nel limite inferiore di tale fascia avendo ottenuto un punteggio pari a 69.17, poco al di sopra del valore soglia, mentre la versione che implementa i MANUS Prime II

[14] si colloca nella parte alta di tale fascia, avendo ottenuto un punteggio pari a 78.54 e non distante dal valore di 80.3, soglia al di sopra della quale l'usabilità del sistema viene considerata eccellente.

Le tabelle 6.3 e 6.4 mostrano, rispettivamente, i risultati del questionario SUS per il sistema basato sugli HTC Vive Controller [67] e sui MANUS Prime II [14].

UTENTE	PARI	DISPARI	TOTALE
U1	16	18	85
U2	8	6	35
U3	16	14	75
U4	16	7	57,5
U5	14	13	67,5
U6	16	10	65
U7	15	13	70
U8	14	10	60
U9	16	19	87,5
U10	18	14	80
U11	19	17	90
U12	13	10	57,5
MEDIA	15,08	12,58	69,17

Tabella 6.3: Risultati SUS del sistema con gli HTC Vive Controller

UTENTE	PARI	DISPARI	TOTALE
U1	15	16	77,5
U2	16	14	75
U3	18	14	80
U4	19	14	82,5
U5	12	12	60
U6	18	14	80
U7	18	18	90
U8	15	12	67,5
U9	15	20	87,5
U10	20	12	80
U11	16	19	87,5
U12	16	14	75
MEDIA	16,50	14,92	78,54

Tabella 6.4: Risultati SUS del sistema con i MANUS Prime II

Di seguito, invece, sono riportati i risultati nel dettaglio per le singole affermazioni contenute all'interno del SUS [63] per entrambi i sistemi provati (figure 6.2 e 6.3).

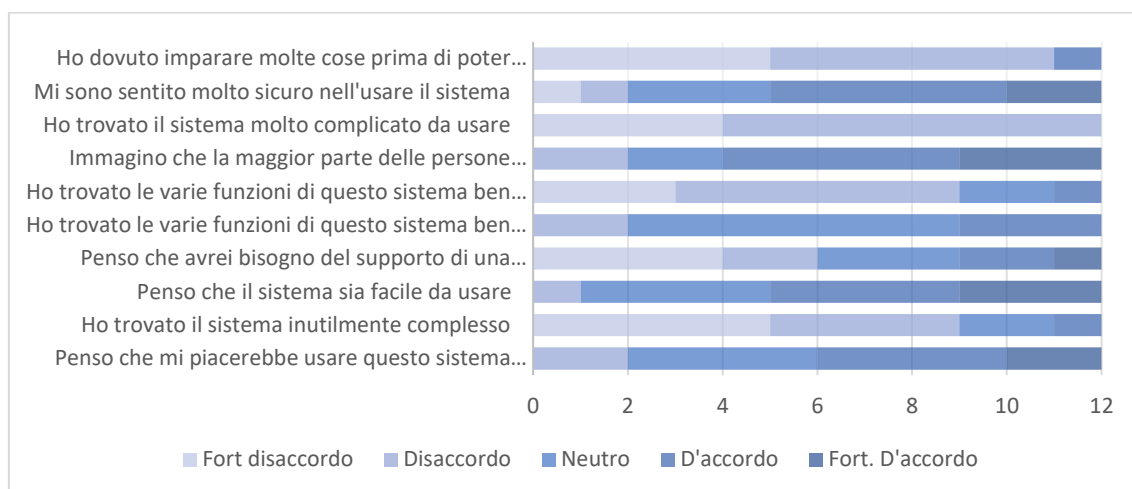


Figura 6.2: rappresentazione delle risposte al SUS per il sistema basato sugli HTC Vive Controller

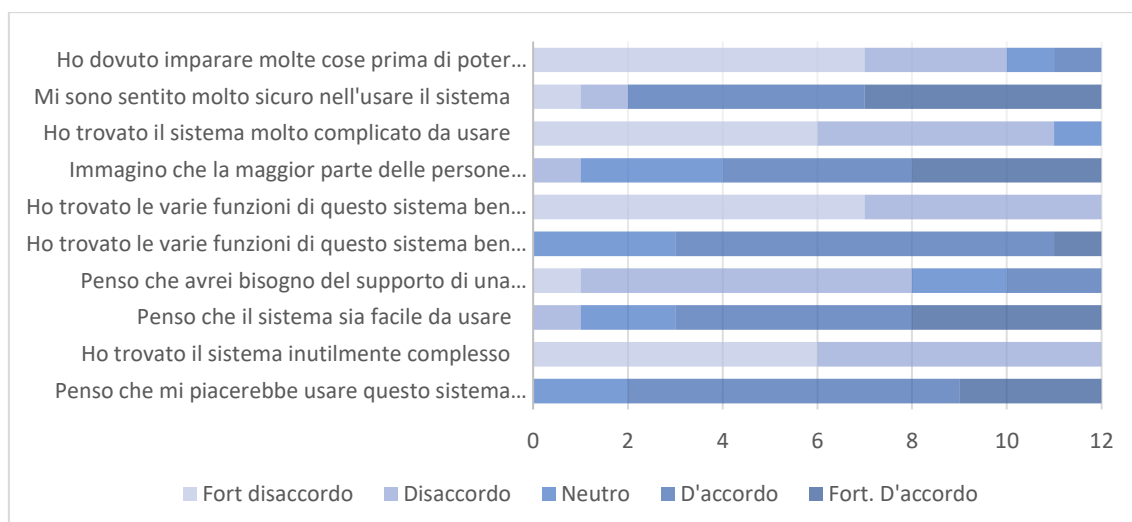


Figura 6.3: rappresentazione delle risposte al SUS per il sistema basato sui MANUS Prime II

Di seguito, invece, sono riportati i risultati derivanti dal NASA – TLX [95]: il grafico (in figura 6.4) presenta sull'asse delle ascisse le sei affermazioni che compongono il test mentre sull'asse delle ordinate la media dei valori che ogni utente ha assegnato ad ogni affermazione. Il grafico è strutturato in modo da confrontare, per ogni affermazione, le medie ottenute dai test effettuati sul sistema basato sui Vive Controller [67] e sui MANUS Prime II [14].

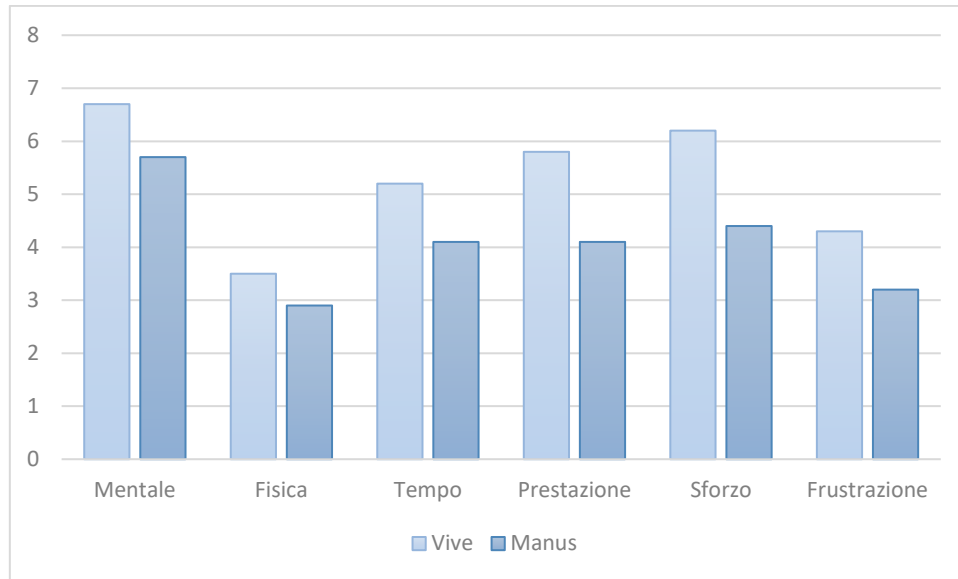


Figura 6.4: confronto risultati NASA-TLX per entrambi i sistemi testati

Quello che si può subito notare è che, in generale, il sistema basato sui guanti MANUS [14] è risultato essere migliore rispetto a quello basato sui controller in tutte le categorie prese in considerazione dal NASA-TLX [95]. Come si può vedere, infatti, sia a livello di attività mentale che fisica la media ottenuta è inferiore rispetto a quella ricavata dall'altra versione del sistema così come la pressione temporale percepita, la frustrazione e lo sforzo necessario a completare il task proposto. Anche la prestazione media risulta essere migliore nonostante la media ottenuta sia inferiore rispetto a quella riguardante il sistema con i controller: questo risultato, in realtà, è da interpretare in senso contrario poiché, per come è strutturato il test, valori bassi per quanto riguarda la prestazione significano maggiore soddisfazione nella performance.

Per quanto riguarda, invece, le categorie che hanno ottenuto la media maggiore e quella minore c'è concordanza tra i due sistemi testati: in entrambi i casi il fattore che è stato percepito di più è stato quello inerente l'attività mentale richiesta per il completamento del task mentre quello che è stato percepito meno riguarda l'attività fisica necessaria a riprodurre la struttura dell'LCA proposto.

Capitolo 7

Conclusioni e sviluppi futuri

Con il presente studio si sono voluti confrontare due sistemi di interazione applicabili ad esperienze in Realtà Virtuale, gli HTC Vive Controller [67] in un caso e i MANUS Prime II [14] nell'altro.

Il caso studio è quello della prototipazione di Low Cost Automation (LCA) in VR, aspetto che sta interessando un numero sempre maggiore di industrie per i vantaggi che questo porta sia per quanto riguarda un risparmio in termini di tempo sia per quello in termini di risorse economiche che risultano essere in quantità minore di quelle che sarebbero necessarie per realizzare i prototipi fisicamente ogni qualvolta ci si accorge che degli aspetti necessitano delle modifiche e/o ottimizzazioni.

Il lavoro fatto è partito, quindi, da una fase di ricerca che ha riguardato da un lato gli attuali utilizzi della VR nel campo dell'Industria 4.0 e, in particolare, nell'ambito della prototipazione (spaziando dal mondo del design di interni, come per l'applicazione IKEA VR Experience [40], a quello dell'automobile presentando il caso particolare di BMW [43] e dell'utilizzo che fa della Realtà Virtuale), dall'altro sono stati studiati dei sistemi che già implementano l'interazione attraverso il movimento e i gesti delle mani per capirne gli aspetti già in utilizzo e i limiti attualmente non risolti.

Lo scopo di questo lavoro è stato quello di presentare e far provare all'utente un applicativo che integra al suo interno i due diversi modi di interagire in un mondo virtuale e, attraverso un questionario, andare a capire quale dei due risulta essere il migliore in termini di usabilità e di senso di immersione all'interno della simulazione.

Prendendo l'applicativo per HMD che implementa gli HTC Vive Controller [67], quindi, è iniziato il processo di integrazione dei MANUS Prime II [14] al suo interno, andandone a modificare struttura e script in modo tale che questi si adattassero alla nuova modalità di interazione prevista.

È stata sviluppata, inoltre, sempre attorno al caso d'uso della prototipazione di un LCA, una prima versione del sistema da utilizzare all'interno di un CAVE, un'altra soluzione hardware appartenente al mondo della Realtà Virtuale tramite la quale l'utente si trova all'interno di un mondo digitale pur avendo ancora coscienza di quello reale non essendo completamente isolato da questo. In particolare, sono state riportate tutte le funzioni principali in modo tale da permettere all'utente di portare a termine l'assemblaggio dell'LCA proposto.

Gli utenti oggetto di test hanno avuto la possibilità di testare il sistema attraverso entrambe le modalità di interazione e dalle risposte raccolte tramite i questionari è emerso che sia la versione con i controller che quella con i guanti hanno raggiunto il livello di sufficienza previsto dal test SUS [63] somministrato. Tuttavia, nel caso dell'utilizzo dei MANUS Prime II [14], il punteggio è stato non solo sufficiente ma anche molto vicino alla soglia a partire dalla quale un sistema viene valutato come eccellente in termini di usabilità (mentre il punteggio assegnato alla versione che implementa i Vive Controller [67] è stato di pochissimo superiore alla soglia della sufficienza).

Per come è strutturato, il sistema offre la possibilità di inserire all'interno del mondo virtuale altri oggetti di base in maniera abbastanza immediata, così come permette di definire altre pose riproducibili dall'utente per poterci interagire. È sufficiente, infatti, partire dal template degli script che riconoscono le pose attualmente implementate e andare a modificare i valori presenti al loro interno per poterne riconoscere altre.

Lo step successivo dell'applicativo utilizzabile all'interno del CAVE riguarda un suo miglioramento in generale in modo che le informazioni siano sempre coerenti tra il server e i client ad esso connessi; questo permetterà di andare a testarlo insieme alla versione per l'HMD e individuare quale delle due viene ritenuta più usabile e immersiva da parte degli utenti.

Appendice

Questionario sottoposto all'utente

LCA Workplace Design

Utente *

La tua risposta

Età *

La tua risposta

Genere *

☐ Uomo
☐ Donna
☐ Preferisco non specificarlo

Hai mai vissuto un'esperienza di Realtà Virtuale utilizzando un visore? *

☐ Sì
☐ No

Parte 1: Il questionario proposto permette di fornire una misurazione della valutazione dell'usabilità complessiva (SUS)

Valutazione dell'accuratezza della risposta del sistema [1: Fortemente in disaccordo - 5: Fortemente in accordo] *

	1	2	3	4	5
Penso che mi piacerebbe usare questo sistema frequentemente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ho trovato il sistema inutilmente complesso	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Penso che il sistema sia facile da usare	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Penso che avrei bisogno del supporto di una persona tecnica per poter utilizzare questo sistema	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ho trovato le varie funzioni di questo sistema ben integrate	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ho pensato che ci fosse troppa incoerenza in questo sistema	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Immagino che la maggior parte delle persone imparerebbe ad usare questo sistema molto rapidamente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ho trovato il sistema molto complicato da usare	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Mi sono sentito molto sicuro nell'usare il sistema	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ho dovuto imparare molte cose prima di poter utilizzare questo sistema	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Parte 2a: Il questionario proposto permette di misurare il carico di lavoro percepito dalle azioni proposte (NASA)

Quanta attività mentale e percettiva è stata richiesta (es. pensare, decidere, riflettere, ricordare, osservare, ricercare ecc)? Il task era facile o difficile, semplice o complesso, preciso o tollerante? *

1 2 3 4 5 6 7 8 9 10

Poca ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Molta

Quanta attività fisica è stata richiesta (es. premere, spingere, ruotare, controllare, attivare ecc)? Il task è stato facile o impegnativo, lento o rapido, lasco o faticoso, rilassante o laborioso? *

1 2 3 4 5 6 7 8 9 10

Poca ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Molta

Quanta pressione temporale hai sentito a causa della frequenza con cui si susseguivano i task o gli elementi dei task? Il ritmo è stato lento e rilassato o rapido e frenetico? *

1 2 3 4 5 6 7 8 9 10

Poca ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Molta

Quanto pensi di aver avuto successo nel raggiungimento degli obiettivi dei task preparati dallo sperimentatore (o da te stesso)? Quanto sei soddisfatto della tua performance nel raggiungimento degli obiettivi? *

1 2 3 4 5 6 7 8 9 10

Molto ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Poco

Quanto duro hai dovuto lavorare (mentalmente e fisicamente) per raggiungere il tuo livello di performance? *

1 2 3 4 5 6 7 8 9 10

Poco ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Molto

Quanto ti sei sentito insicuro, scoraggiato, irritato, stressato e infastidito rispetto a sicuro, gratificato, soddisfatto, rilassato e compiaciuto durante il task? *

1 2 3 4 5 6 7 8 9 10

Poco ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ Molto

Parte 2b - Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro percepito per il task (NASA)

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro percepito per il task *

- ☐ Tempo - Quanta pressione temporale hai sentito a causa della frequenza con cui si susseguivano i task o gli elementi dei task? Il ritmo è stato lento e rilassato o rapido e frenetico?
- ☐ Frustrazione - Quanto ti sei sentito insicuro, scoraggiato, irritato, stressato e infastidito rispetto a sicuro, gratificato, soddisfatto, rilassato e compiaciuto durante il task?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro percepito per il task *

- ☐ Frustrazione - Quanto ti sei sentito insicuro, scoraggiato, irritato, stressato e infastidito rispetto a sicuro, gratificato, soddisfatto, rilassato e compiaciuto durante il task?
- ☐ Sforzo - Quanto duro hai dovuto lavorare (mentalmente e fisicamente) per raggiungere il tuo livello di performance?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Sforzo - Quanto duro hai dovuto lavorare (mentalmente e fisicamente) per raggiungere il tuo livello di performance?
- ☐ Performance - Quanto pensi di aver avuto successo nel raggiungimento degli obiettivi dei task preparati dallo sperimentatore (o da te stesso)? Quanto sei soddisfatto della tua performance nel raggiungimento degli obiettivi?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Richiesta fisica - Quanta attività fisica è stata richiesta (es. premere, spingere, ruotare, controllare, attivare ecc)? Il task è stato facile o impegnativo, lento o rapido, lasco o faticoso, rilassante o laborioso?
- ☐ Frustrazione - Quanto ti sei sentito insicuro, scoraggiato, irritato, stressato e infastidito rispetto a sicuro, gratificato, soddisfatto, rilassato e compiaciuto durante il task?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Performance - Quanto pensi di aver avuto successo nel raggiungimento degli obiettivi dei task preparati dallo sperimentatore (o da te stesso)? Quanto sei soddisfatto della tua performance nel raggiungimento degli obiettivi?
- ☐ Richiesta mentale - Quanta attività mentale e percettiva è stata richiesta (es. pensare, decidere, riflettere, ricordare, osservare, ricercare ecc)? Il task era facile o difficile, semplice o complesso, preciso o tollerante?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Tempo - Quanta pressione temporale hai sentito a causa della frequenza con cui si susseguivano i task o gli elementi dei task? Il ritmo è stato lento e rilassato o rapido e frenetico?
- ☐ Sforzo - Quanto duro hai dovuto lavorare (mentalmente e fisicamente) per raggiungere il tuo livello di performance?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Richiesta fisica - Quanta attività fisica è stata richiesta (es. premere, spingere, ruotare, controllare, attivare ecc)? Il task è stato facile o impegnativo, lento o rapido, lasco o faticoso, rilassante o laborioso?
- ☐ Performance - Quanto pensi di aver avuto successo nel raggiungimento degli obiettivi dei task preparati dallo sperimentatore (o da te stesso)? Quanto sei soddisfatto della tua performance nel raggiungimento degli obiettivi?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Richiesta mentale - Quanta attività mentale e percettiva è stata richiesta (es. pensare, decidere, riflettere, ricordare, osservare, ricercare ecc)? Il task era facile o difficile, semplice o complesso, preciso o tollerante?
- ☐ Sforzo - Quanto duro hai dovuto lavorare (mentalmente e fisicamente) per raggiungere il tuo livello di performance?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Richiesta fisica - Quanta attività fisica è stata richiesta (es. premere, spingere, ruotare, controllare, attivare ecc)? Il task è stato facile o impegnativo, lento o rapido, lasco o faticoso, rilassante o laborioso?
- ☐ Tempo - Quanta pressione temporale hai sentito a causa della frequenza con cui si susseguivano i task o gli elementi dei task? Il ritmo è stato lento e rilassato o rapido e frenetico?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Frustrazione - Quanto ti sei sentito insicuro, scoraggiato, irritato, stressato e infastidito rispetto a sicuro, gratificato, soddisfatto, rilassato e compiaciuto durante il task?
- ☐ Richiesta mentale - Quanta attività mentale e percettiva è stata richiesta (es. pensare, decidere, riflettere, ricordare, osservare, ricercare ecc)? Il task era facile o difficile, semplice o complesso, preciso o tollerante?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Tempo - Quanta pressione temporale hai sentito a causa della frequenza con cui si susseguivano i task o gli elementi dei task? Il ritmo è stato lento e rilassato o rapido e frenetico?
- ☐ Richiesta mentale - Quanta attività mentale e percettiva è stata richiesta (es. pensare, decidere, riflettere, ricordare, osservare, ricercare ecc)? Il task era facile o difficile, semplice o complesso, preciso o tollerante?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Performance - Quanto pensi di aver avuto successo nel raggiungimento degli obiettivi dei task preparati dallo sperimentatore (o da te stesso)? Quanto sei soddisfatto della tua performance nel raggiungimento degli obiettivi?
- ☐ Frustrazione - Quanto ti sei sentito insicuro, scoraggiato, irritato, stressato e infastidito rispetto a sicuro, gratificato, soddisfatto, rilassato e compiaciuto durante il task?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Performance - Quanto pensi di aver avuto successo nel raggiungimento degli obiettivi dei task preparati dallo sperimentatore (o da te stesso)? Quanto sei soddisfatto della tua performance nel raggiungimento degli obiettivi?
- ☐ Tempo - Quanta pressione temporale hai sentito a causa della frequenza con cui si susseguivano i task o gli elementi dei task? Il ritmo è stato lento e rilassato o rapido e frenetico?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Richiesta mentale - Quanta attività mentale e percettiva è stata richiesta (es. pensare, decidere, riflettere, ricordare, osservare, ricercare ecc)? Il task era facile o difficile, semplice o complesso, preciso o tollerante?
- ☐ Richiesta fisica - Quanta attività fisica è stata richiesta (es. premere, spingere, ruotare, controllare, attivare ecc)? Il task è stato facile o impegnativo, lento o rapido, lasco o faticoso, rilassante o laborioso?

Clicca sul fattore che rappresenta il contributo più importante sul carico di lavoro *
percepito per il task

- ☐ Sforzo - Quanto duro hai dovuto lavorare (mentalmente e fisicamente) per raggiungere il tuo livello di performance?
- ☐ Richiesta fisica - Quanta attività fisica è stata richiesta (es. premere, spingere, ruotare, controllare, attivare ecc)? Il task è stato facile o impegnativo, lento o rapido, lasco o faticoso, rilassante o laborioso?

Parte 3: Ulteriori parametri di valutazione soggettiva

Ulteriori valutazioni [1: Fortemente in disaccordo - 5: Fortemente d'accordo] *

	1	2	3	4	5
Il visore HTC Vive risulta comodo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Il visore HTC Vive risulta ingombrante	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I controller/MANUS Gloves risultano facili da utilizzare	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Avrei preferito altri controller	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ritengo utili i pannelli che mostrano i comandi	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Suggerimenti

Bibliografia

- [1] INTELLIGENZA ARTIFICIALE - Il portale dell'Intelligenza Artificiale. URL: <https://www.intelligenzaartificiale.it/realta-virtuale/>
- [2] Bosch. URL: <https://www.bosch.it/>
- [3] Acatech - Accademia tedesca delle scienze e dell'ingegneria. URL: <https://en.acatech.de/>
- [4] L. Maci, «Che cos'è l'Industria 4.0 e perché è importante saperla affrontare» 15 Settembre 2022. URL: <https://www.economyup.it/innovazione/cos-e-l-industria-40-e-perche-e-importante-saperla-affrontare/>
- [5] «Che cos'è l'INDUSTRIA 4.0?» URL: <https://www.pin.unifi.it/industria4>
- [6] E. Convertini, «Le Smart Technologies alla base della Quarta Rivoluzione Industriale» 26 Ottobre 2018. URL: https://blog.osservatori.net/it_it/smart-technologies-quarta-rivoluzione-industriale
- [7] Competence Industry Manufacturing - CIM 4.0. URL: <https://cim40.com/>
- [8] Politecnico di Torino. URL: <https://www.polito.it/>
- [9] Università di Torino. URL: <https://www.unito.it/>
- [10] General Motors. URL: <https://www.gm.com/>
- [11] Avio. URL: <https://www.avio.com/it>
- [12] Thales Alenia. URL: <https://www.thalesgroup.com/en>
- [13] Stellantis. URL: <https://www.stellantis.com/it>
- [14] MANUS. URL: <https://www.manus-meta.com/>
- [15] Vive. URL: <https://www.vive.com/us/>
- [16] Unity3D. URL: <https://unity.com/>

- [17] «LOW COST AUTOMATION, CHIAMATA KARAKURI KAIZEN» URL: <https://www.utekvision.com/it/blog/low-cost-automation-karakuri-kaizen.html>.
- [18] National Endowment for the Arts. URL: <https://www.arts.gov/>
- [19] Massachusetts Institute of Technology. URL: <https://www.mit.edu/>
- [20] Nintendo. URL: <https://www.nintendo.it/>
- [21] Oculus Rift. URL: https://www.oculus.com/rift-s/?locale=it_IT
- [22] Google Cardboard. URL: <https://arvr.google.com/cardboard/>
- [23] Sony. URL: <https://www.sony.it/>
- [24] Play Station 4. URL: <https://www.playstation.com/it-it/ps4/>
- [25] Play Station VR. URL: <https://www.playstation.com/it-it/ps-vr/>
- [26] Meta Quest Pro. URL: <https://www.meta.com/it/quest/quest-pro/>
- [27] HTC Vive Pro. URL: <https://www.vive.com/us/product/vive-pro/>
- [28] HTC Vive Pro 2.
URL: <https://www.vive.com/us/product/vive-pro2-full-kit/overview/>
- [29] HTC Vive Pro Eye.
URL: <https://www.vive.com/us/product/vive-pro-eye/overview/>
- [30] HTC Vive XR Elite.
URL: <https://www.vive.com/us/product/vive-xr-elite/overview/>
- [31] Play Station VR2. URL: <https://www.playstation.com/it-it/ps-vr2/>
- [32] Play Station 5. URL: <https://www.playstation.com/it-it/ps5/>
- [33] R. a. S. M. a. W. M. C. Skarbez, «Revisiting Milgram and Kishino's Reality-Virtuality Continuum» *Frontiers in Virtual Reality*, vol. 2, 2021.
URL: <https://www.frontiersin.org/articles/10.3389/frvir.2021.647997>
DOI: 10.3389/frvir.2021.647997

- [34] M. A. Muhanna, «Virtual reality and the CAVE: Taxonomy, interaction challenges and research directions» *Journal of King Saud University - Computer and Information Sciences*, vol. 27, n. 3, pp. 344 - 361, 2015.
DOI: <https://doi.org/10.1016/j.jksuci.2014.03.023>
URL: <https://www.sciencedirect.com/science/article/pii/S1319157815000439>
- [35] Electronic Visualization Lab. URL: <https://www.evl.uic.edu/>
- [36] Università dell'Illinois. URL: <https://illinois.edu/>
- [37] Leap Motion Controller.
URL: <https://www.ultraleap.com/product/leap-motion-controller/>
- [38] MIT Media Lab. URL: <https://www.media.mit.edu/>
- [39] Sacco Marco, Stefano Mottura, «Realtà Aumentata e Realtà Virtuale per la competitività delle aziende» *Automazione e Strumentazione*, pp. 36-39, 2012. URL: https://automazione-plus.it/wp-content/uploads/sites/3/2012/10/AS_008_036_0391.pdf
- [40] IKEA VR Experience. URL: https://store.steampowered.com/app/447270/IKEA_VR_Experience/
- [41] IKEA. URL: <https://www.ikea.com/it/it/>
- [42] ShapesXR. URL: <https://www.shapesxr.com/>
- [43] BMW. URL: <https://www.bmw.it/it/home.html>
- [44] «Auto Hand - Asset Store di Unity3D» URL: <https://assetstore.unity.com/packages/tools/game-toolkits/auto-hand-vr-physics-interaction-165323>
- [45] Meta Presence Platform. URL: <https://developer.oculus.com/presence-platform/>.
- [46] Interaction SDK. URL: <https://developers.facebook.com/blog/post/2022/11/22/building-intuitive-interactions-vr/>
- [47] «Meta Presence Platform - HandGrab» URL: <https://developer.oculus.com/documentation/unity/unity-isdk-hand-grab-interaction/>

- [48] «Meta Presence Platform - Poke» URL:
<https://developer.oculus.com/documentation/unity/unity-isdk-poke-interaction/>
- [49] «Meta Presence Platform - HandPose» URL:
<https://developer.oculus.com/documentation/unity/unity-isdk-hand-pose-detection#pose-prefabs>
- [50] «Meta Presence Platform - Gesture» URL:
<https://developer.oculus.com/documentation/unity/unity-isdk-hand-pose-detection/#sequences>
- [51] «Meta Presence Platform - Distance Grab» URL:
<https://developer.oculus.com/documentation/unity/unity-isdk-distance-hand-grab-interaction/>
- [52] «Meta Presence Platform - Touch Hand Grab» URL:
<https://developer.oculus.com/documentation/unity/unity-isdk-touch-hand-grab-interaction>
- [53] «Meta Presence Platform - Ray Interactions» URL:
<https://developer.oculus.com/documentation/unity/unity-isdk-ray-interaction/>.
- [54] Ultraleap. URL: <https://www.ultraleap.com/>
- [55] Kinect. URL: <https://azure.microsoft.com/it-it/products/kinect-dk>
- [56] Unreal Engine. URL: <https://www.unrealengine.com/en-US>
- [57] Hanna-Riikka Rantamaa, Jari Kangas, Sriram Kumar, Helena Mehtonen, Jorma Järnstedt, Roope Raisamo, «Comparison of a VR Stylus with a Controller, Hand Tracking, and a Mouse for Object Manipulation and Medical Marking Tasks in Virtual Reality» *Applied Sciences*, vol. 13, 2023.
DOI: 10.3390/app13042251
- [58] Varjo. URL: <https://varjo.com/>
- [59] Ultraleap Stereo IR 170. URL: <https://www.ultraleap.com/product/stereo-ir-170/>
- [60] Valve Index Controllers. URL: <https://www.valvesoftware.com/it/index/controllers>
- [61] Logitech VR Ink. URL: <https://www.logitech.com/it-it/promo/vr-ink.html>

- [62] Ardhika Wida Pangestu, Clara Hetty Primasari, Thomas Adi Purnomo Sidhi, Yohanes Priadi Wibisono, Djoko Budiyo Setyohadi, «Comparison Analysis of Usability Using Controllers and Hand Tracking in Virtual Reality Gamelan (Sharon) Based On User Experience» *Journal of Intelligent Software Systems*, vol. 1, p. 89, 2022.
DOI: 10.26798/jiss.v1i2.750
- [63] «SUS - System Usability Scale» URL: <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>
- [64] «USE Questionnaire: Usefulness, Satisfaction, and Ease of use»
URL: <https://garyperelman.com/quest/quest.cgi?form=USE>
- [65] Luis Almeida, Eunice Ramos Lopes, Beril Yalçinkaya, Rodolfo Martins, Ana Lopes, Paulo Menezes, Gabriel Pires, «Towards natural interaction in immersive reality with a cyber-glove» 2019.
DOI: 10.1109/SMC.2019.8914239
- [66] HTC Vive Trackers. URL: <https://www.vive.com/eu/accessory/tracker3/>
- [67] HTC Vive Controllers. URL: <https://www.vive.com/eu/accessory/controller/>
- [68] Molina Guillermo, Gimeno Jesús, Portalés Cristina, Casas-Yrurzum Sergio, «A comparative analysis of two immersive virtual reality systems in the integration and visualization of natural hand interaction» *Multimedia Tools and Applications*, vol. 81, 2022. DOI: 10.1007/s11042-021-11760-9
- [69] SteamVR. URL: <https://store.steampowered.com/app/250820/SteamVR/>
- [70] Wireless Adapter. URL: <https://www.vive.com/eu/accessory/wireless-adapter/>
- [71] HTC Vive Trackers (2018).
URL: <https://www.vive.com/nz/accessory/vive-tracker/>
- [72] Xsens. URL: <https://www.movella.com/products/xsens>
- [73] MANUS Prime II - Xsens Edition.
URL: <https://www.manus-meta.com/products/xsens-gloves#xsens-gloves>
- [74] Autodesk 3DS Max. URL: <https://www.autodesk.it/products/3ds-max/free-trial>
- [75] Autodesk Maya. URL: <https://www.autodesk.it/products/maya/free-trial>

- [76] Autodesk MotionBuilder.
URL: <https://www.autodesk.com/products/motionbuilder/free-trial>
- [77] Houdini. URL: <https://www.sidefx.com/products/houdini/>
- [78] Cinema 4D. URL: <https://www.maxon.net/it/cinema-4d>
- [79] Barco UDM 4k15. URL: <https://www.barco.com/it/product/udm-4k15>
- [80] Volfoni Edge VR. URL: <http://volfoeni.com/en/edge-vr/>
- [81] OptiTrack. URL: <https://optitrack.com/>
- [82] OptiTrack PrimeX 22. URL: <https://optitrack.com/cameras/primex-22/>
- [83] Plugin per Unity di MANUS.
URL: <https://documentation.manus-meta.com/v1.9.0/unity-core/index.html>
- [84] Reply. URL: <https://www.reply.com/it>
- [85] Asset Store di Unity. URL: <https://assetstore.unity.com/>
- [86] Microsoft. URL: <https://www.microsoft.com/it-it/>
- [87] Visual Studio. URL: <https://visualstudio.microsoft.com/it/>
- [88] «Plugin SteamVR per Unity» URL:
<https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647>
- [89] MANUS Core. URL: <https://blog.manus-vr.com/unreal-engine-plugin/>
- [90] Software Motive. URL: <https://optitrack.com/software/motive/>
- [91] Calibration Wand. URL: <https://optitrack.com/accessories/calibration-tools/>
- [92] Calibration Tools. URL: <https://optitrack.com/accessories/calibration-tools/>
- [93] Libreria Mirror per Unity. URL: <https://mirror-networking.gitbook.io/docs/>
- [94] «Measuring and Interpreting System Usability Scale (SUS)» URL:
<https://uiuxtrend.com/measuring-system-usability-scale-sus/>

- [95] Sandra G. Hart and Lowell E. Staveland, «Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research» in *Human Mental Workload*, vol. 52, North-Holland, 1988, pp. 139 - 183.
DOI: [https://doi.org/10.1016/S0166-4115\(08\)62386-9](https://doi.org/10.1016/S0166-4115(08)62386-9)
URL: <https://www.sciencedirect.com/science/article/pii/S0166411508623869>