Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

Progettazione e realizzazione di una applicazione di Realtà Virtuale immersiva utilizzando la tecnologia HTC Vive.

Relatore:
Chiar.mo Prof.
Lorenzo Donatiello

Presentato da:
Marco Valerio Manzini

Co-Relatore:
Dott.
Gustavo Marfia

III Sessione
A.A. 2015/2016

"When one path to your destiny is blocked,
another will appear "

[Gabriel Knight 3: Blood of the sacred blood of the damned.]

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

L'attuale tecnologia, sviluppatasi negli ultimi 20 anni, ha permesso l'introduzione nel mercato di consumo diversi dispositivi di Realtà Virtuale (VR) a basso costo, grazie ai quali negli ultimi sette anni si è visto un'esplosione di applicazioni legate ai campi più disparati, di cui i principali sono: intrattenimento, addestramento, apprendimento, commerciale, e perfino in campo medico.

Il lavoro di tesi descrive la realizzazione di una applicazione immersiva di Realtà Virtuale basata su Head-Mounted Display(HMD) HTC Vive e piattaforma Game Engine Unity3D 5.5.0f3. Per sviluppare tale applicazione è stato creato un ambiente di sviluppo che si intende utilizzare per la realizzazione di altre applicazioni. Nel lavoro di tesi vengono anche presentate le caratteristiche delle tecnologie attualmente disponibili sul mercato.

La struttura della tesi prevede nella prima parte la definizione dello **Stato dell'arte**:

- **Definizioni di Realtà Virtuale**: contenente le principali interpretazioni e definizioni di VR di numerosi autori esperti nel settore;

- **Storia della Realtà Virtuale**: un excursus cronologico di fallimenti, successi e delle principali tecnologie di VR sviluppate nel corso della storia;

- **Panoramica dei dispositivi**: il confronto tra i principali HMD attualmente in commercio mostrandone pregi e difetti sia sul fronte prestazionale che applicativo, con particolare enfasi sull'HMD HTC Vive sviluppato da Valve Corporation;

- **Campi di applicazione**: la presentazione dei principali campi in cui la VR è applicata, comprensiva di esempi dettagliati.

Nella seconda parte della tesi vengono descritti i punti salienti dello sviluppo e progettazione generale di applicazioni in **Realtà Virtuale**:

- **Concetti Teorici**: Immersione, Presenza, e le linee guida alla progettazione di una "buona"applicazione in VR;

- **Percezione dell'utente**: come il sistema uditivo e visivo degli utenti viene influenzato all'interno di un mondo virtuale;

- **Effetti nocivi della VR**: principali conseguenze e soluzioni legate all'uso sbagliato della VR o ad una applicazione sviluppata senza rispettare alcune linee guida fondamentali;

- **Realtà Aumentata vs. Realtà Virtuale**: un confronto conclusivo sull'uso di VR e Realtà aumentata (AR) in diversi contesti, corredato da esempi e applicazioni note del settore.

La terza e ultima parte della tesi tratterà la **Creazione dell'applicazione in VR *Fashion Island VR***:

- **Costituzione del laboratorio**: descrizione delle tecnologie utilizzate per costituire il laboratorio e le motivazioni che hanno portato a scegliere tali tecnologie;

- **Progettazione dell'applicazione**: descrizione della fase progettuale, contenente le scelte e i concetti teorici alla base della applicazione stessa;

- **Dettagli implementativi dell'applicazione**: descrizione di ciascun aspetto implementativo dell'applicazione con riferimenti a parti di codice e librerie esterne di sviluppo.

Parte I

Stato dell'arte

Capitolo 1

Definizioni di Realtà Virtuale

Il termine *Realtà Virtuale* è comunemente usato dai media per descrivere mondi immaginari che esistono solo nei nostri computer o nelle nostre teste. Andiamo ora a definire con precisione cosa significa secondo l'interpretazione di diversi esperti nel settore.

- Ivan Sutherland, creatore di uno dei primi sistemi di VR al mondo nei primi anni sessanta dichiarò: "*Il sistema di VR finale sarebbe, naturalmente, una camera entro cui il computer può controllare l'essenza stessa della materia. Una sedia visualizzata in tale camera sarebbe abbastanza buona per sedersi. Delle manette visualizzate in una camera così sarebbero imprigionanti, e un proiettile visualizzato in una camera così sarebbe fatale.*"[1]

- Fuchs e Bishop, 1992, "*Grafica real-time interattiva con modelli tridimensionali, combinata con una tecnologia di visualizzazione che offre all'utente l'immersione nel mondo modellato e la sua manipolazione diretta*"[2]

- Gigante, 1993, "*L'illusione di una partecipazione in un ambiente sintetico piuttosto che l'osservazione esterna di tale ambiente. VR fa affidamento su un Head-tracker display tridimensionale e stereoscopico,*

su un hand-tracking/body-tacking e sul suono binaurale. VR è una esperienza immersiva multi-sensoriale."[3]

- Cruz, 1993, "*VR si riferisce ad ambienti immersivi, interattivi, multi-sensoriali, tridimensionali, generati al computer e la combinazione di tecnologie richieste per costruire questi ambienti.*"[4]

- Più recentemente il sito web merriam-webster.com, nel 2015, ha definito la VR come "*un ambiente virtuale che viene sperimentato attraverso stimoli sensoriali (come immagini e suoni) forniti da un computer, e in cui le proprie azioni in parte determinano ciò che accade nell'ambiente.* "[5]

- Infine, Jason Jerald nel suo libro definisce la VR come "*un ambiente digitale generato dal computer che può essere vissuto ed interagito come se quell'ambiente fosse reale.*"[6]

  Sebbene ci siano alcune differenze tra queste definizioni, sono assolutamente equivalenti, definiscono tutte un ambiente virtuale generato per mezzo del computer nel quale l'utente può interagire con esso, e vedere i risultati delle sue interazioni per mezzo di stimoli visivi e sonori immersivi.

Capitolo 2

Storia della Realtà Virtuale

In questo capitolo farò un excursus cronologico di fallimenti, successi e delle principali tecnologie di VR sviluppate nel corso della storia dagli inizi del 1800 fino agli anni 2000.

2.1 La nascita, nell' 800

I primi sviluppi in questo campo avvennero nel 1832 da parte di *Sir Charles Wheatstone*, il quale inventò lo stereoscopio, un dispositivo che usa degli specchi angolati di 45° per riflettere le immagini all'interno dell'occhio dal lato sinistro e destro ampliandone così il campo visivo.

Figura 2.1: Stereoscopio di Charles Wheatstone

David Brewster, che precedentemente inventò il caleidoscopio, usò le lenti per creare un piccolo stereoscopio portatile nel 1851, il quale stimò di averne

venduti circa mezzo milione.[7] Questa prima frenesia per il 3D incluse numerose versioni autoprodotte con cartoni, inoltre come si può vedere dalla figura 2.2 il design è concettualmente uguale all'odierno Google Cardboard o ai sistemi di VR basati su smatphone.

Figura 2.2: Stereoscopio di Brewster Wheatstone (1860)

2.2 Il crollo, il '900

Gli sviluppi in campo di VR continuarono per tutto il '900, se ne vede una sua prima applicazione nel 1916 in campo militare, Pratt [8] ideò un elmetto completo di sistema di puntamento a periscopio e fuoco grazie al quale il soldato era capace di sparare soffiando semplicemente in una cannula che reggeva con la bocca.

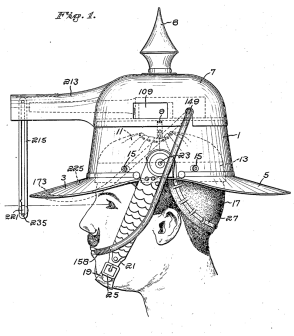

Figura 2.3: Elmetto di Pratt (1916)

Poco dopo una decade dall'invenzione di Pratt, Edwin Link sviluppò il primo simulatore di volo meccanico (1928), completo di base mobile.

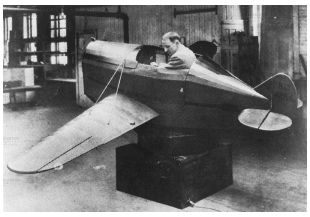

Figura 2.4: Primo simulatore di volo di Edwin Link (1928)

Solo nel 1960 Morton Heiling progettò il primo Head-Mounted Display (HMD)[9] capace di dare un campo di visione (FOV) di 140° orizzontale e verticale, dotato di cuffie stereo e ugelli di scarico dell'aria che forniscono un senso di brezza a diverse temperature o profumo.

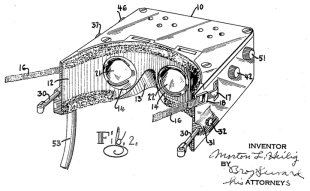

Figura 2.5: HMD di Heiling (1960)

Heiling progettò inoltre il così detto Sensorama che dava allo spettatore una visione immersiva del film, fornendo una visione stereoscopica con ampio FOV, suono stereo, vibrazioni, profumi e vento.

Figura 2.6: Sensorama (1960)

Nel 1961 la Philco Corporation costruì il primo modello funzionante di HMD con incluso il sistema di posizionamento della testa, l'utente muovendo la testa in una certa direzione permetteva ad una videocamera in un'altra stanza di muoversi contestualmente al movimento dell'utente.

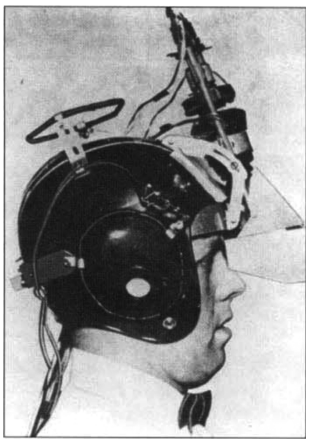

Figura 2.7: HMD Philco Corp. (1961)

Nel 1985 Scott Fisher, con altri ricercatori della NASA svilupparono il primo HMD commercialmente vendibile chiamato VIVED che successivamente fu riprogettato e rinominato VIEW nel 1988, era dotato di un sistema di visione ad ampio FOV e suono 3D permettendo un immersione totale dell'utente.

Figura 2.8: HMD View della NASA. (1988)

La VR esplose negli anni 90 in cui numerose compagnie concentrate soprattutto sul mercato della ricerca professionale e dell'intrattenimento die-

dero il loro contributo, si possono citare le famose compagnie: Sega, Disney, General Motors. Nel 1993 *Wired* predisse che nel giro di cinque anni una persona su dieci indosserà un HMD mentre viaggia in bus, treno o in aereo[10]. Purtroppo la predizione fu errata di una decina di anni, a causa della scarsa potenza computazionale che richiedeva, nel 1996 l'industria della VR andò a picco e cominciò a contrarsi lentamente fino ad andare in disuso nel 1998.

2.3 La rinascita, gli anni 2000

La prima decade del ventunesimo secolo è conosciuto come il "VR winter"durante il quale compagnie, governi, università, e laboratori militari in tutto il mondo continuarono la ricerca in campo di VR.

Ciò che mancava agli HMD del '900 era un largo FOV, senza il quale gli utenti non riuscivano a percepire la "magica"sensazione della presenza.[6], a tal proposito si crearono dispositivi con FOV di 150°, come il Wide5, e si studiò più approfonditamente l'effetto della FOV sugli utenti. Da questi studi ne venne fuori, per esempio, che gli utenti potevano giudicare le distanze in maniera più accurata quando avevano una FOV maggiore.[11]

Solo nel 2012 si vide la vera rinascita di questi dispositivi con l'uscita del famoso Oculus Rift, iniziato come progetto di un piccolo laboratorio, e infine comprato da Facebook nel 2014 per 2 miliardi di dollari, segnando così la nascita di una nuova era per il mercato della VR che negli ultimi quattro anni ha raggiunto globalmente la quota di 4.5 miliardi di dollari, e si stima che entro il 2020 raggiungere i 105.2 miliardi di dollari.[12]

Figura 2.9: Oculus Rift DK1 (2012)

Capitolo 3

Panoramica dei dispositivi

In questo capitolo presenterò i principali dispositivi HMD presenti in commercio, partirò descrivendo Oculus Rift, in seguito Google Daydream, infine HTC Vive usato per il mio progetto di tesi.

3.1 Oculus Rift

Dopo essere stato comprato da Facebook per la modica cifra di 2 miliardi di dollari, Oculus Rift si è trasformato ed evoluto e oggigiorno si posiziona tra i dispositivi di realtà virtuale di alta gamma non solo per il suo prezzo di mercato ma anche per le sue performance e caratteristiche che vedremo nel dettaglio.[13]

Figura 3.1: Oculus Rift CV1 (2016)

3.1.1 Setup iniziale

L'Oculus ha il beneficio di essere molto semplice, sin dall'apertura della scatola si può notare la semplicità delle componenti e la facilità di utilizzo a differenza di altri HMDs, essa contiene:

- HMD Oculus con cuffie integrate
- Sensore Oculus
- Oculus Remote (Controller)
- Controller Xbox One Wireless
- Cavi di collegamento dell'HMD

L'assemblaggio è piuttosto semplice dal momento che è solo un HDMI e un paio di cavi USB da collegare, oltre alla configurazione e al posizionamento del sensore che può essere posizionato su una scrivania senza bisogno di alcun fissaggio al muro, il tutto viene guidato attraverso il software di setup scaricabile dal sito web https://www.oculus.com/en-us/setup/.

3.1.2 Il visore

Il visore monta due lenti AMOLED con una risoluzione di 1200x1080 ciascuna e una frequenza di aggiornamento di 90 Hz, ciò permette di creare delle esperienze in VR molto fluide e immersive a discapito però di grandi necessità computazionali, si consumano infatti 233 milioni di pixel al secondo, rispetto i classici 123 milioni di un gioco con risoluzione 1080x1900 con frequanza di aggiornamento di 60 Hz. Rift utilizza una coppia di lenti di Fresnel ibride personalizzate che gli conferiscono un FOV abbastanza ampio. Oculus non specifica esattamente quanto è il FOV, ma hanno dichiarato essere maggiore del modello DK2 - che aveva un FOV di 100 gradi.

Il visore possiede delle cuffie integrate che permettono un audio pulito e in stereo, sostituibili con le classiche cuffie da walkman, a discrezione dell'utente.

3.1.3 Il Controller e il sensore

Nella dotazione iniziale è presente un controller per XBox utile soprattutto per lo sviluppo di applicazioni per Oculus, per l'utente medio è stato inserito un più piccolo e pratico controller: L'Oculus Remote che permette la navigazione dell'ambiente e l'interazione di esso per mezzo di semplici bottoni e levette analogiche. Purtroppo L'Oculus Remote come il controller Xbox non permettono di "Usare le proprie mani dentro l'applicazione", ma vedremo una soluzione a ciò in seguito.

Il sensore permette un puntamento dell'HMD molto preciso attraverso l'uso di raggi infrarossi che identificano i tre led infrarossi posizionati sul HMD stesso, ciò permette di potersi muovere in una stanza molto larga (3m x 3,5m) e fino a che il sensore riesce a puntare anche uno solo dei tre led la precisione di puntamento non viene penalizzata.

3.1.4 Specifiche tecniche e hardware raccomandato

Ecco l'elenco completo delle specifiche tecniche dell'hardware necessario che sono state ufficialmente confermate da Oculus:

- **Scheda grafica**: NVIDIA GTX 970 / AMD R9 290 equivalente o superiore;
- **Processore**: Intel i5-4590 equivalente o superiore;
- **Memoria**: 8 GB di RAM;
- **Uscite**: uscita 1.3 video compatibile con HDMI
- **Sistema operativo**: Windows 7 SP1 a 64 bit o più recente;
- **Ingressi**: 3x porte USB 3.0, 1x USB 2.0.

Purtroppo queste specifiche permettono di raggiungere un esperienza in VR minimale, per riuscire a gustarsi a pieno l'esperienza occorre investire su schede grafiche migliori, meglio se duplici con l'abilitazione a lavorare contempo-

raneamente (Scalable Link Interface Enabled), e sicuramente aumentare la potenza del processore.

3.1.5 Oculus Touch

Venduti a parte è possibile acquistare i controller Touch che permettono di "Utilizzare le proprie mani"all'interno dell'applicazione in uso, per mezzo di un ulteriore sensore e un sistema di puntamento simile a quello del visore è possibile afferrare, colpire, lanciare, sparare con Oculus Touch introducendo nell'esperienza in VR un ulteriore elemento per interagire con l'applicazione, aumentando di conseguenza il livello di immersività della stessa.

Figura 3.2: Oculus Touch (2016)

3.2 Google Daydream

Se si è in possesso di uno smartphone Android di ultima generazione come: Google Pixel, Moto z, Asus ZenPhone AR, Huawei Mate 9 Pro, ZTE Axon 7 allora grazie a Google Daydream è possibile vivere un'esperienza in VR semplicemente sfruttando le potenzialità del proprio smartphone e del visore acquistabile per poco più di 80$.

3.2.1 Il visore

L'uso del visore è ridicolmente semplice. Per installare Daydream e usarlo, è sufficiente posizionare il proprio smartphone nell'apposito alloggiamen-

Figura 3.3: Google Daydream (2016)

to, il quale possiede delle piccole sporgenze capacitive che aiutano a rilevare dove lo schermo deve essere centrato, mentre un chip NFC (integrato nell'alloggiamento) permette di avviare le applicazioni desiderate.

Le specifiche di FOV e angolo di visione non sono state rilasciate da Google ma sono approssimativamente 90° secondo Wired.com[14], mentre le specifiche di frequenza di aggiornamento dello schermo dipendono dal telefono stesso. Per Google Pixel, si ha una risoluzione di 1080 x 1920 pixel (441ppi) a 60Hz e mentre per la versione XL la risoluzione è di 1440 x 2560 pixel (534ppi) a 60Hz.

3.2.2 Controller

Il controller Daydream View è più grande di un controller di Oculus Rift e più piccolo di un Wiimote, una volta usato può essere riposto nell'apposito alloggiamento all'interno del visore.

Il controller si adatta comodamente al palmo della mano ed è ultra leggero e resistente agli urti accidentali.

Sono presenti solo cinque pulsanti, tra cui un trackpad che funge anche da tasto, sotto di esso, il pulsante di applicazione, che se premuto fa comparire il menu, il pulsante per mettere in pausa, tornare indietro o cambiare modalità a seconda della applicazione stessa. Infine è presente il tasto home, che se premuto riporta alla home di Google Daydream e centra la visuale, è possibile inoltre scorgere sul lato destro del controller i pulsanti del volume.

E' presente una porta USB-C sul fondo del controller che permette di caricare la batteria, la quale secondo Google ha una durata media di 12 ore, in ogni caso è presente un indicatore di stato nella parte inferiore del controller che lampeggia tre volte dopo aver premuto il tasto home e consente di sapere se deve essere ricaricato.

Il rilevamento del controller avviene tramite tecnologia Bluetooth e l'uso di accelerometri e giroscopi, diverse recensioni riportano che purtroppo durante l'esperienza occorre fermarsi spesso per ricalibrare il controller che tende a vagare con mente propria.[15]

Figura 3.4: Google Daydream Controller (2016)

3.3 HTC Vive

HTC Vive è prodotto dalla Valve Corporation, uscito sul mercato ad aprile 2016, a differenza degli HMDs visti in precedenza, esso permette una configurazione della stanza in "Room Scale"consentendo un ampio ambiente di gioco, ed inoltre da la possibilità di avere un fedele tracciamento delle

mani dell'utente per mezzo di due controller ergonomici venduti assieme al visore.

Figura 3.5: HTC Vive, controller e sensori (2016)

3.3.1 Setup iniziale

Il montaggio iniziale è piuttosto elaborato rispetto i precedenti HMDs, prima di tutto i sensori o base-stations vanno posizionati ad una altezza di almeno due metri, posti in diagonale l'uno rispetto l'altro, è consigliabile fissarli al muro, o come nel mio caso, sfruttare dei tripodi mobili per poterli spostare liberamente nella stanza. Dopo di che occorre collegare il visore attraverso due cavi USB e un cavo HDMI al computer, infine è necessario scaricare e far partire il software di configurazione che guiderà l'utente all'impostazione dell'area di gioco, che può arrivare fino ad un area 5m x 5m e alla calibrazione del visore e dei controller.

3.3.2 Il visore

Il visore monta due lenti di Fresnel Amoled con una risoluzione di 1080x1200 per occhio le quali hanno una frequenza di aggiornamento di 90 Hz ed un off-screen rendering di circa 1.4x in ogni dimensione ottenendo così un frame buffer di 1512x1680 per occhio, permettendo una esperienza in VR ultra realistica; ovviamente tale risoluzione si paga in termini di pixel al secondo che raggiungono i 457 milioni, abbassabili a 378 con alcuni accorgimenti spiegati da Alex Vlachos dutrante il GDC 2015 [16]. La FOV del visore è superiore

ai 110° superando di gran lunga i precedenti HMDs e riuscendo ad ottenere una immersività totale dell'utente.

Sebbene possieda un alloggiamento per le classiche cuffie da walkman il visore non ha integrato alcun dispositivo audio, ciò però è previsto nel futuro come vedremo successivamente.

3.3.3 I controller e i sensori

I controller, unici nel loro genere, permettono un tracciamento fedele delle posizioni delle mani, o di qualsiasi cosa sia attaccato ad essi. Nella parte frontale a partire dall'alto troviamo:

- **Il tasto Applicazione**, usato, nella maggior parte delle applicazioni che ho provato per aprire menu a scomparsa o utlity di gioco;
- **Il trackpad**, esso permette il riconoscimento della posizione dell'utente con coordinate cartesiane, inoltre il trackpad funge anche da bottone;
- **Il tasto Menu**, usato per richiamare la schermata di home o per accendere e spegnere i controller;
- **Il tasto trigger**, posizionato sul retro del controller, il quale assomiglia ad un grilletto di una pistola;
- **Tasti di Grip**, posti lateralmente rispetto l'impugnatura del controller, e solitamente usati nelle applicazioni per afferrare oggetti.

I sensori posizionati l'uno in diagonale rispetto all'altro e con canali di comunicazione diversi, permettono il tracciamento per mezzo di fasci di raggi infrarossi a 120 Hz raccolti da un totale di 70 sensori posti sul visore e sui controller che comunicano la posizione all'applicazione attraverso Bluetooth Low Energy (BLE) o cavo HDMI al computer, il quale interpola le posizioni registrate dai due sensori e restituisce la posizione finali al visore.

Il risultato è che il Vive non solo monitora come la testa si muove verso l'alto, in basso e lateralmente, ma traccia anche dove ci si trova in una stanza e quali movimenti le mani stanno facendo in relazione a dove si sta in piedi.

3.3.4 Specifiche Tecniche e Hardware

Ecco l'elenco completo delle specifiche tecniche dell'hardware necessario che sono state ufficialmente confermate da SteamVR:

- **Scheda grafica**: NVIDIA GTX 1060 equivalente o superiore;
- **Processore**: Intel i5-4590 equivalente o superiore;
- **Memoria**: 16 GB di RAM o superiore;
- **Sistema operativo**: Windows 7 SP1 a 64 bit o più recente;
- **Ingressi**: 3x porte USB 3.0, 1x USB 2.0, 1x HDMI.

Come si può notare le caratteristiche minime sono nettamente superiori a quelle di Oculus, tali infatti permettono di avere una buona esperienza in VR in termini di Frame Per Seconds (FPS), migliorabile se vengono aggiornate le GPU usando anche multiple GPU (SLI enabled o CrossFire) come consigliato da Alex Vlachos durante il GDC del 2015[16].

3.3.5 Sviluppi futuri

Durante il CES 2017 il mondo della VR è stata la vera protagonista della mostra, anche Valve Corporation ha mostrato cosa ci aspetta nel futuro dell'HTC Vive, in particolare:

- **Vive Tracker**, un piccolo dispositivo che consente agli sviluppatori di tenere traccia praticamente di qualsiasi oggetto nella realtà virtuale. Tracker potrà essere integrato negli accessori di terze parti ed apre interessanti scenari nello sviluppo di applicazioni e giochi;
- **TPCast**, è un dispositivo che permette di usare HTC Vive senza l'uso dei fastidiosi cavi posteriori, perciò in modalità wireless. L'adattatore TPCast precedentemente disponibile solo in Cina verrà commercializzato a livello globale. Sarà disponibile a partire dal secondo trimestre

Figura 3.6: Vive Tracker (2017)

del 2017, e costerà 249 dollari. Questo dispositivo ha autonomia di circa 90 minuti, ma più avanti sarà disponibile anche un battery pack XL in grado di garantire 5 ore di durata;

Figura 3.7: TPCast (2017)

- **Vive Deluxe Audio Strap**, consiste in cuffie regolabili integrate, proprio come l'Oculus Rift, applicabili al HTC Vive per mezzo di comodi nastri con velcro.

Figura 3.8: Vive Deluxe Audio Strap (2017)

Modello	**HTC vive**	**Oculus Rift**	**Google Daydream**
Display	OLED	OLED	Dipendente dal dispositivo
Risoluzione	2160x1200 e off-screen rendering di 1.4x	2160x1200	Dipendente dal dispositivo
Refresh Rate	90Hz	90Hz	60Hz
FOV	più di 110°	110°	Dipendente dal dispositivo
Audio	Cuffie non integrate	Cuffie Integrate	Cuffie Integrate
Vantaggi	• Alta FOV • Ottima risoluzione • Presenza dei controller	• Buona FOV e risoluzione; • Cuffie integrate	• Economico • Semplice all'uso
Svantaggi	• Costo elevato • Cavi ingombranti • Richiede GPU di alta gamma	• Costo elevato • Assenza di controller • Richiede GPU di alta gamma	• Tracciamento del visore e del controller instabili • FOV limitata

Tabella 3.1: Tabella di confronto tra HMDs

3.4 Confronto tra HMDs

Quale di questi HMDs, mostrati nella tabella 3.1 sia il migliore spetta all'utente giudicare in base all'applicazione che userà e quanto vuole spendere. Secondo la mia modesta opinione HTC Vive offre una esperienza completa e immersiva, le applicazioni se ben progettate sono fluide e grazie alla grande quantità di sensori è quasi impossibile uscire dal tracciamento, infine grazie alle migliorie che si vedranno a metà anno 2017 l'esperienza diventerà sempre più vicina alla definizione di VR di Sutherland [1]; ovviamente tutti questi benefici devono essere pagati in termini di prestazioni del proprio computer e dimensione del proprio portafogli.

Capitolo 4

Campi di applicazione

In questo capitolo mostrerò una lista dei principali campi in cui la VR è applicata, comprensiva di esempi dettagliati per ciascun campo e le particolarità di ciascuna applicazione.

4.1 Campo Videoludico

La realtà virtuale attualmente conta tre le sue fila una innumerevole quantità di applicazioni videoludiche, si pensi solo alle più di 800 applicazioni su Steam e le quasi 100 app per Google Daydream, ne spicca sicuramente una che può fare da modello per le future generazioni di videogiochi in VR.

4.1.1 Eagle Flight - Ubisoft

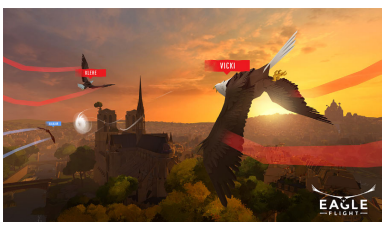

Figura 4.1: Eagle Flight (2017)

Sono passati cinquant'anni da quando l'umanità è sparita dalla Terra: la fauna e la natura si sono riprese le città, trasformando Parigi in un ambiente in cui gli animali la fanno da padroni. Nei panni di un'aquila, sorvoleremo monumenti come la Torre Eiffel e la cattedrale di Notre Dame e scenderemo in picchiata nei vicoli per combattere gli avversari e proteggere il territorio. Eagle Flight ci da l'incredibile possibilità di scoprire Parigi librandoci nei suoi cieli.

L'esperienza è portata anche in multigiocatore (fino a 6 partecipanti) usando le proprie abilità di volo si può condurre la propria squadra alla vittoria in due modalità multigiocatore.

Se si è sempre desiderato vedere Parigi dal punto di vista di un volatile o vivere il brivido di cacciare come un rapace, Eagle Flight ci permette di realizzare questo sogno.

L'innovazione di questo videogioco risiede nella sua modalità di controllo del movimento e allo stesso tempo la quasi totale eliminazione di sgraditi effetti collaterali come la Motion sickness. Il continuo movimento dell'aquila nel mondo di gioco, le sue accelerazioni in picchiata, i rapidi giramenti, tutti controllati con la testa dell'utente avrebbero portato a inevitabili effetti collaterali se gli sviluppatori non avessero adottato tecniche appositamente studiate da ricercatori della NASA per ridurre la nausea degli astronauti tramite giochi di luce, come spiega Olivier Palmieri Game Director presso Ubisoft Montréal[17]. Il movimento è altamente controllato con spostamenti di tipo continuo e quando l'utente decide di cambiare velocità o angolazione ciò viene fatto molto gradualmente dando all'utente la sensazione dell'accelerazione facendolo focalizzare sulla parte di schermo laterale che gradualmente si offusca e diventa nero, bloccando di fatto la visione periferica degli utenti creando il così detto "effetto tunnel"e, come hanno studiato gli sviluppatori Ubisoft, ciò permette al cervello degli utenti di concentrarsi su un solo obiettivo, riducendo o addirittura eliminando gli effetti della Motion Sickness.

4.2 Campo Militare

La realtà virtuale è stata adottata nel campo militare, da esercito, marina e aviazione, per scopi di formazione. Ciò è particolarmente utile per l'addestramento dei soldati in situazioni di combattimento o altre impostazioni pericolose in cui devono imparare a reagire in modo appropriato. Una simulazione di realtà virtuale permette loro di addestrarsi, ma senza il rischio di morte, di un grave infortunio, spreco di risorse come carburante, munizioni o apparecchiature mediche. Possono rivivere uno scenario particolare, per esempio ingaggiare un combattimento con un nemico in un ambiente specifico, ma senza i rischi del mondo reale. Questo metodo si è dimostrato essere più sicuro e meno costoso rispetto ai metodi tradizionali di formazione.

4.2.1 Situazione negli USA

Il paese con l'addestramento in VR più avanzato sono gli Stati Uniti, dove queste tecnologie sono nate. Dal 2012, l'Esercito statunitense impiega per il training il sistema DSTS — Dismounted Soldier Training System — sul quale ha investito la bellezza di 57 milioni di dollari, una combinazione di armi realistiche e sensori corporali che trasmettono i movimenti delle reclute nell'ambiente, offrendo inoltre agli istruttori la preziosa possibilità di riesaminare il comportamento dei partecipanti a missione conclusa.

Figura 4.2: DSTS (2012)

Nel frattempo, anche il corpo dei Marines e la Marina Militare si stanno dando da fare, rispettivamente con il progetto AITT — Augmented Immersive Team Training — e BEMR — Battlefield Exploitation of Mixed Reality - , focalizzati su tecnologie non immersive basate sulla parziale visualizzazione dell'ambiente reale.[18]

Figura 4.3: BEMR (2012)

4.2.2 Situazione in Italia

L'Areonautica Militare nella base aerea di Ghedi (BS) sta impiegando simulatori di VR con riproduzioni di velivoli reali Tornado IDS per l'addestramento dei piloti, che possono così provare a volare seguendo tutte le dovute procedure in un ambiente simulato che ricrea esattamente ambienti reali come veri aereoporti militari italiani. Questo simulatore riproduce fedelmente tutte le missioni, emergenze e situazioni tipiche che si possono incontrare volando con il Tornado, garantendo al tempo stesso un notevole risparmio economico.

Anche la Marina Militare Italiana utilizza la realtà virtuale per l'addestramento, ma in questo caso concentrandosi sullo sviluppo delle capacità procedurali e comunicative. La Marina ne ha commissionato un adattamento come gioco mobile (disponibile per iOs e Android) giocabile in modalità VR, che permette ai giocatori di interagire con mezzi reali e attuali della Marina Militare assumendo il comando di navi, aerei ed elicotteri, in scenari

italiani reali accuratamente ricreati, incluse la navigazione e l'esplorazione della nave scuola Amerigo Vespucci.

Figura 4.4: Marina Militare - Italian Navy Sim (2016)

Infine il Corpo Forestale ha sviluppato "Forest Fire Area Simulator", una simulazione VR che serve a formare e addestrare figure professionali impiegate per contrastare gli effetti degli eco-reati. Diverse aree boschive aggredite dal fuoco vengono riprodotte insieme a tutti i sistemi e le attrezzature ordinariamente utilizzati nelle reali operazioni.

Figura 4.5: Forest Fire Area Simulator (2015)

Sempre in Italia, nel 2014, la divisione Airborne & Space System di Finmeccanica — denominata Leonardo — ha intrapreso una collaborazione con il centro di ricerca dell'università Bicocca CESCOM per sviluppare un progetto in cui la Realtà Virtuale viene sfruttata per il training di abilità come la presa di decisione sotto stress—indirizzato al personale militare e ad operatori di emergenza. Ne è nata una collaborazione che ha portato, ad aprile 2016, la *Dott.ssa Federica Pallavicini*, alla guida del progetto, a partecipare

ad un incontro della NATO per una sessione dedicata a nuove soluzioni per il training di personale militare in condizioni di stress. Ora Il team vorrebbe portare avanti il progetto proponendolo ad altro personale interessato, come, ad esempio, la Protezione Civile o qualsiasi ambito professionale che richieda una certa abilità decisionale o determinate capacità di agire in condizioni di stress estremo.[18]

4.3 Campo Medico

La realtà virtuale in medicina è stata adottata per diversi scopi dal campo chirurgico, alla cura della sindrome da stress post-traumatico, al semplice apprendimento di nozioni per aspiranti medici, vediamone ora alcuni esempi di eccellenza italiani ed esteri.

4.3.1 Situazione in italia, CAVE - Forge Reply

È la versione 2.0 della riabilitazione fisica e cognitiva, la tecnologia messa in campo dall'IRCCS Auxologico Italiano. L'Istituto ospedaliero - specializzato anche nella ricerca sulla Realtà Virtuale Immersiva applicata alle terapie riabilitative - ha presentato nel corso dell'anno 2016 l'atteso progetto che ha condotto alla realizzazione dei *Cave*, due ambienti adibiti per l'attuazione di programmi virtuali a sostegno dell'intervento terapeutico.

Grazie al "Cave", una intera stanza virtuale dove si sperimenta la Telepresenza Immersiva Virtuale (TIV), è possibile simulare i tipici scenari in cui vengono trattati alcuni disturbi tra cui quelli cognitivi nelle fasi iniziali, motori quali quelli conseguenti a ictus e Parkinson, o psicologici come ansia, fobie, stress. Questo approccio innovativo consente, infatti, di migliorare l'efficacia dei programmi di riabilitazione per alcune funzioni compromesse, al fine di gestire, superare o ridurre tali deficit e consentendo al paziente di beneficiare del supporto terapeutico negli ambienti in cui si sviluppano comunemente i disturbi.

Dal punto di vista strettamente tecnologico, il Cave è un sistema integrato che permette di ricostruire una realtà vera, considerando le sollecitazioni cognitive, uditive e visive (e nello sviluppo del progetto pure olfattive e tattili). Grazie alla visione 3D stereoscopica, legata a un sistema di tracciamento della posizione, il sistema permette una corretta lettura degli spazi, dei volumi e delle distanze, dando così la netta sensazione di essere immersi all'interno della scena virtuale proiettata sugli schermi.

Il progetto del Cave è stato realizzato in intesa con il Ministero della Salute e in conto capitale di circa 1 milione di euro (50% a carico della struttura ospedaliera milanese). l'Auxologico si occupa, infatti, da oltre vent'anni della ricerca nelle applicazioni cliniche della Realtà Virtuale.

Forge Reply è il partner tecnologico scelto da Auxologico nella progettazione e realizzazione del *Cave*, mettendo a disposizione competenze uniche ed altamente specializzate sull'integrazione di sistemi di realtà virtuale immersiva, declinandole per la prima volta sul territorio nazionale in ambito medicale.[19]

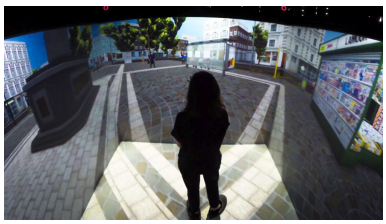

Figura 4.6: CAVE (2016)

4.3.2 Situazione negli USA

Ha fatto notizia il caso di *Erin Martucci*, 40 anni, la prima donna al mondo che ha partorito usando come analgesico la realtà virtuale. Davanti agli occhi le passavano le immagini di un falò al tramonto, su una spiaggia deserta, con il mare placido che mormorava accanto. Quando il suo ginecologo, *Ralph Anderson*, dell'Orange Regional Medical Centre di New York, che nel

frattempo la stava monitorando, le ha detto che poteva togliere gli occhiali perché era il momento di spingere, sembra che Erin sia rimasta sorpresa. «La paziente ha tenuto il visore per due ore durante il travaglio, non voleva un'epidurale e io le ho offerto un'esperienza di rilassamento con la realtà virtuale (...) Siamo ancora agli inizi, ma credo che la realtà virtuale diventerà un'alternativa molto richiesta», afferma il Dott. Anderson.[20]

Secondo un rapporto pubblicato nel 2016 su Psychology of Consciousness[21] , soggetti trattati con la realtà virtuale avrebbero sentito un calo del dolore molto elevato (lo studio parla di una riduzione media dell'82%). Un altro lavoro dell'ospedale Cedars-Sinai di Los Angeles parla di una diminuzione media del dolore del 25%, rispetto a una del 13% legata alla visione di un semplice video rilassante.[22]

«Ormai esistono abbastanza studi su questo impiego della realtà virtuale in ambito medico e possiamo usarla con tranquillità. È quasi priva di effetti collaterali, a parte qualche caso di sensazione di nausea o un temporaneo giramento di testa, e il peggio che può succedere è che il paziente non ottenga alcun benefcio» dice *Brennan Spiegel*, docente di Medicina dell'Università della California e gastroenterologo del Cedars-Sinai, dove ha lanciato il più vasto programma di sperimentazione con la realtà virtuale. «Nel nostro catalogo abbiamo quindici software con scenari diversi. Pensatela come una farmacia dalla quale i medici possono scegliere la terapia più appropriata per il proprio paziente, proprio come sceglierebbero un film su Netfix». Tra i programmi disponibili ci sono la meditazione sulla spiaggia, un volo in elicottero sopra i ghiacci dell'Islanda, una nuotata con i delfini, un viaggio nel Grand Canyon, una visita al santuario delle foche. E un paio di giochi semplici che consistono in sfamare gattini affamati o centrare il bersaglio in un mondo cartoonesco. Tutti i contenuti sono stati creati, appositamente per uso medico, dalla compagnia *AppliedVr*, una delle tante che negli ultimi due anni sono apparse nella Silicon Valley spinte dalla convinzione che l'applicazione della Vr in ambito sanitario diventerà in futuro molto lucrosa.

«Entro cinque anni sarà offerta ai pazienti come routine» dice Spiegel

Figura 4.7: Il Dott. Brennan osserva un paziente che usa un HMD (2016)

«perché grazie alla sua natura immersiva coinvolge e distrae i soggetti, che non si trovano più in ospedale ma vengono catapultati in un altro mondo. Il cervello viene ingannato e la percezione del dolore cambia». Al Cedars-Sinai sono già un centinaio i pazienti che hanno usato la realtà virtuale in fase postoperatoria, a volte rinunciando ai farmaci. Ora sta partendo uno studio per capire se questa possa addirittura accorciare la degenza e far risparmiare l'ospedale. «La stiamo offrendo a soggetti di tutte le età, e in generale i giovani la accettano meglio. Solo il 10% dice di sentire girare la testa e la rifuta», afferma il Dott. Spiegel[23].

Parte II

La Realtà Virtuale

Capitolo 5

Concetti teorici

Nel seguente capitolo mostrerò i concetti chiave che stanno alla base della realtà virtuale, quali: Immersione e Presenza, inoltre presenterò una lista di linee guida alla progettazione di una "buona"applicazione in VR per rendere l'applicazione con un livello superiore di presenza e priva di effetti collaterali come Motion Sickness o danni fisici.

5.1 Immersione

Parola molto usata fino ad ora, definita come "Il grado oggettivo al quale un sistema in VR progetta gli stimoli sui recettori sensoriali dell'utente, in modo tale che siano quanto più estensivi, congruenti, circondanti, vividi, interattivi, e informanti della trama"[24] Vediamo cosa significa:

- **Estensibilità**, è la quantità di tipi di stimoli sensoriali forniti all'utente, come visivi, uditivi, fisici, ecc..;

- **Congruenza**, è la corrispondenza tra stimoli forniti all'utente e gli stimoli che arrivano al suo cervello (per esempio, l'inclinazione della testa a destra e l'appropriata inclinazione della testa del giocatore nell'applicazione);

- **Circondare**, gli stimoli sono di tipo panoramico, per esempio una larga FOV, un audio stereo o un tracciamento a 360°;
- **Vividezza**, la qualità degli stimoli trasmessi, fanno da esempio una buona risoluzione, la luce, il frame rate, ecc...;
- **Interagibilità**, la capacità dell'utente di poter cambiare il mondo virtuale in cui è immerso, le risposte di entità virtuali ad azioni dell'utente, e l'abilità dell'utente di influenzare eventi futuri;
- **Trama**, la rappresentazione coerente di un messaggio o di un'esperienza, lo svolgersi della sequenza dinamica degli eventi, e il comportamento del mondo e delle sue entità.

L'immersione è solo la tecnologia che ha il potenziale di coinvolgere gli utenti nell'esperienza in VR. Tuttavia, l'immersione è solo una parte dell'esperienza VR che necessita anche di un essere umano per percepire e interpretare gli stimoli artificiali presentati. L'immersione può dirigere la mente, ma non riesce a controllarla. Come l'utente sperimenta soggettivamente l'immersione è conosciuta come *presenza*.

5.2 Presenza e illusione

La presenza è il senso di essere dentro uno spazio, sebbene fisicamente si ci trovi altrove. Siccome la presenza è uno stato psicologico interno dell'utente, una forma di comunicazione viscerale è difficile descriverlo a parole.

La seguente definizione di presenza è stata forgiata da un gruppo di studio, durante la primavera del 2000:

"*Presenza è quello stato psicologico o percezione soggettiva nella quale sebbene parte o tutta la corrente esperienza dell'individuo sia generata e/o filtrata attraverso una tecnologia artificiale, parte o tutta la percezione dell'individuo fallisce ad appurare accuratamente il ruolo che la tecnologia abbia avuto nell'esperienza*"(International Society for Presence Research, 2000).

Da tale definizione si può comprendere come i concetti di immersione e presenza sino legati, la presenza di fatti è in funzione sia dell'utente che dell'immersione. L'immersione è capace di indurre nell'utente il senso di presenza, più elevato è il livello di immersione più alto è la possibilità per il sistema di generare nell'utente la presenza.

Al contrario il *rompersi della presenza*, è il momento in cui l'illusione generata dal mondo virtuale si spezza e l'utente si sveglia e vede dov'è veramente - nel mondo reale indossando un HMD.[25]

La presenza può essere categorizzata in base all'illusione che va a creare:

- **Essere in un posto spazialmente stabile**, chiamato da Slater "Illusione di un posto"[26], è la sensazione dell'utente di essere in un ambiente fisico in cui tutti gli stimoli sensoriali dell'utente sono congruenti con ciò che gli si presenta come feedback;

- **Impersonificazione**, è la percezione dell'utente di avere un proprio corpo all'interno dell'applicazione, molte esperienze di VR non soddisfano questo requisito, ciò non si può dire per l'applicazione che ho realizzato. Diverse ricerche hanno portato alla luce i benefici di "vestire i panni di un altro"come mezzo per insegnare empatia riducendo, per esempio, la discriminazione razziale[27]. In applicazioni che fanno uso di questa tecnica il controllo del movimento del corpo virtuale è fondamentale per non spezzare la presenza e di conseguenza l'illusione creata.

- **Interazione Fisica**, la mera esplorazione visiva non è sufficiente per gli utenti per credere di essere in un mondo diverso, occorrono dei feedback fisici, come: vibrazioni del controller, audio di sottofondo, o tramite specifiche apparecchiature per il feedback tattile come l'innovativa *Hardlight VR Suit* da poco approdata su Kickstarter.com;

- **Comunicazione sociale**, l'illusione di stare veramente comunicando, sia verbalmente che attraverso il linguaggio del corpo, con altri personaggi , siano essi controllati dal computer (NPCs) o da utenti.

5.3 Linee guida alla progettazione in VR

Grazie ai capitoli precedenti ci possiamo essere fatti una buona idea di come un buon sistema in VR debba funzionare, si può perciò riassumere quanto detto in pratiche linee guida da seguire:

- Focalizzarsi sulla User Experience piuttosto che sulla tecnologia usata;
- Semplificare e armonizzare la comunicazione tra utente e tecnologia, ad esempio per mezzo di interfacce diegetiche[28] o controlli semplici;
- Scegliere quale forma di realtà si vuole creare e quale hardware di output o input usare;
- Creare una storia concettualmente forte per immergere l'utente nell'applicazione;
- Massimizzare l'immersione tramite migliorie tecnologiche così da poter abbassare la soglia di inizio della sensazione di presenza dell'utente;
- Minimizzare le rotture della presenza;
- Focalizzarsi prima di tutto sulla stabilità dell'ambiente virtuale, poi concentrarsi sull'interazione fisica e solo infine gestire la comunicazione sociale.

Capitolo 6

Percezione dell'utente

Quello che si percepisce del mondo intorno a se è "tutto nella propria testa". E' necessario comprendere che la luce intorno a noi forma delle immagini sulle nostre retine che catturano i colori, i movimenti, e le relazioni spaziali nel mondo fisico. Per coloro con una vista normale, queste immagini catturate possono apparire con perfetta chiarezza, velocità, precisione, e risoluzione, pur essendo distribuite su un ampio campo visivo. Tuttavia, veniamo ingannati.

Vedremo in questo capitolo che questa apparente perfezione della nostra visione è in gran parte un'illusione perché le strutture neurali si stanno riempiendo di dettagli plausibili per generare un quadro coerente nelle nostre teste che sia coerente con le nostre esperienze di vita e preconcetti. Quando si costruisce la tecnologia VR che coopta questi processi, è importante capire come funzionano. Essi sono stati progettati per fare di più con meno dati, e ingannare questi processi neurali con la VR produce molti effetti collaterali imprevisti perché la tecnologia di visualizzazione non è una replica perfetta del mondo circostante.[29]

6.1 Percezione dello spazio

Normalmente più unità sensoriali spesso cooperano per darci la nostra posizione spaziale. *La vista* certamente è la modalità spaziale superiore in quanto è più precisa, più veloce nel percepire la posizione (anche se l'udito è più veloce quando il compito è di rilevamento della posizione), è accurata per tutte e tre le direzioni(mentre l'udito è ragionevolmente buono solo nelle direzioni laterali alla testa), e lavora bene nelle lunghe distanze.[30]

I **giudizi esocentrici**, sono il senso di dove gli oggetti sono relativamente ad altri oggetti, il mondo, o qualche altro riferimento esterno. I giudizi esocentrici comprendono il senso di gravità, di direzione geografica (per esempio, nord), e la distanza tra due oggetti.

La percezione della posizione degli oggetti comprende anche la direzione e distanza rispetto ai nostri corpi. I **giudizi egocentrici** sono il senso di dove (direzione e distanza) gli oggetti sono rispetto all'osservatore. I giudizi egocentrici possono essere eseguiti dai nostri sensi primari come: vista, udito, propriocezione, e tatto. Diverse variabili influenzano la nostra percezione egocentrica, sicuramente più stimoli sono disponibili, più i nostri giudizi sono stabili.

Per meglio comprendere come percepiamo lo spazio intorno a noi, esso può essere diviso, a seconda della distanza da noi in tre regioni egocentriche[31]:

- **Spazio personale**, è considerato come il volume tra il nostro corpo e la massima estensione delle nostre braccia. Questo spazio raggiunge solitamente i due metri rispetto il punto di vista dell'utente, esso include anche i piedi dell'utente in posizione ferma. Lavorare all'interno di spazio personale offre vantaggi rispetto il sistema propriocettore, si può ottenere un mapping più diretto tra movimento della mano e il movimento oggetto, una più forte visione binoculare e una precisione di movimento della testa e del corpo.

- **Spazio di azione**, consiste nello "spazio pubblico" in cui noi ci possiamo muovere, parlare con altri o lanciare oggetti. Questo spazio va

dai due metri ai venti metri dall'utente.

- **Spazio di vista**, posizionato oltre i venti metri di distanza dall'utente, è dove gli stimoli percettivi sono solo accennati, e sono utilizzati solo con lo scopo di ingannare l'occhio e fargli credere che ci troviamo in uno spazio reale.

La percezione naturale dello spazio all'interno di un ambiente in VR rimane ancora una sfida per la tecnologia di visualizzazione, come anche la creazione di contenuti. Per esempio, mentre la percezione della distanza egocentrica può essere abbastanza buona per il mondo reale, in VR la distanza egocentrica è fin troppo compressa.[32][33] Sebbene la percezione spaziale veritiera non è essenziale per tutte le applicazioni in VR, è estremamente importante in alcune applicazioni come: l'architettura, il campo automobilistico e applicazioni militari o mediche.

6.2 Percezione del tempo

Il tempo può essere espresso come la percezione personale o l'esperienza di cambiamenti successivi. Il **tempo percepito** può essere visto come una costruzione del cervello che può essere manipolato e distorto in alcune circostanze.

La distorsione temporale o **percezione ritardata** avviene quando il tempo dal ricevimento dello stimolo al tempo di reazione passa un lasso di tempo troppo lungo (oltre i 200 ms) sebbene perciò noi viviamo nel passato di un paio di centinaia di millisecondi, non vuol dire che vada bene la stessa cosa in sistemi in VR, questo latenza esterna al corpo aggiuntiva è la maggior causa di *Motion Sickness* che discuteremo nel capitolo successivo[6].

6.3 Percezione del movimento

La percezione non è statica, ma varia continuamente nel tempo. La percezione del movimento in un primo momento può sembrare piuttosto banale.

Tuttavia, la percezione del movimento è un processo complesso che coinvolge un numero elevato di sistemi sensoriali e componenti fisiologiche[34].

Siamo in grado di percepire il movimento anche quando gli stimoli visivi non si muovono sulla retina, come ad esempio durante l'inseguimento di un oggetto in movimento con i nostri occhi. Altre volte invece, gli stimoli si muovono sulla retina ma noi non percepiamo il movimento; per esempio, quando gli occhi si muovono per esaminare oggetti diversi. Nonostante l'immagine del mondo si muove sulle nostre retine, noi percepiamo un mondo stabile.

Un mondo percepito senza movimento può essere sia inquietante che pericoloso. Una persona che soffre di achinetopsia (cecità al movimento) vede sia le persone che gli oggetti apparire o scomparire improvvisamente a causa del fatto di non essere in grado di vederli avvicinare. Tale situazione può essere indotta anche dalla VR nel caso in cui il sistema costruito soffra di pesati latenze, ciò ovviamente può comportare problemi di coerenza al sistema vestibolare dell'utente con conseguenze quali motion sickness, e danni fisici.

6.4 Linee guida allo sviluppo in VR

Definendo in cosa consiste la percezione dell'utente e vari tipi di percezioni, ci possiamo fare un'idea delle principali linee guida da seguire per costruire un sistema in VR, senza o con un ridotto contenuto di effetti collaterali legati alla percezione:

- Pensare in termini di Spazio personale, Spazio di Azione e di vista per modellare gli ambienti e le interazioni;

- Usare diversi tecniche di indicazione della profondità, come l'aggiunta di ombre o tecniche di occlusione per aumentare il riconoscimento dello spazio dell'utente e di conseguenza la presenza;

- Usare testo posizionato nello spazio in modo diegetico [28] invece che un pannello 2D che segue la vista dell'utente;

- Evitare qualsiasi tipo di ritardo nei movimenti o discrepanza temporale per evitare fastidiosi effetti quali motion sickness, o danni fisici;

Capitolo 7

Effetti nocivi della VR

Un problema medico o un effetto nocivo per la salute è un qualsiasi problema causato da un sistema o un'applicazione in realtà virtuale che degrada la salute dell'utente come:

- Nausea;
- Affaticamento oculare;
- Mal di testa;
- Vertigini;
- Lesioni fisiche;
- Malattie trasmissibili.

Le cause di questi effetti includono:

- La percezione di un proprio movimento attraverso l'ambiente o della scena stessa;
- Una calibrazione scorretta dei sistemi di rilevamento della posizione;
- La latenza;
- Ostacoli fisici;

- Una scarsa igiene.

Non si potranno mai eliminare in pieno tutti gli aspetti negativi legati alla VR per tutti gli utenti, ma andando a capire i problemi e perché sorgono, si possono progettare sistemi o applicazioni in realtà virtuali che almeno riducono la loro gravità e la loro durata.

7.1 Motion Sickness

Non esiste una definizione specifica di Motion Sickness, in generale però può essere espressa come: l'effetto collaterale con segni osservabili e documentabili associati all'esposizione ad un movimento reale o apparentemente reale.[35]

E' classificato come uno degli effetti collaterali più comuni legati al mondo della realtà virtuale. I sintomi sono molteplici : malessere generale, nausea, capogiri, mal di testa, disorientamento, vertigini, pallore, sudorazione, sonnolenza e nel peggiore dei casi vomito[36].

Diverse cause possono ricondurci alla motion sickness e degradare di conseguenza l'esperienza dell'utente:

- **Movimento della scena**, è il movimento della visuale dell'intero ambiente che normalmente non accade nel mondo reale [37], ciò può essere causato in modo *intenzionale* da parte dello sviluppatore dell'applicazione o in modo *non intenzionale* a causa di tecnologia scadente o una calibrazione inaccurata dell'apparecchiatura.

- **Movimento di sé**, il proprio movimento all'interno dello spazio virtuale non sempre causa motion sickness, possiamo prendere ad esempio il videogioco Eagle Flight della Ubisoft descritto precedentemente, in cui la scena si muove ad una velocità quanto più costante possibile senza generare scompensi; ma se la scena si muovesse in altri modi sicuramente provocherebbe malessere nell'utente. Quando una scena si

muove in modo non contestuale con il movimento fisico dell'utente avviene un incoerenza fra ciò che vede e ciò che prova, in quanto si genera una accelerazione virtuale che gli organi otoliti dell'utente non provano e perciò l'utente non trova niente con cui confrontare la velocità che sta vedendo, generando sintomi come capogiri o nausea.

Esistono diverse teorie sulla ragione per cui avviene la Motion sickness, vediamone alcune:

- **Teoria del conflitto sensoriale**, è la teoria più largamente accettata, essa dice che la motion sickness sia dovuta ad una tale alterazione dell'ambiente da rendere incompatibili, tra loro e con il nostro modello mentale o aspettative, ogni informazione in arrivo verso gli organi sensoriali principali (vista, udito e sistema vestibolare)[38]. Infatti i conflitti principali che possono verificarsi nei sistemi virtuali sono le incongruenze tra sistema vestibolare e sistema visivo; nella maggior parte delle applicazioni in VR dove si prova motion sickness il sistema visivo percepisce un movimento ma il sistema vestibolare non lo prova.

- **Teoria dell'avvelenamento**, ci offre una ragione per cui il movimento ci fa star male, è fondamentale per la nostra sopravvivenza percepire correttamente il movimento del nostro corpo e il movimento del mondo che ci circonda. Se otteniamo informazioni contrastanti dai nostri sensi, significa che qualcosa non va con i nostri sistemi percettivi e motori. I nostri corpi si sono evoluti in modo tale da proteggerci, riducendo al minimo i disturbi fisiologici prodotti da tossine assorbite. Protezione che avviene attraverso: espulsione del veleno attraverso la sudorazione e vomito, o causando nausea e malessere al fine di scoraggiarci da ingerire tossine simili in futuro. La risposta associata alla motion sickness può verificarsi perché il cervello interpreta le mancate corrispondenze sensoriali come un segno di intossicazione, e innesca nausea o vomito come programma di autodifesa[39].

7.2 Effetti fisici e igiene

L'uso di un'apparecchiatura di realtà virtuale può portare a innumerevoli effetti fisici, tra cui:

- **Gorilla arm**, nel mondo reale noi umani abbiamo una posizione rilassata delle braccia, nelle applicazioni di realtà virtuali dove si necessita l'uso delle braccia, spesso occorre tenerle di fronte a se costringendo l'utente a provare un grande affaticamento delle braccia, chiamato gorilla arm per la sensazione che provoca. Le applicazioni perciò dovrebbero essere progettate per riuscire ad evitare questo fastidioso effetto facendo riposare l'utente tra un'interazione ed un'altra.

- **Traumi e ferite**, con l'uso di HMDs, nel mondo reale si è di fatto cechi, tale condizione può portare l'utente a collidere con oggetti vicini, perdere l'equilibrio e cadere, inciampare in eventuali cavi dell'HMD stesso, sbattere contro il muro, ecc... Questi spiacevoli eventi possono essere evitati configurando una area sicura entro la quale non si può incorrere nei pericoli sopracitati, infatti molti HMDs attualmente sul mercato permettono di definire virtualmente l'area sicura e vederne i confini all'interno dell'applicazione attraverso un reticolato.

- **Igiene**, l'HMD è un dispositivo fisico intimo capace di trasportare agenti patogeni come batteri, virus, o funghi, trasmissibili tra diversi utilizzatori dello stesso HMD; la soluzione a ciò, come consigliato dalle case produttrici, è la pulizia costante dell'HMD e la sostituzione dei cuscinetti che vanno a contatto con la faccia dell'utente.

Capitolo 8

Realtà Aumentata vs. Realtà Virtuale

Dopo aver visto cosa si intende per realtà virtuale, in questo capitolo conclusivo mostrerò un confronto tra l'uso della Realtà virtuale(VR) e la Realtà aumentata (AR), il tutto corredato da esempi e applicazioni note del settore per meglio caratterizzare questi due mondi che sebbene siano abbastanza simili differiscono in alcuni aspetti fondamentali.

8.1 Il confronto

La realtà aumentata viene definita come "*un ambiente artificiale creato attraverso la combinazione del mondo reale e dati generati dal computer*" [Collins dictionary]

La definizione stessa introduce un elemento aggiuntivo rispetto alla definizione di realtà virtuale, l'uso del mondo reale. Infatti, sebbene sia AR che VR facciano uso di HMDs, nella VR i visori sono opachi senza possibilità di vedere il mondo reale circostante ma permettono una immersione totale nel mondo virtuale; mentre l'AR usando HMDs che consentono la visione del mondo esterno, perciò gli utenti possono vedere il mondo reale attraverso i visori e le informazioni aggiuntive che il visore sovrappone alla realtà.

Attualmente esistono diversi HMDs per realtà aumentata, tra i più noti si possono citare: Google Glass, Microsoft HoloLens e Magic Leap; inoltre ultimamente è possibile utilizzare il proprio smartphone, di cui viene sfruttato la fotocamera, per intensificare la visione della realtà dell'utente.

Sebbene la tecnologia, l'elaborazione di immagini in real-time, e la calibrazione tra elementi virtuali e il mondo reale debbano ancora maturare e svilupparsi a pieno; le applicazioni per AR sono innumerevoli e in costante ascesa, tra cui:

- **HoloAnatomy**, è un'applicazione che consente agli utenti di esaminare gli organi di un corpo da ogni punto di vista e interagirci. Si tratta di un passo avanti rispetto dissezione dei tessuti morti e la visualizzazione di illustrazioni 2D sui classici libri di medicina, trasformando l'educazione medica attraverso l'uso di Microsoft HoloLens. Dispositivo di Microsoft che proietta oggetti olografici nel campo visivo dell'utente, mescolando il virtuale con la realtà ed essenzialmente migliorando l'apprendimento del corpo umano.

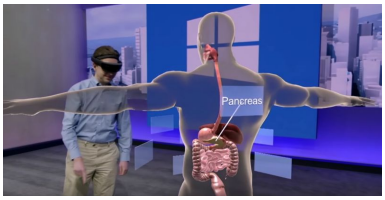

Figura 8.1: HoloAnatomy (2016)

- **ARC4**, è un visore appositamente progettato per le forze militari statunitensi, tale sistema permette ai comandanti di inviare mappe e altre informazioni direttamente nel campo visivo del soldato che può vedere le informazioni dell'ambiente circostante, attraverso l'uso di un HMD (indicazioni, avvertimenti, nemici vicini) senza la necessità di chinare il capo a consultare mappe o computer. Il visore si attacca direttamen-

te all'elmetto militare, e può anche essere integrato con il sistema di controllo delle armi.

Figura 8.2: ARC4 (2014)

- **Pokémon GO**, sviluppato da Niantic per iOS e Android. Il gioco è stato il risultato di una collaborazione tra Niantic e Nintendo. Nel gioco, i giocatori utilizzano le potenzialità del GPS del dispositivo mobile per localizzare, catturare, combattere, e addestrare creature virtuali, chiamati Pokémon, che appaiono sullo schermo come se fossero nella stessa posizione del mondo reale del giocatore, sfruttando l'uso della fotocamera del dispositivo dell'utente per catturare la realtà.

Figura 8.3: Pokémon GO (2016)

Sebbene le tecnologie di VR e AR siano molto simili, differiscono in alcuni aspetti che riassumerò nella seguente tabella:

	Realtà Virtuale	**Realtà Aumentata**
Vantaggi	• Aumenta la velocità di apprendimento; • Immerge totalmente l'utente in una realtà artificiale; • Ampia FOV.	• Aumenta la velocità di apprendimento; • Arricchisce la realtà aumentando la velocità delle informazioni trasmesse all'utente.
Svantaggi	• Motion sickness; • Estraniamento sociale dalla realtà.	• Distrae dalla vera realtà; • Problemi di Privacy legati al riconoscimento facciale e l'uso di fotocamere • FOV limitata

Tabella 8.1: Realtà Virtuale vs. Realtà Aumentata

Parte III

Creazione dell'applicazione Fashion Island VR

Capitolo 9

Costituzione del laboratorio

L'applicazione che ho sviluppato, per la sua natura di applicazione in realtà virtuale, necessita di uno spazio apposito per lo sviluppo ed una macchina dedicata costruita ad-hoc per tale scopo. In questo capitolo descriverò la costituzione del laboratorio di realtà virtuale, le tecnologie e apparecchiature utilizzate all'interno, e le ragioni che hanno mi hanno portato a tali scelte.

9.1 Il Laboratorio e l'hardware

L'installazione del laboratorio è partita prima di tutto nel capire quando spazio occorreva per avere un'area di gioco ragionevolmente grande per permettere il testing delle future applicazioni.
Dalle specifiche tecniche di montaggio di HTC Vive e Oculus si indica un area di gioco in cui ciascuna base-station dista l'una dall'altra al massimo di 3.5 m circa e minimo 1.5m circa. Il Professor Donatiello ha trovato una stanza abbastanza grande per soddisfare tali requisiti in via Malaguti 1D (Bologna), completa di scrivanie, sedie e lavagna.

La scelta del HMD da utilizzare è avvenuta in seguito ad uno studio di settore esposto nel capitolo 3, infine si è optato per l'HTC Vive della Valve Corporation soprattutto perché è completo di controller e ha prestazioni

più elevate rispetto a Oculus Rift di Facebook o Google Daydream. Inoltre sono stati acquistati anche due tripodi per poter posizionare le base-station dell'HTC Vive nella stanza senza bisogno di fissarle al muro tramite tasselli e viti, permettendomi di spostarle facilmente ampliando o contraendo l'area di gioco a seconda della necessità.

La macchina utilizzata per lo sviluppo dell'applicazione l'ho scelta partendo dalle specifiche tecniche che HTC consiglia, esposte nella sezione 3.3.4. Da tali specifiche è facile evincere il fatto che lo sviluppo di applicazioni in VR sia un compito GPU-intensive. Grazie ai discorsi alle Game Developers Conference (GDC) del 2015[16] e del 2016[40] di Alex Vlachos, Senior Graphic Programmer presso Valve, nelle quali specifica che per ogni lente vengono computati 378 milioni di pixel al secondo, e come almeno due GPU che lavorano in accordo siano consigliabili per aumentare quasi del 35% le prestazioni di elaborazione, ho deciso di optare per un computer con le seguenti caratteristiche tecniche:

- **Processore**: Intel Core i7-6850K;
- **Scheda Video**: Doppia scheda NVIDIA GeForce 1080 (SLI enabled);
- **Memoria**: DDR4 da 32 GB a 2133 MHz;
- **Disco Rigido**: Unità SSD principale da 512 GB e Unità di storage SATA da 4 TB e 7200 rpm;
- **Sistema Operativo**: Windows 10 Pro (64 bit).

Nella figura 9.1 si può ammirare il laboratorio completamente assemblato dal sottoscritto e pronto all'uso.

Figura 9.1: Laboratorio VR Unibo (2016)

Capitolo 10

Progettazione dell'applicazione

L'applicazione è nata per esplorare tutti i concetti appresi e presentati nei precedenti capitoli, e riuscire a costruire un ambiente di lavoro generale per lo sviluppo di applicazioni altre applicazioni.

Io sottoscritto, Il Prof. Lorenzo Donatiello e il Dott. Gustavo Marfia abbiamo progettato l'applicazione da sviluppare in seguito a diversi incontri, scambi di idee, e sessioni di brainstorming, alla fine delle quali abbiamo definito le linee generali da rispettare durante lo sviluppo:

1. Il tema principale dell'applicazione è la moda e il settore vestiario;
2. Il cuore dell'applicazione sta nella possibilità dell'utente di avere un proprio corpo virtuale, con la capacità di muovere le proprie mani e se stesso all'interno del mondo virtuale;
3. L'utente può guardare se stesso attraverso uno specchio posto all'interno del mondo virtuale;
4. Tramite opportuni comandi, l'utente, può selezionare e indossare i vestiti che desidera;
5. L'ambiente in cui è immerso l'utente deve essere un luogo pacifico, e rilassante in cui l'utente può dimenticarsi di essere in un mondo virtuale e immergersi completamente nell'applicazione.

Queste indicazioni generali le ho caratterizzate personalmente durante la fase di progettazione all'interno del laboratorio, facendo brainstorming delle idee sulla lavagna a disposizione in laboratorio. Alla fine ho ultimato la progettazione aggiungendo queste specifiche:

1. L'ambiente virtuale potrà essere una foresta, un isola deserta o una stanza di atelier di moda;

2. Il corpo dell'utente dovrà avere la capacità di muoversi contestualmente alla testa dell'utente;

3. le braccia dell'utente dovranno muoversi contestualmente alla posizione dei controller nelle mani dell'utente.

4. I vestiti nel mondo virtuale saranno selezionabili tramite un sistema di puntamento, perciò la loro posizione nel mondo di gioco dovrà essere visibile dall'utente e facilmente raggiungibile, che ha la capacità di ruotare i vestiti o girarci intorno fisicamente;

5. L'utente dovrà avere la possibilità di muoversi, attraverso un sistema di teletrasporto, nel mondo virtuale;

6. Occorrerà una piccola interfaccia grafica con cui l'utente potrà selezionare i vestiti da far comparire nel mondo di gioco.

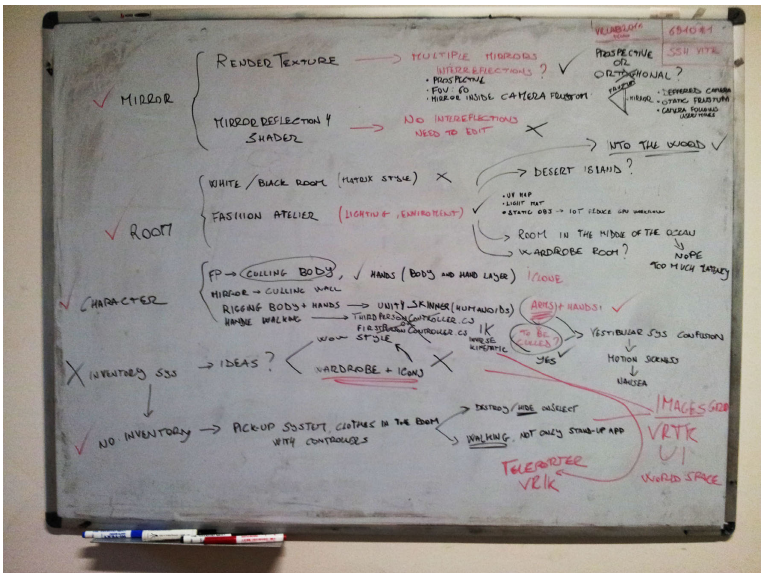

Figura 10.1: Lavagna per il brainstorming (2016)

Capitolo 11

Dettagli Implementativi dell'applicazione

Nel seguente capitolo descriverò ciascuna scelta implementativa dell'applicazione nei dettagli con riferimenti a parti di codice e librerie esterne di sviluppo.

11.1 Il motore di gioco

L'applicazione è stata interamente costruita all'interno di un motore di gioco completo, il quale include un motore fisico, che permette di simulare la fisica degli oggetti, rilevare collisioni, l'uso del suono, degli script e delle animazioni; inoltre include un motore di rendering per la grafica 2D e 3D. Come motore di gioco ho scelto di usare *Unity 5.5.0f3* per le seguenti ragioni:

1. Ho già avuto pregresse esperienze nell'uso di questo Game Engine;

2. E' facile all'uso e ben documentato;

3. Dalla versione 5.1 in poi permette l'integrazione con sistemi di realtà virtuale;

4. Dispone di un *Asset Store* dal quale è possibile scaricare contenuti creati da altri utenti, contenuti che variano dai modelli 3D ad ogni sorta di script;

5. Permette di esportare l'applicazione creata in formati leggibili da sistemi operativi diversi (Windows, Linux, iOS, Playstation, Android, XBox, ecc...);

6. E' compatibile con la libreria di supporto allo sviluppo dell'HTC Vive (*SteamVR*);

11.2 L'ambiente virtuale

Non essendo né un grafico né tanto meno un modellatore 3D ho deciso di affidarmi ai contenuti già presenti sull'asset store per la creazione dell'ambiente di gioco. Ho provato diverse soluzioni, tra cui:

1. *Nature Starter Kit 2*, un'ambientazione che simula una foresta, completa di illuminazione, alberi, erba, sentieri e simulazione del vento. Purtroppo questo ambiente comportava un frame rate troppo elevato (superiore ai 90 FPS), a causa della ricchezza dell'ambiente e l'alta quantità di particolari e di modelli 3D, perciò sono stato costretto a scartarlo.

Figura 11.1: Nature Starter Kit 2 (2015)

2. *Dynamic Soft Shadows Based on Local Cubemaps*, questa seconda soluzione ha attirato la mia attenzione per l'attenzione alle ombre, l'illu-

minazione e la cura nei dettagli della stanza, purtroppo sono stato costretto a scartarla come soluzione perché una illuminazione così curata comportava un innalzamento eccessivo del frame rate.

Figura 11.2: Dynamic Soft Shadows Based on Local Cubemaps (2015)

3. *Island Assets*, una piacevole ambientazione che simula un isola, completa di mare circostante, palme, pontile e una cassa del tesoro. Ho infine deciso di utilizzare questa ambientazione perché grazie al fatto che le geometrie di questo scenario hanno pochi poligoni e non sono eccessive permettono di non degradare l'esperienza di gioco mantenendo un frame rate di 90 FPS, inoltre l'isola deserta offre all'utente uno scenario pacifico e rilassante.

Figura 11.3: Island Assets (2015)

Per dare un tocco personale all'ambiente ho aggiunto alcuni particolari:

- Uno scheletro, in low-poly, che cerca di raggiungere il forziere;
- Una pala e un piccone accanto allo scrigno;

- Un'ascia, conficcata in una palma;
- Una piccola barca a remi sul fondo del pontile, immaginandomi che l'utente sia arrivato sull'isola per mezzo di essa;
- Un grande pannello in legno con lo specchio che descriveremo in seguito.

Figura 11.4: Isola personalizzata (2016)

11.3 Lo specchio

Per l'implementazione dello specchio ho provato due tecniche diverse:

- **Render Texture**, sono delle speciali texture create e aggiornate a tempo di esecuzione. Per utilizzarle, per prima cosa occorre creare una struttura Render Texture da posizionare sopra un pannello nella scena, dopo di che, bisogna definire una Camera che renderà ciò riprende direttamente sulla Render Texture come se questa fosse uno schermo di un televisore. Tale soluzione l'ho scartata in quanto la Camera che catturava l'immagine dell'utente era fissa nell'ambiente non permettendo di creare un effetto specchio veritiero, inoltre le prestazione erano molto degradate a causa del continuo aggiornamento della texture che doveva essere necessariamente ad alta definizione;
- **MirrorReflection4**, questo shader corredato da script è reso disponibile da Unity, il problema è che non è stato progettato per la realtà virtuale, dove il riflesso deve essere visualizzato su entrambi gli occhi in

maniera stereoscopica; grazie al forum Unity, molto attivo per quanto riguardo questo argomento, ho recuperato lo script per ottenere la riflessione stereo dello specchio, si può consultare il codice utilizzato alla sezione dell'appendice A.1 e lo shader alla sezione A.2.

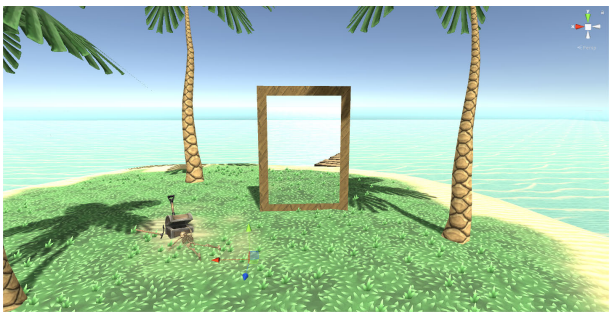

Figura 11.5: Lo specchio sull'isola (2016)

11.4 L'avatar

Valve Corporation, produttore di HTC Vive, ha messo a disposizione sull'asset store di Unity un'importante libreria chiamata *SteamVR* la quale contiene numerose funzionalità da cui imparare e con cui si può partire per costruire un rudimentale sistema in realtà virtuale, personalmente ho appreso molto da questo pacchetto, di cui ho utilizzato: **[CameraRig]**, l'unità di base in cui sono racchiuse le funzionalità di riconoscimento automatico dei controller e HMD, inoltre permette di impostare la grandezza dell'area di gioco entro la quale l'utente è confinato. All'avvio del gioco, grazie a questo "Prefab" vengono caricati tutti gli elementi sopracitati e si è immersi nel mondo di gioco con l'abilità di vedere e muovere i controller di fronte a sé.

A questo punto mancava l'avatar stesso, che ho generato e esportato dal software per creazione di personaggi in 3D *iClone6 Character Creator*, completo di un set base di vestiti ed uno scheletro già impostato. Ho poi impostato il modello 3D del personaggio come figlio del prefab [CameraRig], in modo tale che se viene spostato l'uno si sposta anche l'altro, ho regolato

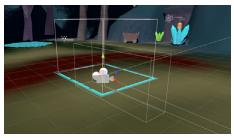

Figura 11.6: CameraRig prefab (2016)

Figura 11.7: iClone6 Character Creator (2016)

l'altezza del personaggio così che non esca dai bordi dell'area di gioco ed infine ho disabilitato tutti i vestiti così che il personaggio alla partenza del gioco ne sia privo.

Ovviamente non rimaneva altro che impostare il movimento del personaggio contestualmente al movimento dell'HMD e dei controller. A tale scopo ho importato la libreria di Cinematica Inversa *FinalIK 1.5*, essa permette, selezionando correttamente i riferimenti alla testa e alle mani dello scheletro del personaggio 3D, di far piegare e ruotare le braccia in relazione alla posizione dei controller, e di ruotare o piegare le spalle, il bacino e le gambe in relazione alla posizione dell'HMD. Dopodiché ho impostato le posizioni delle mani dell'utente e l'altezza della testa così che potesse vedere il proprio corpo dalla sua prospettiva e le mani fossero in una posizione quanto più naturale possibile.

Infine ho recuperato una copia dei modelli dei vestiti dell'utente, e li ho posizionati attorno a lui nel mondo di gioco, come sospesi a mezz'aria, ad una altezza consona per la visualizzazione e la selezione di ciascun capo. Ad essi è stato necessario applicare un *Toon Shader* così che le cavità dei vestiti venissero visualizzate correttamente senza discontinuità nelle texture.

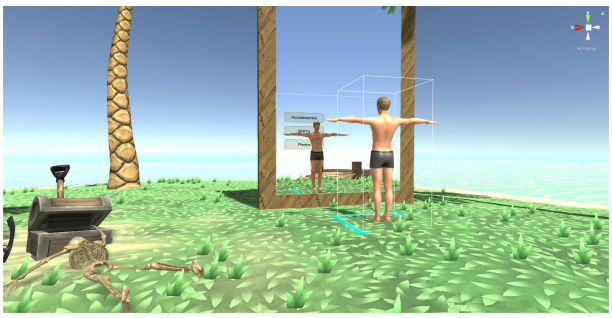

Figura 11.8: Personaggio all'interno della zona di gioco (2016)

Figura 11.9: Personaggio all'interno della zona di gioco con vestiti (2016)

11.5 Interfaccia grafica

Ho deciso di far comparire o scomparire i vestiti per mezzo dell'uso di una semplice Interfaccia Grafica (UI) composta da tre bottoni, che interagiscono con: il set di accessori, il set di camice e il set di pantaloni. I tre bottoni li ho posti di fronte allo specchio così che siano sempre visibili dall'utente e sono integrati in modo diegetico con l'ambiente, assomigliando più ad un oggetto di gioco che ad una UI.

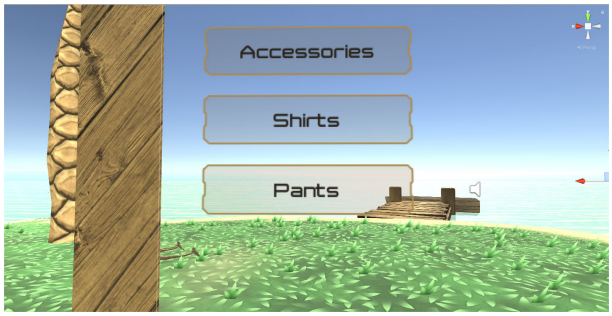

Figura 11.10: Interfaccia grafica diegetica (2016)

Un elemento di interfaccia grafica aggiuntivo l'ho posto sulle mani virtuali dell'utente stesso. All'inizio della sessione di gioco un utente non sa come interagire con l'ambiente e quali comandi usare, per questo motivo ho aggiunto i modelli dei controller del Vive corredati da istruzioni semplici contenute in label diegetiche che indicano i comandi di gioco. Tali istruzioni e modelli possono essere abilitati e disabilitati dell'utente quando vuole premendo il bottone applicazione sul controller sinistro(figure 11.11,11.12).

11.6 Interazioni

L'utente può interagire con il mondo di gioco attraverso i controller che tiene nelle mani. A ciascun bottone dei controller corrisponde un'azione specifica, in particolare con il controller destro è possibile:

- **Bottone Applicazione**, far scomparire i vestiti intorno dall'utente e rendere visibile o invisibile la UI sopra lo specchio (codice alla sezione A.4);

- **Bottone Trackpad**, permette di far apparire un raggio blu che esce dal dito indice della mano destra dell'utente con il quale è possibile indicare i vestiti o i bottoni della UI. In particolare se con il puntatore si passa sopra ad un bottone esso si anima ed emette un suono, se invece si passa sopra un vestito esso rotea grazie allo script nella sezione A.6;

- **Bottone Trigger**, se si è spinto il bottone al punto precedente, e si è sopra uno dei tre bottoni, spingendo il bottone trigger è possibile far apparire o scomparire il set di vestiti corrispondenti (camicie, pantaloni o accessori). Nel caso in cui il puntatore sia su un vestito e il giocatore spinge il tasto trigger allora quel vestito viene indossato o rimosso dall'utente a seconda che esso lo indossi già o meno. Il tutto è controllato dallo script presente alla sezione A.5 che utilizza un sistema di raycast sugli oggetti che vengono discriminati a seconda del "*Layout*" di appartenenza.

Figura 11.11: Istruzioni diegetiche mano destra (2016)

Mentre con il controller sinistro è possibile:

- **Bottone Applicazione**, far apparire e scomparire i controller con le istruzioni che si trovano vicino alle mani dell'utente;

- **Bottone Trackpad**, permette di far apparire una curva tratteggiata che collide contro il terreno disegnando un bersaglio in cui l'utente può teletrasportarsi se in contemporanea spinge anche il **Bottone Trigger**.

Puntamento e teletrasporto sono elementi che ho sfruttato dalla libreria gratuita sull'asset store *VRToolkit*, permettendomi di risparmiare molto tempo di sviluppo.

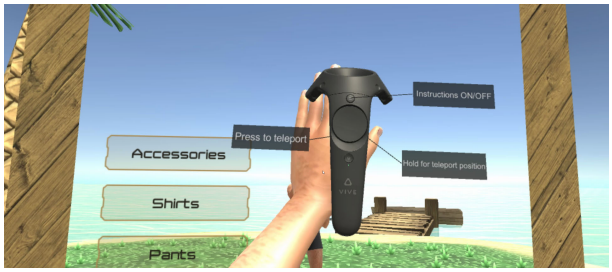

Figura 11.12: Istruzioni diegetiche mano sinistra (2016)

11.7 Suoni e musica

Per garantire una esperienza completamente immersiva per l'utente ho aggiunto al di gioco i seguenti suoni scaricati dal sito web *www.freesound.org*, privi di alcun tipo di royalty:

- Un suono di sottofondo riprodotto in loop, che rievoca il suono delle onde sulla spiaggia e il richiamo dei gabbiani;

- Un feedback sonoro per indicare un'azione compiuta con successo, per esempio: il cambio di abito, la pressione dei bottoni della UI o sul controller. Il tutto controllato per mezzo dello script alla sezione A.5.

Grazie alle potenzialità di Unity ho impostato i suoni in 3D così che l'utente possa discriminare la sorgente di ciascun suono e di conseguenza aumentare l'immersione nell'ambiente.

Conclusioni

Grazie all'applicazione sviluppata, sono riuscito ad apprendere in pieno i concetti presentati nella prima e seconda parte della tesi, provandole sulla mia stessa pelle. Durante lo sviluppo e il testing infatti non sono mancati momenti in cui si sono provati gli effetti di immersione, presenza e tal volta a mio malgrado anche di Motion Sickness. Dopo tutto l'applicazione risultante è piacevole all'uso e immerge pienamente nell'ambiente permettendo all'utente di personificarsi con l'avatar in gioco, e concedendogli un ampia libertà di azione e movimento grazie alle potenzialità dell'HTC Vive, dei sui controller e dell'ampiezza dell'area di gioco.

L'applicazione è stata costruita non solo per dimostrare concetti teorici, ma anche per offrire un ambiente di sviluppo estensibile, facilmente modificabile e replicabile da altri sviluppatori, si può vedere tale sistema come un primo gradino verso lo sviluppo di applicazioni in realtà virtuale con complessità e livello di dettaglio sempre più elevato, amplificando i concetti di presenza, illusione e immersione descritti nella seconda parte della tesi.

Infine, di tale applicazione ho girato un video in cui nel riquadro in alto a destra mi si può vedere usare l'applicazione mentre nella schermata principale si possono vedere gli effetti dei miei movimenti sull'avatar in gioco (figura 11.13).

Figura 11.13: Video della applicazione Fashion Island VR, https://youtu.be/hL32VUfJpU8

Nonostante gli obiettivi che mi sono prefissato di raggiungere siano stati tutti soddisfatti, restano alcuni punti in cui l'applicazione può essere migliorata:

- Ampliando il set di vestiti, e organizzandoli in un sistema di inventario a scomparsa tramite comandi da controller;
- Dando la possibilità agli utenti di poter personalizzare il proprio avatar scegliendo il sesso e ogni caratteristica fisica del modello 3D così che l'utente si personifichi di più nell'avatar aumentando il livello di presenza;
- Rendendo l'applicazione utilizzabile in rete con altri utenti per permettere di aggiungere l'elemento social che a questa applicazione manca;
- Poter personalizzare l'ambientazione a seconda dai gusti dell'utente;
- In un non così lontano futuro, utilizzare il dispositivo per rendere Wireless l'HTC Vive, permettendo un grado superiore di immersione;
- Utilizzando i Vive Tracker, che saranno disponibili a metà 2017, sarà possibile tracciare il movimento dei piedi dell'utente e di conseguenza, tramite l'uso del sistema di Cinematica inversa utilizzato in questa applicazione, dare la possibilità all'utente di poter sollevare i piedi e le gambe così che possa guardarsi gli eventuali pantaloni che ha indossato o semplicemente camminare nell'area di gioco con dei movimenti molto più naturali e fluidi.

Appendice A

Codici

A.1 MirrorReflection.cs

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

[ExecuteInEditMode] // Make mirror live-update even when not in play mode
public class MirrorReflection : MonoBehaviour
{
    public bool m_DisablePixelLights = true;

    // If true, then a single RenderTexture will be used for ALL reflected
        cameras, with the size of that
    // texture defined by the m_SharedTextureSize field.  If false, then
        each reflected camera will have
    // it's own RenderTexture that is sized to match that camera.
    public bool m_UseSharedRenderTexture = false;
    public int m_SharedTextureSize = 256;

    public LayerMask m_ReflectLayers = -1;

    private class ReflectionData
    {
        public Camera camera;
        public RenderTexture left;
        public RenderTexture right;
        public MaterialPropertyBlock propertyBlock;
    }

    private Dictionary<Camera, ReflectionData> m_ReflectionCameras = new
        Dictionary<Camera, ReflectionData>();
```

```
private RenderTexture m_SharedReflectionTextureLeft = null;
private RenderTexture m_SharedReflectionTextureRight = null;

private static bool s_InsideRendering = false;

private static int s_LeftTexturePropertyID;
private static int s_RightTexturePropertyID;

private void Awake()
{
    s_LeftTexturePropertyID = Shader.PropertyToID("_LeftReflectionTex");
    s_RightTexturePropertyID =
        Shader.PropertyToID("_RightReflectionTex");
}

// This is called when it's known that the object will be rendered by
    some
// camera. We render reflections and do other updates here.
// Because the script executes in edit mode, reflections for the scene
    view
// camera will just work!
public void OnWillRenderObject()
{
    var rend = GetComponent<Renderer>();
    if (!enabled || !rend || !rend.enabled)
        return;

    Camera cam = Camera.current;
    if (!cam)
        return;

    // Safeguard from recursive reflections.
    if (s_InsideRendering)
        return;
    s_InsideRendering = true;

    ReflectionData reflectionData;
    CreateMirrorObjects(cam, out reflectionData);

    // Optionally disable pixel lights for reflection
    int oldPixelLightCount = QualitySettings.pixelLightCount;
    if (m_DisablePixelLights)
        QualitySettings.pixelLightCount = 0;

    // Invert culling as the reflection of the mirrors will reverse the
        winding order for everything rendered
```

```
bool oldInvertCulling = GL.invertCulling;
GL.invertCulling = !oldInvertCulling;

UpdateCameraModes(cam, reflectionData.camera);

if (cam.stereoEnabled)
{
    if (cam.stereoTargetEye == StereoTargetEyeMask.Both ||
        cam.stereoTargetEye == StereoTargetEyeMask.Left)
    {
        Vector3 eyePos =
            cam.transform.TransformPoint(SteamVR.instance.eyes[0].pos);
        Quaternion eyeRot = cam.transform.rotation *
            SteamVR.instance.eyes[0].rot;
        Matrix4x4 projectionMatrix =
            GetSteamVRProjectionMatrix(cam,
            Valve.VR.EVREye.Eye_Left);
        RenderTexture target = m_UseSharedRenderTexture ?
            m_SharedReflectionTextureLeft : reflectionData.left;
        reflectionData.propertyBlock.SetTexture(s_LeftTexturePropertyID,
            target);

        RenderMirror(reflectionData.camera, target, eyePos, eyeRot,
            projectionMatrix);
    }

    if (cam.stereoTargetEye == StereoTargetEyeMask.Both ||
        cam.stereoTargetEye == StereoTargetEyeMask.Right)
    {
        Vector3 eyePos =
            cam.transform.TransformPoint(SteamVR.instance.eyes[1].pos);
        Quaternion eyeRot = cam.transform.rotation *
            SteamVR.instance.eyes[1].rot;
        Matrix4x4 projectionMatrix =
            GetSteamVRProjectionMatrix(cam,
            Valve.VR.EVREye.Eye_Right);
        RenderTexture target = m_UseSharedRenderTexture ?
            m_SharedReflectionTextureRight : reflectionData.right;
        reflectionData.propertyBlock.SetTexture(s_RightTexturePropertyID,
            target);

        RenderMirror(reflectionData.camera, target, eyePos, eyeRot,
            projectionMatrix);
    }
}
else
{
```

```
            RenderTexture target = m_UseSharedRenderTexture ?
                m_SharedReflectionTextureLeft : reflectionData.left;
            reflectionData.propertyBlock.SetTexture(s_LeftTexturePropertyID,
                target);
            RenderMirror(reflectionData.camera, target,
                cam.transform.position, cam.transform.rotation,
                cam.projectionMatrix);
        }

        // Apply the property block containing the texture references to
            the renderer
        rend.SetPropertyBlock(reflectionData.propertyBlock);

        // Restore the original culling
        GL.invertCulling = oldInvertCulling;

        // Restore pixel light count
        if (m_DisablePixelLights)
            QualitySettings.pixelLightCount = oldPixelLightCount;

        s_InsideRendering = false;
    }

    void RenderMirror(Camera reflectionCamera, RenderTexture targetTexture,
        Vector3 camPosition, Quaternion camRotation, Matrix4x4
        camProjectionMatrix)
    {
        // Copy camera position/rotation/projection data into the
            reflectionCamera
        reflectionCamera.ResetWorldToCameraMatrix();
        reflectionCamera.transform.position = camPosition;
        reflectionCamera.transform.rotation = camRotation;
        reflectionCamera.projectionMatrix = camProjectionMatrix;
        reflectionCamera.targetTexture = targetTexture;

        reflectionCamera.cullingMask = ~(1 << 4) & m_ReflectLayers.value;
            // never render water layer

        // find out the reflection plane: position and normal in world space
        Vector3 pos = transform.position;
        Vector3 normal = -transform.up;

        // Reflect camera around reflection plane
        Vector4 worldSpaceClipPlane = Plane(pos, normal);
        reflectionCamera.worldToCameraMatrix *=
            CalculateReflectionMatrix(worldSpaceClipPlane);
```

```
        // Setup oblique projection matrix so that near plane is our
            reflection
        // plane. This way we clip everything behind it for free.
        Vector4 cameraSpaceClipPlane = CameraSpacePlane(reflectionCamera,
            pos, normal);
        reflectionCamera.projectionMatrix =
            reflectionCamera.CalculateObliqueMatrix(cameraSpaceClipPlane);

        // Set camera position and rotation (even though it will be ignored
            by the render pass because we
        // have explicitly set the worldToCameraMatrix). We do this because
            some render effects may rely
        // on the position/rotation of the camera.
        reflectionCamera.transform.position =
            reflectionCamera.cameraToWorldMatrix.GetPosition();
        reflectionCamera.transform.rotation =
            reflectionCamera.cameraToWorldMatrix.GetRotation();

        reflectionCamera.Render();
    }

    // Cleanup all the objects we possibly have created
    void OnDisable()
    {
        if (m_SharedReflectionTextureLeft)
        {
            DestroyImmediate(m_SharedReflectionTextureLeft);
            m_SharedReflectionTextureLeft = null;
        }

        if (m_SharedReflectionTextureRight)
        {
            DestroyImmediate(m_SharedReflectionTextureRight);
            m_SharedReflectionTextureRight = null;
        }

        foreach (ReflectionData reflectionData in
            m_ReflectionCameras.Values)
        {
            DestroyImmediate(reflectionData.camera.gameObject);
            DestroyImmediate(reflectionData.left);
            if (reflectionData.right)
            {
                DestroyImmediate(reflectionData.right);
            }
        }
        m_ReflectionCameras.Clear();
```

```
}


private void UpdateCameraModes(Camera src, Camera dest)
{
    if (dest == null)
        return;
    // set camera to clear the same way as current camera
    dest.clearFlags = src.clearFlags;
    dest.backgroundColor = src.backgroundColor;
    if (src.clearFlags == CameraClearFlags.Skybox)
    {
        Skybox sky = src.GetComponent(typeof(Skybox)) as Skybox;
        Skybox mysky = dest.GetComponent(typeof(Skybox)) as Skybox;
        if (!sky || !sky.material)
        {
            mysky.enabled = false;
        }
        else
        {
            mysky.enabled = true;
            mysky.material = sky.material;
        }
    }
    // update other values to match current camera.
    // even if we are supplying custom camera&projection matrices,
    // some of values are used elsewhere (e.g. skybox uses far plane)
    dest.farClipPlane = src.farClipPlane;
    dest.nearClipPlane = src.nearClipPlane;
    dest.orthographic = src.orthographic;
    dest.fieldOfView = src.fieldOfView;
    dest.aspect = src.aspect;
    dest.orthographicSize = src.orthographicSize;
}

// On-demand create any objects we need
private void CreateMirrorObjects(Camera currentCamera, out
    ReflectionData reflectionData)
{
    if (!m_ReflectionCameras.TryGetValue(currentCamera, out
        reflectionData))
    {
        reflectionData = new ReflectionData();
        reflectionData.propertyBlock = new MaterialPropertyBlock();
        m_ReflectionCameras[currentCamera] = reflectionData;
    }
```

```
// Camera for reflection
if (!reflectionData.camera)
{
    GameObject go = new GameObject("Mirror Refl Camera id" +
        GetInstanceID() + " for " + currentCamera.GetInstanceID(),
        typeof(Camera), typeof(Skybox), typeof(FlareLayer));
    reflectionData.camera = go.GetComponent<Camera>();
    reflectionData.camera.enabled = false;
    go.hideFlags = HideFlags.HideAndDontSave;
}

if (m_UseSharedRenderTexture)
{
    // Get rid of any per camera reflection textures
    if (reflectionData.left)
    {
        DestroyImmediate(reflectionData.left);
        reflectionData.left = null;
    }
    if (reflectionData.right)
    {
        DestroyImmediate(reflectionData.right);
        reflectionData.right = null;
    }

    // Create shared reflection textures of the desired size
    if (!m_SharedReflectionTextureLeft ||
        m_SharedReflectionTextureLeft.width != m_SharedTextureSize)
    {
        if (m_SharedReflectionTextureLeft)
            DestroyImmediate(m_SharedReflectionTextureLeft);
        m_SharedReflectionTextureLeft = new
            RenderTexture(m_SharedTextureSize, m_SharedTextureSize,
            24);
        m_SharedReflectionTextureLeft.name =
            "__MirrorReflectionLeft" + GetInstanceID();
        m_SharedReflectionTextureLeft.hideFlags =
            HideFlags.DontSave;
    }

    if (!m_SharedReflectionTextureRight ||
        m_SharedReflectionTextureRight.width != m_SharedTextureSize)
    {
        if (m_SharedReflectionTextureRight)
            DestroyImmediate(m_SharedReflectionTextureRight);
        m_SharedReflectionTextureRight = new
            RenderTexture(m_SharedTextureSize, m_SharedTextureSize,
```

```
                24);
            m_SharedReflectionTextureRight.name =
                "__MirrorReflectionRight" + GetInstanceID();
            m_SharedReflectionTextureRight.hideFlags =
                HideFlags.DontSave;
        }
    }
    else
    {
        // Get rid of any the shared reflection textures
        if (m_SharedReflectionTextureLeft)
        {
            DestroyImmediate(m_SharedReflectionTextureLeft);
            m_SharedReflectionTextureLeft = null;
        }

        if (m_SharedReflectionTextureRight)
        {
            DestroyImmediate(m_SharedReflectionTextureRight);
            m_SharedReflectionTextureRight = null;
        }

        // Reflection render texture
        if (!reflectionData.left || reflectionData.left.width !=
            currentCamera.pixelWidth || reflectionData.left.height !=
            currentCamera.pixelHeight)
        {
            if (reflectionData.left)
                DestroyImmediate(reflectionData.left);
            reflectionData.left = new
                RenderTexture(currentCamera.pixelWidth,
                currentCamera.pixelHeight, 24);
            reflectionData.left.name = "__MirrorReflectionLeft" +
                GetInstanceID() + " for " +
                currentCamera.GetInstanceID();
            reflectionData.left.hideFlags = HideFlags.DontSave;
            reflectionData.propertyBlock.SetTexture(s_LeftTexturePropertyID,
                reflectionData.left);
        }

        // If stereo is enabled, create right reflection texture
        if (currentCamera.stereoEnabled)
        {
            if (!reflectionData.right || reflectionData.right.width !=
                currentCamera.pixelWidth || reflectionData.right.height
                != currentCamera.pixelHeight)
            {
```

```
                if (reflectionData.right)
                    DestroyImmediate(reflectionData.right);
                reflectionData.right = new
                    RenderTexture(currentCamera.pixelWidth,
                    currentCamera.pixelHeight, 24);
                reflectionData.right.name = "__MirrorReflectionRight" +
                    GetInstanceID() + " for " +
                    currentCamera.GetInstanceID();
                reflectionData.right.hideFlags = HideFlags.DontSave;
                reflectionData.propertyBlock.SetTexture(s_RightTexturePropertyID,
                    reflectionData.right);
            }
        }
    }
}

// Given position/normal of the plane, calculate plane in world space.
private Vector4 Plane(Vector3 pos, Vector3 normal)
{
    return new Vector4(normal.x, normal.y, normal.z, -Vector3.Dot(pos,
        normal));
}

// Given position/normal of the plane, calculates plane in camera space.
private Vector4 CameraSpacePlane(Camera cam, Vector3 pos, Vector3
    normal)
{
    Matrix4x4 m = cam.worldToCameraMatrix;
    Vector3 cpos = m.MultiplyPoint(pos);
    Vector3 cnormal = m.MultiplyVector(normal).normalized;
    return Plane(cpos, cnormal);
}

// Calculates reflection matrix around the given plane
private static Matrix4x4 CalculateReflectionMatrix(Vector4 plane)
{
    Matrix4x4 reflectionMat = Matrix4x4.identity;

    reflectionMat.m00 = (1F - 2F * plane[0] * plane[0]);
    reflectionMat.m01 = (   - 2F * plane[0] * plane[1]);
    reflectionMat.m02 = (   - 2F * plane[0] * plane[2]);
    reflectionMat.m03 = (   - 2F * plane[3] * plane[0]);

    reflectionMat.m10 = (   - 2F * plane[1] * plane[0]);
    reflectionMat.m11 = (1F - 2F * plane[1] * plane[1]);
    reflectionMat.m12 = (   - 2F * plane[1] * plane[2]);
    reflectionMat.m13 = (   - 2F * plane[3] * plane[1]);
```

```
        reflectionMat.m20 = (    - 2F * plane[2] * plane[0]);
        reflectionMat.m21 = (    - 2F * plane[2] * plane[1]);
        reflectionMat.m22 = (1F - 2F * plane[2] * plane[2]);
        reflectionMat.m23 = (    - 2F * plane[3] * plane[2]);

        reflectionMat.m30 = 0F;
        reflectionMat.m31 = 0F;
        reflectionMat.m32 = 0F;
        reflectionMat.m33 = 1F;

        return reflectionMat;
    }

    public static Matrix4x4 GetSteamVRProjectionMatrix(Camera cam,
        Valve.VR.EVREye eye)
    {
        Valve.VR.HmdMatrix44_t proj =
            SteamVR.instance.hmd.GetProjectionMatrix(eye,
            cam.nearClipPlane, cam.farClipPlane,
            SteamVR.instance.graphicsAPI);
        Matrix4x4 m = new Matrix4x4();
        m.m00 = proj.m0;
        m.m01 = proj.m1;
        m.m02 = proj.m2;
        m.m03 = proj.m3;
        m.m10 = proj.m4;
        m.m11 = proj.m5;
        m.m12 = proj.m6;
        m.m13 = proj.m7;
        m.m20 = proj.m8;
        m.m21 = proj.m9;
        m.m22 = proj.m10;
        m.m23 = proj.m11;
        m.m30 = proj.m12;
        m.m31 = proj.m13;
        m.m32 = proj.m14;
        m.m33 = proj.m15;
        return m;
    }
}
```

A.2 MirrorReflection.shader

```
// Upgrade NOTE: replaced 'UNITY_INSTANCE_ID' with
    'UNITY_VERTEX_INPUT_INSTANCE_ID'
```

```
Shader "FX/MirrorReflection"
{
        Properties
        {
                _MainTex("Base (RGB)", 2D) = "white" {}
                [HideInInspector] _LeftReflectionTex("", 2D) = "white" {}
                [HideInInspector] _RightReflectionTex("", 2D) = "white" {}
        }
        SubShader
        {
                Tags{ "RenderType" = "Opaque" }
                LOD 100

                Pass{
                        CGPROGRAM
#pragma vertex vert
#pragma fragment frag
#include "UnityCG.cginc"

                        float4 _MainTex_ST;
                        sampler2D _MainTex;
                        sampler2D _LeftReflectionTex;
                        sampler2D _RightReflectionTex;

                        struct VertexInput
                        {
                                float4 pos : POSITION;
                                float2 uv: TEXCOORD0;
                                UNITY_VERTEX_INPUT_INSTANCE_ID
                        };

                        struct v2f
                        {
                                float2 uv : TEXCOORD0;
                                float4 refl : TEXCOORD1;
                                float4 pos : SV_POSITION;
                                UNITY_VERTEX_INPUT_INSTANCE_ID
                                UNITY_VERTEX_OUTPUT_STEREO
                        };

                        // Same as standard ComputeScreenPos() except that
                            it doesn't call TransformStereoScreenSpaceTex()
                        // when stereo instance rendering is enabled. This
                            is important because we need to be able to
                            sample
                        // from the entire reflection texture, and not just
                            the left/right half, which is what the normal
```

```
                        // ComputeScreenPos() would get us.
                        inline float4 ComputeScreenPosIgnoreStereo(float4
                            pos) {
                            float4 o = pos * 0.5f;
#if defined(UNITY_HALF_TEXEL_OFFSET)
                            o.xy = float2(o.x, o.y*_ProjectionParams.x)
                                + o.w * _ScreenParams.zw;
#else
                            o.xy = float2(o.x, o.y*_ProjectionParams.x)
                                + o.w;
#endif
                            o.zw = pos.zw;
                            return o;
                        }

                        v2f vert(VertexInput v)
                        {
                            UNITY_SETUP_INSTANCE_ID(v);
                            v2f o;
                            UNITY_INITIALIZE_OUTPUT(v2f, o);
                            UNITY_TRANSFER_INSTANCE_ID(v, o);
                            UNITY_INITIALIZE_VERTEX_OUTPUT_STEREO(o);

                            o.pos = mul(UNITY_MATRIX_MVP, v.pos);
                            o.uv = TRANSFORM_TEX(v.uv, _MainTex);
                            o.refl =
                                ComputeScreenPosIgnoreStereo(o.pos);
                            return o;
                        }

                        fixed4 frag(v2f i) : SV_Target
                        {
                            UNITY_SETUP_INSTANCE_ID(i)
                            fixed4 tex = tex2D(_MainTex, i.uv);
                            fixed4 refl;

                            bool leftEye;

#ifdef UNITY_SINGLE_PASS_STEREO
                            leftEye = unity_StereoEyeIndex == 0;
#else
                            leftEye = (unity_CameraProjection[0][2] <=
                                0);
#endif
                            if (leftEye)
                            {
                                refl =
```

```
                                        tex2Dproj(_LeftReflectionTex,
                                        UNITY_PROJ_COORD(i.refl));
                                }
                                else
                                {
                                        refl =
                                        tex2Dproj(_RightReflectionTex,
                                        UNITY_PROJ_COORD(i.refl));
                                }
                                return tex * refl;
                        }
                        ENDCG
                }
        }
}
```

A.3 GameController.cs

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



public class GameController : MonoBehaviour
{

        public void ToggleClothes(GameObject clothesToToggle){
                clothesToToggle.SetActive (!clothesToToggle.activeSelf);
        }


}
```

A.4 HandController.cs

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

public class HandController : MonoBehaviour {

        [Tooltip ("Ingame Canvas.")]
        public GameObject canvas;
```

```
[Tooltip ("Ingame Clothes.")]
public GameObject pants,acc,shirts;

[Tooltip ("Canvas Audio when appear.")]
public AudioSource soundToggleCanvas;

[Tooltip ("Tracker model.")]
public GameObject[] models;

[Tooltip ("Tooltip canvas.")]
public GameObject[] tooltips;

[Tooltip ("Is the right controller canvas.")]
public bool isRight;


private Valve.VR.EVRButtonId menuButton =
    Valve.VR.EVRButtonId.k_EButton_ApplicationMenu;
private Valve.VR.EVRButtonId gripButton =
    Valve.VR.EVRButtonId.k_EButton_Grip;
private Valve.VR.EVRButtonId triggerButton =
    Valve.VR.EVRButtonId.k_EButton_SteamVR_Trigger;
private SteamVR_Controller.Device device { get { return
    SteamVR_Controller.Input((int)hand.index); } }

private SteamVR_TrackedObject hand;
private GameObject objectPickedUp;


// Use this for initialization
void Start () {
        hand = GetComponent <SteamVR_TrackedObject> ();
}

// Update is called once per frame
void Update () {
        if (device == null){
                Debug.Log ("Controller not initialized");
                return;
        }


        if(device.GetPressDown(menuButton)){
                soundToggleCanvas.Play ();
                if (isRight) {
                        canvas.SetActive (!canvas.activeSelf);
```

```
                        pants.SetActive (false);
                        acc.SetActive (false);
                        shirts.SetActive (false);
                    } else {
                        foreach(GameObject model in models){
                            model.SetActive (!model.activeSelf);
                        }
                        foreach(GameObject tooltip in tooltips){
                            tooltip.SetActive
                                (!tooltip.activeSelf);
                        }
                    }
                }
}
```

A.5 RaycastOnObj.cs

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

public class RaycastOnObj : MonoBehaviour
{

[Tooltip ("Character's Pants.")]
public GameObject[] pants;
[Tooltip ("Character's Shirts.")]
public GameObject[] shirts;
[Tooltip ("Character's Accessories.")]
public GameObject[] accessories;
[Tooltip ("The raycasts' length.")]
public float rayLength = 100f;
[Tooltip ("Layers to hit when raycasting.")]
public LayerMask layerToRaycast;
[Tooltip ("Character's underwear.")]
public GameObject underwear;
[Tooltip ("Character's hair.")]
public GameObject hair;
[Tooltip ("Equip sound.")]
public AudioSource equipSound;


private Valve.VR.EVRButtonId gripButton =
    Valve.VR.EVRButtonId.k_EButton_Grip;
private Valve.VR.EVRButtonId padButton =
    Valve.VR.EVRButtonId.k_EButton_SteamVR_Touchpad;
```

```
private Valve.VR.EVRButtonId triggerButton =
    Valve.VR.EVRButtonId.k_EButton_SteamVR_Trigger;

private SteamVR_Controller.Device device { get { return
    SteamVR_Controller.Input ((int)controller.index); } }

private SteamVR_TrackedObject controller;

// Use this for initialization
void Start ()
{
controller = GetComponent <SteamVR_TrackedObject> ();
}

// Update is called once per frame
void Update ()
{
if (device == null) {
Debug.Log ("Controller not initialized");
return;
}

if (device.GetPress (padButton)) {

Ray pointerRaycast = new Ray (transform.position, transform.forward);
RaycastHit pointerCollidedWith;
var rayHit = Physics.Raycast (pointerRaycast, out pointerCollidedWith,
    rayLength, layerToRaycast);

if (pointerCollidedWith.collider != null) {
pointerCollidedWith.transform.SendMessage ("HitByRay");
if (device.GetPressDown (triggerButton)) {
        equipSound.Play ();
        if (pointerCollidedWith.collider.gameObject.layer ==
            LayerMask.NameToLayer ("Pants")) {
                foreach (GameObject pant in pants) {
                        if ((pant.name ==
                            pointerCollidedWith.collider.name)) {
                                if (pant.activeSelf) {
                                        pant.SetActive (false);
                                        underwear.SetActive (true);
                                } else {
                                        pant.SetActive (true);
                                        underwear.SetActive (false);
                                }
                        } else {
                                pant.SetActive (false);
```

```
                    }
                }
            }
            if (pointerCollidedWith.collider.gameObject.layer ==
                LayerMask.NameToLayer ("Shirts")) {
                    foreach (GameObject shirt in shirts) {
                        if ((shirt.name ==
                            pointerCollidedWith.collider.name)) {
                                if (shirt.activeSelf) {
                                        shirt.SetActive (false);
                                } else {
                                        shirt.SetActive (true);
                                }
                        } else {
                                shirt.SetActive (false);
                        }
                    }
            }
            if (pointerCollidedWith.collider.gameObject.layer ==
                LayerMask.NameToLayer ("Accessories")) {
                    foreach (GameObject accessory in accessories) {
                        if ((accessory.name ==
                            pointerCollidedWith.collider.name)) {
                                if (accessory.activeSelf) {
                                        accessory.SetActive (false);
                                        if (accessory.name.Contains
                                            ("hat")) {
                                                hair.SetActive (true);
                                        }
                                } else {
                                        accessory.SetActive (true);
                                        if (accessory.name.Contains
                                            ("hat")) {
                                                hair.SetActive (false);
                                        }
                                }
                        } else {
                                if (accessory.name.Contains ("Boots") &&
                                    pointerCollidedWith.collider.name.Contains
                                    ("Boots")) {
                                        accessory.SetActive (false);
                                }
                        }
                    }
            }
}
}
```

```
}
}

}
```

A.6 Rotator.cs

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

public class Rotator : MonoBehaviour {

        public int speed;

        // Use this for initialization
        void Start () {

        }

        // Update is called once per frame
        void Update () {
                if (this.isActiveAndEnabled) {
                        transform.Rotate (Vector3.forward * Time.deltaTime
                            * speed);
                }
                this.enabled = false;
        }

        void HitByRay(){
                this.enabled = true;
        }

}
```

Bibliografia

[1] I. E. Sutherland, "The ultimate display," *Multimedia: From Wagner to virtual reality*, 1965.

[2] H. Fuchs and G. Bishop, "Research directions in virtual environments," 1992.

[3] M. A. Gigante, "Virtual reality: definitions, history and applications," *Virtual Reality Systems*, pp. 3–14, 1993.

[4] C. Cruz-Neira, "Virtual reality overview," in *SIGGRAPH*, vol. 93, pp. 1–1, 1993.

[5] Merriam-Webster, "Virtual reality," June 2015. [Online; posted 01-June-2015].

[6] J. Jerald, *The VR book: Human-centered design for virtual reality*. Morgan & Claypool, 2015.

[7] D. Brewster, *The Stereoscope; Its History, Theory and Construction, with Its Application to the Fine and Useful Arts and to Education, Etc.* John Murray, 1856.

[8] A. B. Pratt, "Weapon.," May 16 1916. US Patent 1,183,492.

[9] Heailing *et al.*, "Stereoscopic-television apparatus for individual use," Oct. 4 1960. US Patent 2,955,156.

[10] N. Negroponte, "Virtualreality: Oxymoron or pleonasm.," June 1993. [Online; posted 01-June-1993].

[11] J. A. Jones, E. A. Suma, D. M. Krum, and M. Bolas, "Comparability of narrow and wide field-of-view head-mounted displays for medium-field distance judgments," in *Proceedings of the ACM Symposium on Applied Perception*, pp. 119–119, ACM, 2012.

[12] S. Cumming, "New vr and ar applications driving market growth, reports bcc research.," March 2016. [Online; posted 30-March-2016].

[13] W. Mason, "Oculus rift review: The age of virtual reality begins here.," March 2016. [Online; posted 28-March-2016].

[14] J. White, "Google daydream view vr review: lots of style but lacks substance.," November 2016. [Online; posted 10-November-2016].

[15] L. Prasuethsut, "Google daydream view review.," November 2016. [Online; posted 10-November-2016].

[16] A. Vlachos, "Advanced vr rendering," March 2015. [Online; posted 6-March-2015].

[17] A. Wawro, "Eagle flight dev shares lessons learned about making comfy vr games," November 2016. [Online; posted 3-November-2016].

[18] T. Gori, "Come si addestra l'esercito con la realtà virtuale e aumentata in italia," December 2016. [Online; posted 13-December-2016].

[19] F. Reply, "Nasce l'era della cybertherapy in medicina," June 2016. [Online; posted 22-June-2016].

[20] J. Durbin, "Mother uses vr headset during childbirth to avoid epidural," December 2016. [Online; posted 21-December-2016].

[21] M. P. Kenney and L. S. Milling, "The effectiveness of virtual reality distraction for reducing pain: A meta-analysis.," 2016.

[22] T. Amirtha, "Can virtual reality help women cope with childbirth?," December 2016. [Online; posted 9-December-2016].

[23] C. C. Ian King, "Hospitals try giving patients a dose of vr," August 2016. [Online; posted 29-August-2016].

[24] M. Slater and S. Wilbur, "A framework for immersive virtual environments (five): Speculations on the role of presence in virtual environments," *Presence: Teleoperators and virtual environments*, vol. 6, no. 6, pp. 603–616, 1997.

[25] M. Slater and A. Steed, "A virtual presence counter," *Presence: Teleoperators and virtual environments*, vol. 9, no. 5, pp. 413–434, 2000.

[26] M. Slater, "Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 364, no. 1535, pp. 3549–3557, 2009.

[27] T. C. Peck, S. Seinfeld, S. M. Aglioti, and M. Slater, "Putting yourself in the skin of a black avatar reduces implicit racial bias," *Consciousness and cognition*, vol. 22, no. 3, pp. 779–787, 2013.

[28] P. Salomoni, C. Prandi, M. Roccetti, L. Casanova, L. Marchetti, and G. Marfia, "Diegetic user interfaces for virtual environments with hmds: a user experience study with oculus rift," *Journal on Multimodal User Interfaces*, pp. 1–12, 2017.

[29] S. M. La Valle, *Virtual Reality*. Cambridge University Press, 2016.

[30] R. Welch and D. Warren, "Handbook of perception and human performance, vol 1, sensory processes and perception," 1986.

[31] J. E. Cutting, "Potency, and contextual use of different in_fi) rmation about depth," *Perception of space and motion*, p. 69, 1995.

[32] J. M. Loomis and J. M. Knapp, "Visual perception of egocentric distance in real and virtual environments," *Virtual and adaptive environments*, vol. 11, pp. 21–46, 2003.

[33] R. S. Renner, B. M. Velichkovsky, and J. R. Helmert, "The perception of egocentric distances in virtual environments-a review," *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, p. 23, 2013.

[34] S. Coren, L. M. Ward, and J. T. Enns, "Sensation and perception," 1994.

[35] B. D. Lawson, "Motion sickness symptomatology and origins.," 2014.

[36] R. S. Kennedy and M. G. Lilienthal, "Implications of balance disturbances following exposure to virtual reality systems," in *Virtual Reality Annual International Symposium, 1995. Proceedings.*, pp. 35–39, IEEE, 1995.

[37] J. J. Jerald, *Scene-motion-and latency-perception thresholds for head-mounted displays*. PhD thesis, University of North Carolina at Chapel Hill, 2010.

[38] J. T. Reason and J. J. Brand, *Motion sickness*. Academic press, 1975.

[39] M. Treisman, "Motion sickness: an evolutionary hypothesis," *Science*, vol. 197, no. 4302, pp. 493–495, 1977.

[40] A. Vlachos, "Advanced vr rendering performance," March 2016. [Online; posted 16-March-2016].

Ringraziamenti

Ringrazio il Professor Donatiello e il Dott. Marfia per il costante aiuto e incoraggiamento che mi hanno dato nello sviluppo dell'applicazione e nella stesura della tesi.

Ringrazio i colleghi e amici che ho avuto il piacere di conoscere e con i quali ho condiviso diversi progetti durante questi due anni di Università a Bologna: Jacopo, Chun, Andrea, Giacomo, Antonio e Ramy

Ringrazio Maria Chiara per l'enorme aiuto economico e morale, i suoi consigli e la sua perseveranza nello spronarmi a non arrendermi e ad espandere la mia visione delle cose.

Grazie.