ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Corso di Laurea in Ingegneria Informatica

SVILUPPO E PROGETTAZIONE DI APPLICAZIONI DI REALTÀ VIRTUALE PER GOOGLE CARDBOARD E PIATTAFORME GOOGLE VR

Elaborato in
SISTEMI OPERATIVI L-A

Relatore
Prof. ALESSANDRO RICCI

Presentata da
MATTIA BALZANI

Prima Sessione di Laurea
Anno Accademico 2016 – 2017

PAROLE CHIAVE

Realtà Virtuale

Google Cardboard

Sviluppo di Applicazioni VR

Progettazione di Applicazioni VR

Mobile VR

Questo lavoro è dedicato a mia madre Leoni Sfenia. So che sono tempi difficili ma la tua famiglia ti sarà sempre accanto.

Indice

Introduzione

L'evoluzione della tecnologia negli ultimi anni ha permesso di utilizzare nuovi strumenti per aumentare l'immersione all'interno di una realtà virtuale realizzata ad hoc da parte degli sviluppatori. Questi nuovi dispositivi permettono all'utilizzatore di provare una totale immersione (quindi sia a livello visivo che uditivo) all'interno di questa realtà. Tra questi vi sono caschi con visori stereoscopici, guanti per interagire con i vari componenti realizzati all'interno degli applicativi e rivelatori di movimento impiegati come dispositivi di motion tracking ed head tracking. In questa tesi vengono definiti i concetti alla base della realtà virtuale (VR) e di progettazione di un'applicazione che la implementa, descritte le metodologie di interazione con il mondo virtuale insieme alle tecnologie impiegate, illustrate le caratteristiche del visore Google Cardboard congiuntamente agli ambienti di sviluppo ad esso collegati ed analizzato un caso di studio costituito da un'applicazione di realtà virtuale realizzata per questa piattaforma. L'obiettivo principale di questa tesi è approfondire tutte le tematiche sopra citate e fornire una valutazione critica su Google Cardboard, sulla progettazione di applicazioni VR per questa tipologia di dispositivo e sull'ambiente di sviluppo utilizzato. L'idea alla base dell'app presentata è quella di realizzare un piccolo videogioco nel quale l'operazione di puntamento viene effettuata con il movimento della testa e tutte le operazioni vengono gestite con un singolo pulsante. L'elaborato è stato realizzato utilizzando Unity 3D, un software multipiattaforma per la realizzazione di videogiochi sia 2D che 3D. La tesi è suddivisa in 5 capitoli:

1. Nel primo capitolo si propone una introduzione alla realtà virtuale ed allo sviluppo di applicazioni per questo medium, oltre che una panoramica generale sullo stato dell'arte relativo ad essa;

2. All'interno del secondo capitolo vengono illustrate le metodologie di interazione con il mondo virtuale insieme alle tecnologie utilizzate per implementarle;

3. Nel terzo capitolo viene illustrato Google Cardboard insieme a tutte le sue caratteristiche peculiari;

4. Il quarto capitolo è dedicato all'introduzione degli svariati ambienti di sviluppo che supportano Cardboard, all'illustrazione di Unity3D e del SDK Google VR;

5. All'interno dell'ultimo capitolo viene illustrato l'elaborato realizzato seguendo i principi di sviluppo per applicazioni VR.

Capitolo 1

Realtà Virtuale e sviluppo applicazioni

In questo primo capitolo viene introdotta la definizione di realta' virtuale (VR o Virtual Reality), un parte degli elementi che la contraddistinguono, alcune terminologie ad essa collegate, lo stato dell'arte di questa tecnologia, alcuni dei dispositivi piu' diffusi che la implementano ed alcune nozioni riguardo lo sviluppo di applicazioni per essa.

1.1 Realtà virtuale

Negli ultimi anni la realtà virtuale sta prendendo sempre più piede in moltissimi ambiti e sta cominciando a riscuotere anche un buon successo a livello commerciale. Per questo moltissime aziende e software house stanno iniziando ad attrezzarsi per supportare questa nuova piattaforma. Prima di tutto tuttavia bisogna rispondere a una domanda. Cos'è la realtà virtuale?

1.1.1 Definizioni

Dare una definizione univoca di realtà virtuale è molto difficile in quanto questa si presta a svariate interpretazioni. Una interpretazione può derivare dalle definizioni di "realtà" e "virtuale". Cercare di definire la parola "realtà" è molto complicato poiché può derivare in discussioni di carattere filosofico nel momento in cui la si osserva da punti di vista differenti. Il *Webster's New Universal Unabridged Dictionary* [1] la definisce come "Lo stato o la qualità di essere reale. Qualcosa che esiste indipendentemente dall'idea che lo riguarda. Ciò che costituisce una cosa attuale o reale distinta da qualcosa che è semplicemente apparente". La definizione che questo dizionario fornisce della parola virtuale è "Esistere in sostanza, ma non nei fatti". Prendendo queste

due definizioni e semplificandone i concetti si può definire la realtà virtuale come un luogo che esiste e che noi possiamo sperimentare. Esistono altresì alcune definizioni di realtà virtuale fornite da esperti del settore.

A titolo di esempio, uno degli scienziati più attivi nel campo della realtà virtuale, Ivan Sutherland, definisce il concetto di realtà virtuale come il potenziale "Paese delle Meraviglie" realizzabile attraverso un'adeguata programmazione di un display definitivo [2]. L'ingegnere del software Frederick Phillips Brooks, Jr. invece, definisce la realtà virtuale come un'esperienza in cui l'utente è completamente ed effettivamente immerso in un mondo virtuale con la possibilità di averne un controllo dinamico (per esempio: comandi vocali, gesti, ecc.) [3]. In un un altro caso l'ingegnere e programmatore John Carmack, CTO presso Oculus VR, sostiene che nella sua essenza la realtà virtuale non consista in altro che sentirsi liberi dalle limitazioni della realtà attuale. E ancora, William R. Sherman e Alan B. Craig definiscono la realtà virtuale come uno strumento che ci permette di provare esperienze simulate molto simili a quelle della realtà fisica [4]. In queste dichiarazioni vengono utilizzati alcuni termini che sono elementi cardine della definizione di realtà virtuale e che ne costituiscono l'essenza. Questi sono quattro: il *mondo virtuale*, l'*immersione*, il *feedback sensoriale* e l'*interattività*.

1.1.2 Immersione

Considerando che l'utente deve essere immerso all'interno di una realtà alternativa, una definizione più semplice di VR potrebbe essere "immersione all'interno di un punto di vista o realtà alternativi".

L'immersione scaturisce dalla completa immedesimazione che l'utente prova all'interno del mondo virtuale. Questa può essere definita come la sensazione di trovarsi all'interno di un ambiente e può differenziarsi in due stati di immersione distinti: quello *mentale* e quello *fisico*.

Si parla di immersione mentale nel momento in cui l'utente entra in uno stato di coinvolgimento molto profondo all'interno dell'esperienza virtuale, mentre si parla di immersione fisica quando si entra fisicamente all'interno dell'ambiente virtuale. Questo è possibile attraverso l'utilizzo di tecnologie per stimolare i sensi del corpo umano ma non significa che l'intero corpo o tutti i sensi siano coinvolti nell'esperienza.

Un termine che forse interpreta meglio il concetto più generico di immersione all'interno della VR è il senso di presenza. La presenza può essere identificata con il sentirsi mentalmente immersi, il che costituisce il concetto alla base della realtà virtuale.

1.1.3 Feedback sensoriale

A differenza di molti altri *media* tradizionali, la VR permette ai partecipanti di poter vivere e sperimentare esperienze simulate molto simili a quelle della realtà fisica, riducendo in questo modo i pericoli correlati alla suddetta e creando scenari impossibili da sperimentare nel mondo reale. Per far ciò l'elemento fondamentale è il feedback sensoriale. Questo risulta possibile solo attraverso delle risposte sensorie date dal sistema virtuale nei confronti del corpo fisico dell'utente. Questi stimoli possono essere di svariato tipo e dipendono dalla tecnologia utilizzata. Possono essere di tipo visivo, uditivo, tattile, olfattivo e del gusto. La tecnologia odierna ancora non permette ingannare al meglio i sensi di gusto e olfatto, per cui le esperienze di realtà virtuale vengono solitamente trasmesse attraverso la vista, l'udito e, in alcuni casi, il tatto.

1.1.4 Interattività

Per fare in modo che la VR sembri autentica è fondamentale che questa risponda alle azioni dell'utente, cioè che sia interattiva. L'esperienza VR può fornire svariate tipologie di ambienti interattivi, si pensi per esempio ad ambienti costruiti ad hoc per fornire un'esperienza specifica oppure a scenari collaborativi all'interno dei quali più utenti interagiscono tra loro all'interno del mondo virtuale, infine è possibile fornire scenari competitivi che spingano più utenti ad agire l'uno contro l'altro durante le interazioni.

1.2 Definizione derivante da questi elementi

Combinando tutti questi aspetti fondamentali è possibile fornire una definizione più adatta al concetto di realtà virtuale: la realtà virtuale è uno strumento composto da simulazioni informatiche interattive che percepiscono la posizione del partecipante, le sue azioni e sostituiscono o aumentano le risposte ad uno o più dei suoi sensi, lasciandogli in questo modo la sensazione di essere mentalmente immerso o presente all'interno del mondo virtuale.

Questa sensazione di immersione può essere ottenuta tramite da svariate tipologie di tecnologie in base al senso che deve essere trattato. Per quanto riguarda la vista questa operazione viene effettuata utilizzando dei caschi o visori che presentano al proprio interno dei display sui quali ogni fotogramma viene suddiviso per ciascun occhio in modo indipendente. Esistono inoltre tecniche per emulare l'utilizzo dei visori utilizzando dispositivi comunemente in possesso dell'utente, ad esempio gli smartphone, montati su particolari supporti che rendono una sensazione simile a quella che si può ottenere con i dispositivi precedentemente citati ad una frazione del costo. In entrambi i

casi i dispositivi sono dotati di particolari lenti che permettono di percepire l'immagine in maniera corretta anche se lo schermo si trova fisicamente a pochi centimetri di distanza dai bulbi oculari. Per quanto riguarda l'udito invece vengono utilizzate delle semplici cuffie con un buon grado di isolamento per riprodurre l'audio delle varie esperienze virtuali oppure degli altoparlanti possono essere direttamente inseriti all'interno dei vari caschi/visori.

Figura 1.1: Alcune tipologie di visori

Per simulare il tatto sono stati sviluppati dei dispositivi simili a dei guanti che permettono di "toccare" e quindi percepire come veri gli oggetti inseriti nelle varie esperienze. Un esempio di questa tecnologia è un guanto/esoscheletro chiamato Dexmo, progettato da una compagnia cinese chiamata Dexta Robotics [5].

Figura 1.2: Dexmo. Guanto per la realtà virtuale

1.3 Realtà aumentata e realtà mista

Esistono anche altri filoni molto importanti oltre a quello della realtà virtuale. Questi sono la realtà aumentata (AR) e la realtà mista (Mixed Reality o MR).

La realtà aumentata è una tecnologia con cui è possibile arricchire la percezione sensoriale umana tramite informazioni, quindi dati/immagini o modelli 3D, che non sarebbero percepibili con i 5 sensi e quindi dal mondo reale.

La realtà mista invece nasce dalla fusione del mondo reale con un mondo virtuale in maniera tale da produrre un nuovo ambiente in cui oggetti digitali e fisici coesistono e possono interagire in tempo reale. La MR comprende quindi il concetto di AR e VR.

Entrambi questi settori, insieme a quello della realtà virtuale, sono soggetti a ricerca e sviluppo da svariati anni. Spesso queste sono associate agli smartphone o per esempio a tecnologie quali *Google Glass* (per AR) o *Microsoft Hololens* (sia per AR che MR), ma possono essere implementate da qualsiasi dispositivo possegga una camera.

Figura 1.3: Google Glass e Microsoft Hololens

1.4 Stato dell'arte

La storia della realtà virtuale è legata ai computer, allo sviluppo delle interfacce di interazione uomo-macchina e alla computer grafica, anche se inizialmente fu nell'ambiente cinematografico che si iniziò a sperimentare in tal senso.

Nel 1955 Morton Heilig, regista e cameraman americano, nel suo saggio "The Cinema of the Future" riteneva Cinerama (un sistema di proiezione su un grande schermo curvo) e i film 3D come la naturale evoluzione del medium, teorizzando un "cinema dell'esperienza" che potesse coinvolgere tutti i sensi in modo realistico.

Nel 1956 realizzò un dispositivo meccanico chiamato Sensorama (brevettato nel 1962) e cinque film per esso. Il dispositivo consisteva in una cabina con schermi stereoscopici, altoparlanti stereo e sedia semovibile che inoltre produceva vibrazioni, vento ed odori ma non permetteva interazione. Il progetto fallì per gli alti costi e la mancanza di finanziamenti.

Figura 1.4: Una pubblicità del sensorama

Nel 1960 Heilig brevettò un Apparato Stereoscopico-Televisivo per l'Utilizzo Individuale [6], avente molte similitudini con gli HMD degli anni '90, che includeva anche meccanismi per manifestare sensazioni olfattive e uditive oltre che visive.

Nel 1961 gli ingegneri della Philco, Charles Comeau e James Bryan, crearono un HMD che pilotava una videocamera in remoto seguendo i movimenti della testa [7].

Nel 1963 lo studente Ivan Sutherland realizzò Sketchpad [8] come tesi del proprio dottorato accademico presso il MIT. Sketchpad era il primo programma che permetteva di realizzare della computer grafica interattiva. Questo utilizzava una penna ottica, per effettuare operazioni di tipo selettivo o di disegno, in aggiunta a una tastiera.

Nel 1965 Sutherland introdusse il concetto di "Ultimate Display" durante la propria presentazione al International Federation for Information Processing (IFIP) Congress [2]. Questo rappresentava il concetto di un display tramite cui l'utente poteva interagire con oggetti appartenenti ad un mondo in cui non erano rispettate le leggi di quello fisico. Questi lo definì come "uno specchio per il paese delle meraviglie matematico". Nella sua descrizione erano inclusi gli stimoli visivi e tattili.

Nel 1968 Ivan Sutherland, divenuto docente ad Harvard, insieme allo studente Bob Sproull creò quello che è considerato il primo sistema di realtà virtuale e realtà aumentata con visore, chiamato "The Sword Of Damocles" [9]. Il nome fu ispirato dal fatto che il visore, essendo molto pesante, era appeso al soffitto del laboratorio tramite un braccio meccanico. Tale dispositivo è anche considerato un precursore della realtà aumentata in quanto le sue lenti lasciavano intravedere l'ambiente circostante. Nello stesso anno Sutherland, insieme al professore David Evans, fondò la Evans Sutherland Corporation presso l'Università dello Utah.

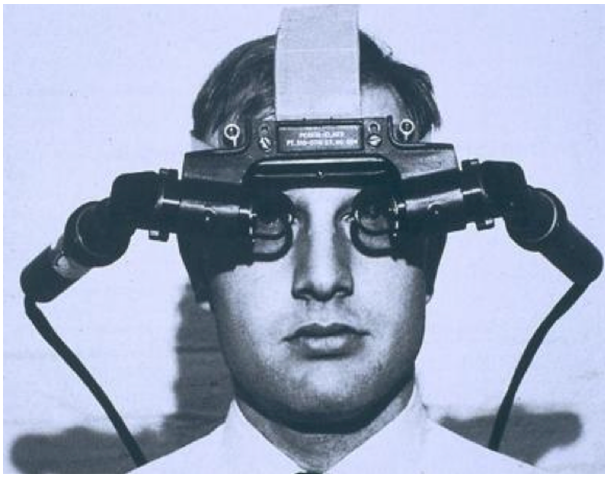

Figura 1.5: Immagine del visore di Sutherland: The Sword of Damocles

Nel 1972 Atari sviluppò Pong, un gioco che portò al pubblico la prima grafica interattiva multi utente in tempo reale.

Nel 1976 fu ultimato il prototipo di un laboratorio di realtà aumentata chiamato Videoplace da Myron Krueger [10]. L'idea alla base del progetto era quella di creare una realtà artificiale che circondasse gli utenti e rispondesse ai loro movimenti e alle loro azioni senza l'utilizzo di altri dispositivi (come occhiali o guanti).

Nel 1977 al MIT venne creato l'Aspen Movie Map, uno dei primi hypermedia e sistemi di realtà virtuale e precursore del moderno Google Street View. Il sistema simulava un tour virtuale ed interattivo nella città di Aspen in Colorado, nella sua versione estiva, invernale e digitale.

Nei primi due casi il sistema si basava su una serie di filmati del luogo montati in modo da coprire ogni possibile percorso tra le strade della città, mentre la terza era una ricostruzione poligonale texturizzata della città (poco realistica a causa dei limiti tecnologici dell'epoca). L'utente interagendo con delle icone poteva muoversi lungo le vie della città assistito da una cartina virtuale e cliccando sugli edifici poteva ottenere informazioni ad essi inerenti quali foto storiche, menu di ristoranti, ecc.

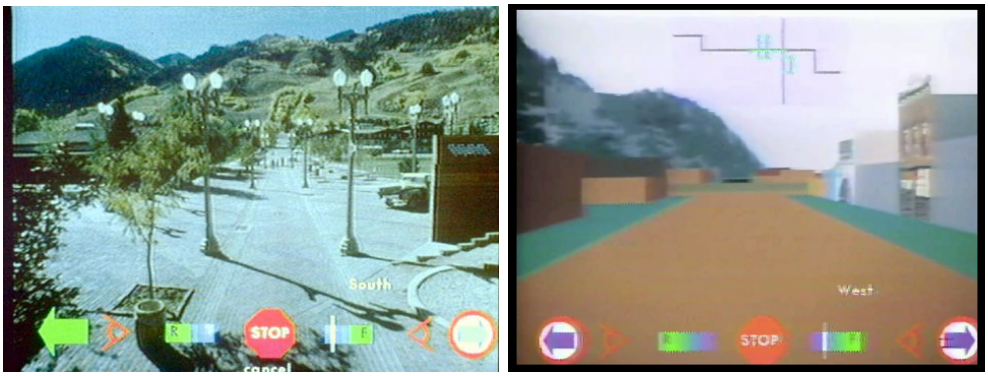

Figura 1.6: Aspen Movie Map (1997), navigazione in modalità foto e virtuale

In questo anno fu anche sviluppato il Sayre Glove presso l'Electronic Visualization Lab [11], Università dell'Illinois (Chicago). Questi erano guanti che utilizzavano dei tubi conduttivi per trasmettere quantità variabili di corrente in base alla piegatura delle dita. L'informazione ottenuta veniva poi interpretata da un computer per stimare la configurazione della mano dell'utente.

Lo stesso anno uscì anche Tron, considerato il primo film a focalizzarsi sulla realtà virtuale.

Figura 1.7: Locandina del film Tron (1982) della Walt Disney

Nel 1979 Eric Howlett sviluppò il sistema LEEP (Large Expanse Enhanced Perspective) per implementare la possibilità di poter mostrare un ampio

orizzonte visivo da un display di piccole dimensioni. Questa tecnologia fu poi integrata nei primi HMD sviluppati dalla NASA.

Nello stesso anno presso AT&T Bell Labs, Gary Grimes sviluppò un dispositivo per misurare la posizione della mano chiamato Digital Entry Data Glove.

Nel 1981 divenne operativo il Super Cockpit (mostrato poi su Aviation Week nel 1985) presso la Wright Patterson Force Air Base [12]. Questi includeva un HMD trasparente, montato sull'elmetto del pilota, che permetteva di visualizzare diverse informazioni in base alla direzione in cui questi rivolgeva lo sguardo. Per esempio guardando l'ala dell'aereo era possibile sapere quanti missili erano a disposizione per il lancio.

Lo stesso anno presso il MIT un team cominciò a lavorare su un prototipo di display di visore per realtà aumentata che permetteva agli utenti di esplorare campi come il disegno 3D, la visualizzazione architettonica e il layout 3D dei chip per computer. Il dispositivo utilizzava uno specchio semiargentato per sovrimprimere un'immagine del computer sopra le mani dell'utente, o altre parti del corpo.

Nel 1984 Thomas Zimmerman e Jaron Lanier fondarono la VPL Research Inc., la prima società a vendere occhiali e guanti per la realtà virtuale (il più famoso è il Data Glove). Nel 1989 Lanier ebbe il merito di rendere popolare il termine "realtà virtuale".

Nel 1987 Scott Foster fu contattato dalla NASA per creare un dispositivo che simulasse l'emanazione del suono generato in uno specifico spazio 3D, utilizzando un algoritmo sviluppato presso l'Università dello Wisconsin (Madison).

Nel 1989 la Mattel produsse Power Glove, un guanto che fungeva da controller per la console Nintendo Entertainment System.

Figura 1.8: Il Power Glove, controller per Nintendo Entertainment System (1989)

Nel 1990 la W-Industries produsse Virtuality, una serie di sistemi di virtual reality per sale giochi. I sistemi comprendevano Head-Mounted Display con schermi LCD, speaker, microfoni, joystick, volanti o cloche da utilizzare in postazione eretta o da seduti (dipendeva dalla tipologia di gioco). Inoltre questi dispositivi permettevano di effettuare un riconoscimento dei movimenti della testa e della posizione dei controller, tramite dei sensori e dei campi magnetici, riproducendoli all'interno dell'ambiente virtuale.

Figura 1.9: Due esempi di cabinato del Virtuality Group

Nel 1991 Sega annunciò Sega VR, un headset per la realtà virtuale per console e videogiochi arcade. Questi possedeva schermi LCD, cuffie stereo e sensori inerziali per il tracciamento della testa.

Figura 1.10: Headset Sega VR per Sega Genesis / MegaDrive (1991) di Sega

Nel 1992 presso l'Electronic Visualization Laboratory venne realizzato il CAVE [13] (Cave Automatic Virtual Environment), un teatro di realtà virtuale con video e audio 3D in alta risoluzione delle dimensioni di una stanza. I video venivano proiettati dall'esterno sulle tre pareti e sul pavimento, mentre la visione stereoscopica era realizzata tramite degli speciali occhiali con sensori di movimento. L'utente poteva inoltre muoversi all'interno della stanza e la prospettiva corretta di visione veniva applicata in real time garantendo un'immersione totale.

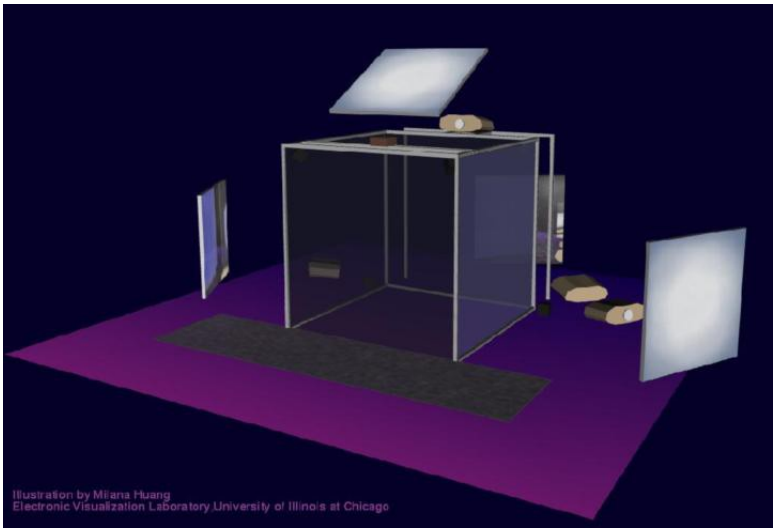

Figura 1.11: Struttura del CAVE (1992), teatro per la realtà virtuale immersiva

Nel 1995 Nintendo produsse Virtual Boy, console stereoscopica monocromatica che si rivelò un insuccesso commerciale.

Figura 1.12: Virtual Boy (1995), console stereoscopica monocromatica di Nintendo

In questo anno EVL introdusse ImmersaDesk, un sistema di proiezione VR a schermo singolo che lavorava con la libreria CAVE, permettendo quindi una semplice migrazione per le applicazioni tra una dispositivo e l'altro.

Nello stesso anno Fakespace, Inc introdusse un sistema che permetteva a due persone di avere differenti prospettive dallo stesso sistema di proiezione. Questo sistema era chiamato Dual User Option (DUO).

Nel 1999 venne rilasciata una libreria open source gratuita per il tracciamento sviluppata per la realtà aumentata chiamata ARToolKit. Questa è nata da una collaborazione tra l'università di Washington, quella di Seattle, l'Human Interfaces Technology Lab e l'ATR Media Integration Communication di Kyoto (Giappone). Sebbene fosse stata realizzata per l'AR, questa libreria forniva un metodo di tracciamento video che rendeva possibile effettuare il tracking utilizzando solo un computer equipaggiato con una videocamera (in modo particolarmente semplice e poco costoso).

In questo anno uscì anche un film di fantascienza vincitore di 4 premi Oscar scritto e diretto da Larry e Andy Wachowski che sdoganò il tema della realtà virtuale. Questi era intitolato "Matrix".

Figura 1.13: Locandina del film Matrix (1999)

Nel 2007 google introdusse StreetView, un servizio che mostrava visuali panoramiche di un numero sempre crescente di posizioni del mondo come strade, edifici e aree rurali.

Il 2014 e l'inizio del 2015 segnano una rinascita per la realtà virtuale, a seguito di un periodo di stagnazione, durante il quale grandi aziende dell'intrattenimento e della socialità virtuale hanno fatto grandi annunci in questo settore.

Nel marzo 2014 Facebook acquista Oculus VR, la compagnia produttrice del promettente Head-Mounted Display Oculus Rift, per due miliardi di dollari: il visore verrà immesso sul mercato in seguito, nel marzo del 2016.

Nello stesso mese alla Game Developers Conference (GDC 2014) Sony annuncia di essere al lavoro su un visore per la realtà virtuale per la console PlayStation 4 chiamato Project Morpheus, poi rinominato PlayStation VR e uscito nell'ultimo trimestre del 2016.

Nel giugno 2014 Google annuncia Google Cardboard durante la conferenza per sviluppatori Google I/O 2014. Questi non è altro che un visore indossabile, nella sua forma base costruito in cartone, su cui inserire uno smartphone.

Nel gennaio 2015 Microsoft annuncia HoloLens, un caschetto wireless di realtà aumentata e mixed reality in cui sono incorporati avanzati sensori e componenti di computazione.

A inizio marzo dello stesso anno HTC e Valve rendono nota la loro collaborazione nella realizzazione di un proprio sistema di realtà virtuale, composto da un visore, una coppia di controller wireless e due stazioni di base per il tracciamento. Questi è chiamato HTC Vive ed è uscito successivamente nell'aprile 2016.

1.5 Visori moderni

Lo sviluppo della realtà virtuale è legato in maniera indissolubile alle evoluzione tecnologiche e ai dispositivi che queste comportano. Oggigiorno esistono due tipologie di dispositivi per realizzare la realtà virtuale:

- I visori che necessitano di un'unità di elaborazione separata collegata al dispositivo (tutti visori di qualità superiore come Oculus, HTC Vive e Playstation VR);,

- quelli liberi da questa tipologia di vincolo (dispositivi mobile con supporto).

Tutti i visori indossabili sul viso (in inglese HMD, *Head-Mounted Display*) più sofisticati hanno delle unità di elaborazione esterne a loro connesse (Personal Computer o Console), per ovvi motivi di potenza computazionale necessaria per elaborare immagini ad alto dettaglio. Di questi i più diffusi sono Oculus Rift (Oculus VR), HTC Vive (Valve e HTC) e Playstation VR (Sony). Tutti e tre implementano anche un tracciamento dell'ambiente mediante delle telecamere da posizionare attorno all'utente per coglierne al meglio i movimenti del corpo e della testa attraverso la rilevazione di alcuni led posti sulla scocca dei visori.

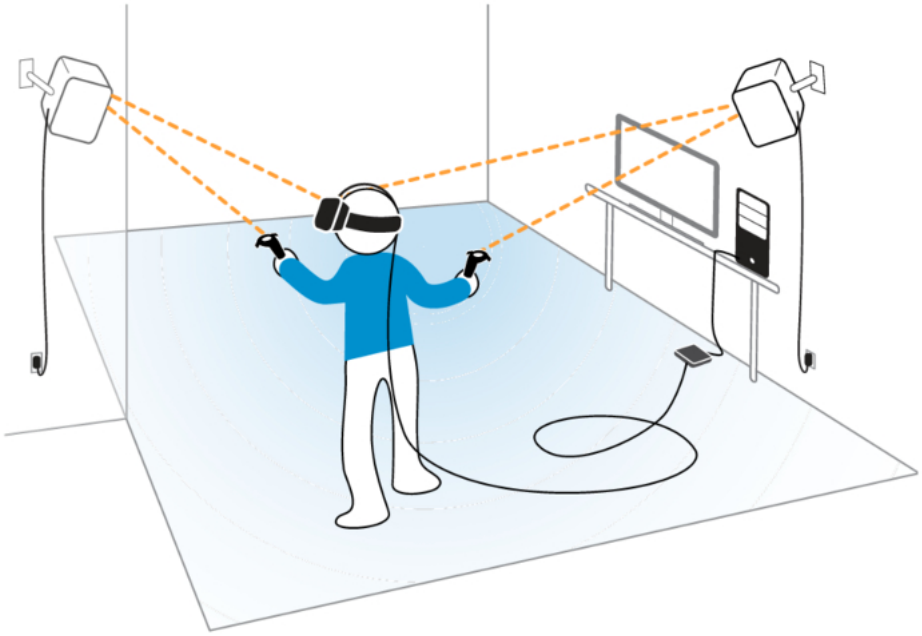

Figura 1.14: Setup della stanza con le telecamere

I visori senza unità di elaborazione esterna in realtà sono semplici supporti creati per essere utilizzati congiuntamente ad uno smartphone. L'utilizzo di questo espediente permette di superare l'alta densità di cavi che pongono vincoli ai movimenti dell'utente durante l'utilizzo dell'apparecchio, pagando il prezzo della maggiore libertà con severe limitazioni tecniche rispetto ai modelli precedentemente illustrati. Esistono svariate tipologie di supporti di questo tipo, che hanno generato un ampio ventaglio di scelta a seconda del prezzo e della qualità costruttiva.

Di seguito sono elencate le caratteristiche di alcuni dei visori di maggior successo partendo da quelli che necessitano di una unità di elaborazione dedicata fino a quelli senza fili.

1.5.1 Oculus Rift [14]

Oculus Rift è un head-mounted display prodotto dalla società Oculus VR fondata da Palmer Luckey e recentemente acquistata da Facebook. Il dispositivo è stato rilasciato sul mercato nel marzo del 2016 ed è stato preceduto da due versioni per gli sviluppatori chiamate Developer Kit 1 e Developer Kit 2. Questo visore è dotato di uno schermo OLED a bassa persistenza da 2160 x 1200 pixel (1080 x 1200 pixel per occhio), con un refresh rate a 90 Hz e un ampio angolo di visione pari a 110 gradi. Al suo interno è dotato di diver-

si sensori che permettono di riprodurre i movimenti all'interno dell'esperienza virtuale: accelerometro, giroscopio, magnetometro e tracking posizionale a 360 gradi. Il visore è dotato anche di microfono e audio 3D integrato basato su una tecnologia di Visisonic chiamata RealSpace 3D Audio [15].

Questo HMD è dotato di un meccanismo che permette di modificare la distanza tra le lenti per una visione più confortevole. Il tracciamento della posizione è effettuato attraverso un sensore a infrarossi da porre anteriormente al giocatore collegato a una porta USB al fine di permettere al visore di essere utilizzato in una stanza mentre si è seduti, in piedi o si sta camminando. Questi si chiama Costellation e può essere usato sia in singolo che insieme ad altri sensori dello stesso tipo.

Figura 1.15: Immagine di Oculus Rift in azione

1.5.2 HTC Vive [16]

HTC Vive è un head-mounted display uscito nell'aprile del 2016 e nato dalla collaborazione tra due società: Valve [17] e HTC [18]. All'interno di questo visore vi è uno schermo OLED con risoluzione 2160x1200 (1080x1200 per occhio), con refresh rate di 90 Hz e un angolo di visualizzazione pari a 110 gradi. I movimenti sono tracciati in un'area equivalente a circa 4,5x4,5 metri, tramite diversi sensori all'interno del visore: accelerometro, giroscopio e videocamera. Inoltre sono usati anche i due rilevatori laser esterni, da posizionare nell'ambiente di utilizzo. Per collegarsi al PC, necessario per elaborare le immagini, HTC Vive ha bisogno di una connessione USB e una HDMI [19]. All'interno della confezione vi sono anche due controller, chiamati Steam VR

Controller, che possono trasformarsi in oggetti virtuali di ogni genere all'interno delle esperienze. Come per Oculus Rift, anche HTC Vive dispone di una videocamera esterna utilizzata per tracciare il movimento del giocatore all'interno dell'ambiente che potrebbe essere utilizzata anche per realizzare alcune esperienze di realtà aumentata.

Figura 1.16: Immagine di HTC Vive in azione

1.5.3 PlayStation VR [20]

PlayStation VR è un HMD rilasciato sul mercato da Sony [21] nell'ottobre del 2016 per la sua console da gioco PlayStation 4 [22]. Oltre a questo dispositivo, per funzionare correttamente, il visore necessita di una periferica chiamata PlayStation Camera, che include quattro microfoni, una doppia videocamera per rilevare la profondità oltre ai movimenti degli utenti e dei controller nello spazio. Questa camera effettua il tracciamento della posizione grazie ai led situati su PlayStation VR, sui controller PlayStation Move e sul gamepad DualShock 4. Questo visore ha uno schermo OLED da 5,7 pollici con risoluzione Full HD di 1920x1080 pixel (960x1080 per occhio), con una frequenza di aggiornamento che varia da 90 Hz a 120 Hz e un campo visivo di circa 100 gradi. Per effettuare il collegamento alla console dispone di una porta HDMI e una USB. Il rilevamento dei movimenti come accennato è affidato all'accelerometro, al giroscopio e ai 9 led disposti sul visore.

Figura 1.17: Immagine di PlayStation VR in azione

1.5.4 Google Cardboard [23] e Daydream View [24]

Il progetto più economico è stato presentato da Google al Google I/O 2014 con il nome commerciale di Google Cardboard. Esso non è altro che un supporto in cartone dotato di lenti su cui è possibile posizionare uno smartphone: con l'esigua cifra di 4 $ è possibile riuscire a vivere un'esperienza di realtà virtuale (o aumentata) semplicemente avviando un'applicazione installata sul proprio telefono cellulare. Come schermo questo visore monta il display dello smartphone utilizzato (quindi ha una risoluzione variabile in base al dispositivo). Questo viene suddiviso in due parti, una per occhio, e le lenti sono montate sul supporto in cartone. La stesa casa di sviluppo ha reso disponibile sul proprio sito una guida per creare un Cardboard, questo per mostrare la semplicità con cui è possibile realizzarne una propria versione.

Figura 1.18: Immagine del visore Google Cardboard

Il Cardboard ha permesso un approccio alla VR totalmente differente rispetto al passato dando la possibilità agli sviluppatori di poter realizzare contenuti senza temere la dipendenza da una periferica costosa (a differenza degli HMD precedentemente accennati). In questo modo l'esperienza VR non risulta più solo ad appannaggio di colui che può permettersi il "costoso" dispositivo ma entra nelle case di chiunque con forza dirompente e ad un prezzo veramente esiguo.

Nel 2016 Google ha annunciato e rilasciato un'evoluzione di fascia alta per questa tipologia di dispositivi chiamata Daydream View con una propria piattaforma denominata Daydream [25] che non fa altro che estendere il concetto di Google Cardboard ad una fascia di smartphone più performate e appositamente creata. Ad oggi solo Google Pixel/Pixel XL, Motorola Moto Z/Moto Z Force, Huawei Mate 9 Pro/Porsche Design Mate 9 e ZTE Axon 7 supportano Daydream insieme a Samsung Galaxy S8/S8+ ed Asus Zenfone AR che verranno immessi sul mercato durante l'anno in corso. Google View presenta, oltre al visore, un controller con sistema di tracking integrato per migliorare l'esperienza offerta da Cardboard. Daydream condivide con Cardboard la libreria Google VR, in questo modo viene permesso di sviluppare applicazioni compatibili con entrambi i dispositivi che differiscono per la modalità di immissione dei comandi, la tipologia di dispositivo utilizzato e performance dovute agli smartphone di fascia più alta.

Figura 1.19: Immagine del visore Daydream View

In seguito all'annuncio di Cardboard sono stati sviluppati svariati modelli di terze parti di questo visore (supportati ufficialmente da Google) con lenti e materiali migliori, spesso dal costo superiore all'originale.

Figura 1.20: Altre tipologie di Google Cardboard

Esistono anche altri visori *wireless* che svolgono la medesima funzione di google Cardboard, alcuni esempi sono Durovis Dive [26] o Samsung Gear VR [27]. Quest'ultimo è nato da una collaborazione tra Oculus VR e Samsung e per funzionare necessita di alcuni dispositivi specifici prodotti da Samsung (Galaxy Note 5, Galaxy S6 Edge+, Galaxy S6, Galaxy S6 Edge).

Figura 1.21: Dive e Samsung Gear VR

1.5.5 Microsoft Hololens [28]

Questo visore per la *realtà mista* (MR), realizzato da Microsoft, non si trova attualmente in commercio ma può essere acquistato nella sua versione per sviluppatori (Development Edition). Hololens può essere descritto come un computer indossabile con lenti trasparenti e che quindi permette di osservare l'ambiente circostante durante l'utilizzo. Esso dispone di sensori avanzati e audio surround (spatial sound) che permette all'utente di capire la direzione di provenienza del suono. La dotazione sensoristica di cui dispone gli permette di misurare parametri come l'inerzia e l'intensità e la direzione di provenienza della luce, inoltre è completato da quattro telecamere deputate all'analisi ambientale e da una telecamera in grado di rilevare la profondità per ricreare l'ambiente circostante oltre ad una fotocamera HD. L'acquisizione dei comandi vocali è affidata a quattro microfoni inseriti all'interno del visore.

Figura 1.22: Microsoft Hololens

Alcune delle sue potenzialità sono state mostrate durante alcuni eventi dedicati ai dispositivi Microsoft, eventi che hanno evidenziato la capacità del

dispositivo di fondere la realtà virtuale con quella fisica, restituendo una realtà mista con la quale l'utente può interagire.

Figura 1.23: Versione per Hololens del famoso gioco Minecraft sviluppato da Mojang e di proprietà di Microsoft

1.6 Campi d'applicazione dei sistemi VR, AR e MR

La realtà virtuale è adatta ad essere sfruttata in numerosi settori: basti immaginare alla possibilità di effettuare simulazioni di situazioni altrimenti troppo pericolose o troppo dispendiose da ricreare nella realtà fisica. La VR può dare la possibilità di visitare altri mondi oppure luoghi impossibili o molto difficili da raggiungere: nei videogiochi permette di visitare mondi virtuali in cui l'utente può interagire con altri giocatori o con entità dotate di una propria intelligenza artificiale. Ancora, si pensi al campo delle arti sceniche ed alla possibilità di assistere ad un evento o ad uno spettacolo in modo indipendente dalla propria posizione nella realtà fisica, semplicemente indossando un visore di realtà virtuale. Tutto questo può essere reso possibile da questo tipo di

realtà simulata, la quale offre una capacità di riscontro da parte dell'utente e un livello di interazione che nessun altro mezzo può garantire.

Grazie all'utilizzo della realtà aumentata è possibile ottenere informazioni inerenti ad un incarico lavorativo da svolgere in maniera immediata senza dover perdere tempo nella ricerca di informazioni. Tutto questo al fine di arricchire la realtà fisica con informazioni che i sensi umani non sarebbero in grado di elaborare in così poco tempo.

Con la realtà mista infine è possibile fondere elementi della realtà virtuale e di quella fisica per crearne una versione in cui gli oggetti di una interagiscono con quelli dell'altra. Un esempio potrebbe essere un applicazione di arredamento in cui visualizzando una stanza è possibile inserire i vari mobili nella posizione desiderata e fermarli nello spazio così da vederne in tempo reale la realizzazione.

Come sovente accade a molte tecnologie innovative, uno dei primi campi di applicazione della realtà virtuale è stato quello militare e aerospaziale. Durante gli anni '60 veniva usata all'interno di simulazioni di volo volte ad addestrare i piloti, impiego che trova tutt'ora riscontro all'interno delle moderne tecniche di formazione [29]. Recentemente l'esercito americano ha introdotto sistemi di realtà aumentata e mista nell'addestramento e nelle esercitazioni in situazioni di guerra anfibia [30] oppure in operazioni belliche allo scopo di monitorare ed ottenere informazioni riguardo e tramite i propri commilitoni in tempo reale, in maniera tale da garantire un controllo costante dell'ambiente e delle risorse umane impiegate per poter intervenire, di conseguenza, molto più rapidamente in caso di situazioni pericolose [31].

Figura 1.24: Soldato americano durante un addestramento con Oculus Rift

La realtà virtuale viene sfruttata anche in campo automobilistico da molte case costruttrici, le quali la utilizzano per svolgere test sui propri veicoli a partire dalle prime fasi dello sviluppo, quando ancora non si dispone di un prototipo funzionante, oppure per sperimentare in sicurezza le modifiche all'assetto e ai parametri della vettura utili a migliorare le prestazione durante le competizioni automobilistiche [32].

Figura 1.25: Immagine promozionale del simulatore Motion Pro II della CXC Simulators con l'utilizzo di Oculus Rift

Anche nel campo della medicina le simulazioni VR rivestono un ruolo importante: recentemente la realtà virtuale è stata introdotta come strumento di formazione per i medici chirurghi, permettendogli così di sperimentare interventi di ogni tipo evitando di rischiare la vita di pazienti reali.

Figura 1.26: Impiego delle tecnologie VR nelle simulazioni di intervento chirurgico

In campo architettonico i sistemi VR offrono la possibilità di rendere ispezionabili i progetti prima della loro costruzione o renderli visitabili ai loro clienti all'interno di un mondo virtuale [33].

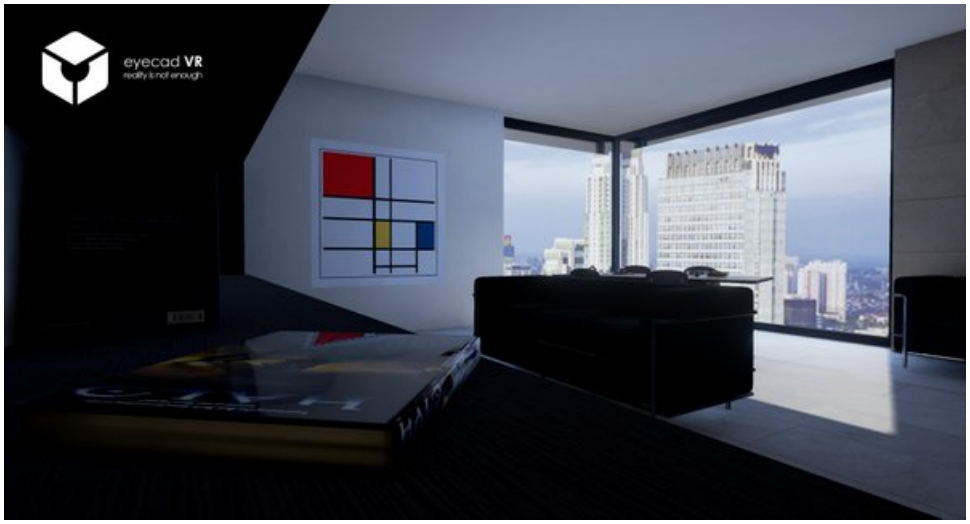

Figura 1.27: Immagine del software Eyecad VR per ambito architettonico

Nel campo delle arti si assiste ad un massiccio impiego della VR: molti musei, anche italiani, si stanno attrezzando affinché vengano realizzate delle riproduzioni virtuali dei propri spazi. A Parigi è stata realizzata un applicazione per Cardboard che permette di visitare alcune parti della città all'interno di un ambiente virtuale esplorabile a 360 gradi [34]. Anche alcuni sviluppatori

stanno realizzando delle applicazioni che permettano di visitare musei virtuali ad alta risoluzione, permettendo così la fruizione ad una vasta platea di una importante fetta del patrimonio culturale mondiale altrimenti di difficile fruizione [35].

Figura 1.28: Impiego delle tecnologie VR per visitare musei

L'applicazione più recente che ha reso celebre la VR a livello commerciale è quella videoludica. Come si evince dalla storia di questo medium questo settore ha sempre avuto una certa affinità con questo tipo di realtà. Già negli anni '90 furono fatti i primi tentativi di approccio alla realtà virtuale ma tutti ebbero scarso successo per colpa di limitazioni tecniche e tecnologiche che non riuscivano a rendere appagante l'esperienza fornita. Negli ultimi anni si è potuto assistere a una ribalta di questo medium in questo settore e praticamente tutte le case produttrici di videogiochi si stanno attrezzando per poter realizzare esperienze virtuali più o meno complesse.

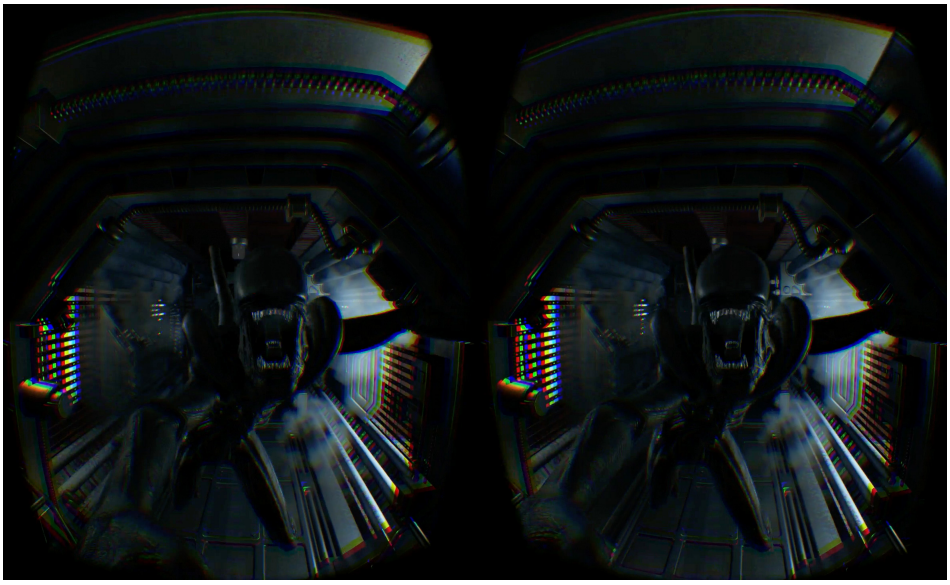

Figura 1.29: Alien Isolation di Sega in esecuzione su Oculus Rift e HTC Vive

1.7 Progettazione e sviluppo di applicazioni di realtà virtuale

La VR al giorno d'oggi può essere applicata su dispositivi di qualsiasi tipologia come sistemi mobile, architetture PC (Mac OS X, Windows e Linux) o simil PC (le console) e non è più solo ad appannaggio di hardware costruiti per l'occorrenza, quali visori o camere per l'esperienza virtuale, ma riesce ad essere implementata direttamente tramite software su dispositivi non nati per questa evenienza (sistemi basati su iOS ed Android quali gli smartphone e i tablet). Come precedentemente accennato la realtà virtuale ha già dimostrato di poter apportare svariati benefici per raggiungere risultati in svariati campi d'applicazione e conseguentemente sono nati alcuni indicatori che aiutano a valutarne l'adeguatezza agli svariati compiti affrontati. Generalmente non è saggio approcciarsi alla VR per *risolvere* un problema, tranne in applicazioni di marketing o che esplorano artisticamente il medium, mentre è consigliassimo valutarne l'applicazione come *possibile soluzione*. Una volta presa una decisione sull'impiego della VR all'interno del proprio applicativo vi sono svariate *tecniche di progettazione standard* che andrebbero seguite per ottimizzare il raggiungimento dell'obiettivo del software.

Molti aspetti di progettazione e sviluppo di applicazioni di realtà virtuale sono comuni a quelli seguiti durante la generazione di mondi in 3 dimensioni non stereoscopici. Uno di questi aspetti consiste nel ricreare dei comparti

grafici e sonori mediante l'utilizzo di appositi programmi di modellazione poligonale quali Blender [36], Maya [37], Modo [38], 3DStudioMax [39], Photoshop [40], ecc. e audio quali Logic Pro X [41], Ableton Live [42], Steinberg Cubase [43], Audacity [44], ecc. La riproduzione di un mondo credibile e coerente, agli occhi dell'utente, è indispensabile per suscitare il senso di immersione caratteristico dei software VR. Un altro aspetto comune alle normali applicazioni risulta essere la costruzione dell'architettura del software e delle interazioni tra i vari componenti all'interno del codice utilizzando dei pattern di progettazione per la risoluzione di problematiche ben note. Ricordando che un'*architettura del software* di un programma viene definita da Bass, Clements e Kazman come la struttura/e di un sistema comprendente i componenti software, le loro proprietà visibili esternamente e le relazioni fra essi [45], un *design pattern* è definibile come una struttura di una soluzione comprovata ed efficace per la risoluzione di problemi molto simili tra loro [46]. Esistono svariate tipologie di pattern, ognuno dei quali adatto a risolvere specifiche problematiche. Alcuni esempi possono essere il pattern *Observer*, per la gestione di differenti comportamenti azionati da eventi, lo *State* pattern, per la gestione di oggetti che mutano il proprio comportamento in base ad uno stato interno, il *Flyweight* pattern per ridurre il consumo di memoria mediante il riutilizzo delle risorse in comune tra svariati oggetti, o il *Model-View-Controller* (*MVC*) pattern per la realizzazione delle interfacce utente all'interno degli applicativi.

Calcolando che difficilmente il publico, attualmente, ha già sperimentato la realtà virtuale e le metodologie di interazione che questa comporta, a livello software, risulta indispensabile guidare l'utente mediante suoni ed elementi grafici all'interno dell'esperienza affinché possa apprenderne le meccaniche di interazione alla base. Per questo motivo la progettazione dovrà essere focalizzata completamente sul rendere il più accessibile possibile l'esperienza per l'utente in maniera ancora più marcata rispetto alle normali applicazioni non stereoscopiche.

Un metodo molto importante per migliorare questo aspetto consiste nel coinvolgere gli utenti attraverso dei test dell'applicativo in tutti gli stadi dello sviluppo in modo tale da poter rifinire l'esperienza implementando nuove funzioni, testando ed analizzando il lavoro svolto direttamente durante il suo avanzamento. In questo maniera è possibile capire quali idee funzionano e quali devono essere necessariamente scartate oppure se l'utente interagisce correttamente con il mondo di gioco o se qualcosa risulta poco intuitivo. Data la giovinezza del campo della realtà virtuale è solo attraverso questo studio e testing continuo che è possibile percepire quali sono gli aspetti più rilevanti nell'interazione tra uomo e mondo virtuale. Sviluppare applicazioni di realtà virtuale *impone dei vincoli*, spesso generati dall'hardware utilizzato, che possono far insorgere delle problematiche durante la progettazione. Risulta quindi

di vitale importanza per il progettista costruire una parte delle meccaniche caratterizzanti il mondo virtuale sulla base delle tecnologie scelte.

Capitolo 2

Interfacciarsi con il mondo virtuale

In questo capitolo viene definito un aspetto fondamentale dei sistemi VR e su cui si costruisce tutta l'esperienza all'interno del mondo generato, l'*interfacciamento dell'utente con il mondo virtuale*. Questo è definito da tre elementi fondamentali che vanno assolutamente tenuti in considerazione durante lo sviluppo di un qualsiasi sistema VR:

- come trasmettere gli input dell'utente al sistema;
- come trasmettere gli output del sistema all'utente;
- come gestire le interazioni dell'utente con gli elementi all'interno del sistema.

Di seguito saranno analizzati tutti questi aspetti.

2.1 Gli input

Il modo in cui i partecipanti interagiscono con un sistema VR influenza moltissimo la loro esperienza con il mondo virtuale. Le modalità di interazione determinano enormemente la semplicità d'uso di un sistema, l'immersione mentale degli utenti e le possibili loro azioni. Un partecipante ad una esperienza VR influenza il mondo virtuale attraverso l'interfaccia di ingresso del sistema. Un componente chiave di questa tecnologia è il tracciatore della posizione (position tracker), che può essere utilizzato per monitorare la posizione del corpo e i movimenti dei partecipanti, così come altri oggetti fisici (i supporti) a loro accessibili. Esiste una grande varietà di tecnologie disponibili per effettuare il tracciamento della posizione, ognuna di queste ha dei pro e

dei contro ed è possibile combinare varie opzioni per migliorare le prestazioni di tracking. Questa si rivela fondamentale per le prestazioni complessive in quanto una porzione significativa dei processi di input consiste nel monitorare le azioni dell'utente. Tracciare la posizione fisica dei partecipanti e i loro movimenti permette una grossa integrazione (immersione) nel mondo virtuale. Un tipico sistema VR normalmente traccia la testa e almeno una mano del partecipante.

2.1.1 Metodologie di input per il mondo virtuale

I sistemi VR hanno svariate funzionalità di tracking per le interazioni dei partecipanti con il mondo virtuale. Questi sistemi variano in funzione di come il computer traccia l'utente e di come questi immette i propri comandi per il mondo virtuale. Sia i movimenti che gli input dei partecipanti sono delle parti importanti di un ambiente virtuale immerso, anche se nella definizione di realtà virtuale precedentemente citata si parla solo di tracciamento del corpo. Possono essere definite due tipologie di input:

- *input passivo* (eventi azionati dal sistema che monitora il partecipante);
- *input attivo* (eventi specificatamente azionati dall'utente).

2.1.2 Position tracking

Un sensore di posizione è un dispositivo che riporta la sua posizione e/o il suo orientamento ad un elabortore. Tipicamente alcuni di questi dispositivi vengono installati in una posizione conosciuta mentre altri vengono applicati sugli oggetti che devono essere tracciati. Di solito questi sensori vengono utilizzati per tracciare la testa dell'utente e le sue mani. Il sensore di posizione è il dispositivo più importante all'interno di un sistema di realtà virtuale poiché comunica al sistema dove si trova l'utente all'interno dello spazio VR. Esistono svariate tipologie di sensori di posizione, ognuno dei quali ha i suoi punti di forza e le sue limitazioni, ed ognuna di queste impone dei limiti allo sviluppo a seconda del sistema su cui sono utilizzate. Questi limiti dipendono sia dalla tipologia di sensore utilizzata ma anche dalla tecnologia messa in campo per determinare la relazione tra un'origine fissa ed un sensore. Per esempio alcuni tracciatori (o trackers) necessitano di una linea visiva ininterrotta tra l'apparato ricevitore e quello trasmettitore poiché se questa venisse interrotta il sistema di tracciamento non potrebbe funzionare correttamente. Nei sistemi position-sensing (percettivi alla posizione), devono essere valutati tre aspetti in contrapposizione l'uno con l'altro (oltre ai costi):

1. la precisione e la velocità di segnalazione della posizione da parte del sensore;
2. il materiale (per esempio metalli o oggetti opachi);
3. l'ingombro (fili e collegamenti meccanici).

Nessuna tecnologia, per quanto possa essere costosa, fornisce delle condizioni ottimali per tutti e tre questi aspetti. Per questo motivo i progettisti devono ben considerare come sarà utilizzato il sistema VR e decidere quali possano essere i compromessi ottimali in base alla tipologia di sistema. L'unica considerazione fondamentale è che questi sia in grado di assicurare un'esperienza accettabile all'utente. Il rumore, la poca precisione del sensore di posizione e i ritardi ad esso collegati diminuiscono il realismo o l'immersività dell'esperienza, portando anche alla nausea in soggetti predisposti (fenomeno del motion sickness), cosa che va assolutamente evitata. Di seguito si riportano le metodologie di tracciamento più diffuse, alcune delle quali comunemente implementate nei sistemi VR attuali:

- elettromagnetico;
- meccanico;
- ottico;
- videometrico;
- ad ultrasuoni;
- inerziale;
- neurale.

2.1.3 Tracking elettromagnetico

Questo metodo di tracciamento utilizza un trasmettitore per generare un campo magnetico da tre bobine poste ortgogonalmente tra loro all'interno dell'unità. A loro volta queste generano correnti in un altro gruppo di bobine nella piccola unità ricevente indossata dall'utente. Il segnale ricevuto da ciascuna bobina del ricevitore è utilizzato per determinarne la propria posizione relativa al trasmettitore. L'unità trasmittente è fissata con una posizione e un orientamento ben noti così che possa essere calcolata la posizione assoluta dell'unità ricevente. Normalmente vengono piazzate più unità riceventi sull'utente (tipicamente sulla testa e sulle mani), o su ogni sostengo utilizzato. Il

tracciamento elettromagnetico funziona poiché le bobine si comportano come delle antenne e il segnale si indebolisce man mano che l'antenna ricevente si allontana dall'antenna trasmittente. La forza del segnale cambia anche in base all'orientamento relativo tra le bobine trasmettenti e le riceventi, ogni bobina ricevente riceve un segnale più forte quando il suo orientamento è lo stesso di quello dell'antenna trasmittente. Analizzando la forza del segnale di tutte le bobine il sistema può determinare in modo corretto la posizione in base ai 6 gradi di libertà (6 -DOF) per ciascuna unità ricevente rispetto alla posizione del trasmettitore (coordinate assi x, y, z, rotazione, inclinazione, imbardata).

Alcune limitazioni di questo sistema di tracciamento sono:

- possibili interferenze causate dal metallo presente nell'ambiente;
- ridotta portata del campo magnetico generato. Se l'utente si allontana troppo la precisione cala drasticamente. Posizionare più trasmettitori per incrementare la portata è possibile ma risulta più difficile da implementare.

I pregi invece:

- non vi è alcuna restrizione dovuta alla linea visiva. Per questo il giocatore può muoversi liberamente anche in ambienti che presentino ostacoli visivi tra lui e il trasmettitore (ostacoli che possono interferire con altre tipologie di dispositivo di tracciamento);
- sistema wireless che riduce notevolmente l'ingombro.

2.1.4 Tracking meccanico

Metodologia di tracciamento in cui vengono utilizzati dispositivi meccanici per misurare la posizione dell'utente. Un esempio è il Fakespace FS2 [47], che si propone come un'asta articolata molto simile a un braccio. Questo dispositivo può essere utilizzato legandolo alla testa oppure manovrandolo con le mani. Il dispositivo segue i movimenti in un raggio molto ridotto restituendo un alto grado di precisione.

Figura 2.1: Immagine di Fakespace FS2

Svantaggi distintivi:

- i collegamenti fisici restringono i movimenti dell'utente in una posizione fissa nel mondo virtuale o in una porzione di area limitata;
- la fluidità dei movimenti può essere compromessa per colpa dell'inerzia dovuta alla struttura;
- non può essere utilizzato per tracciare sia le mani che la testa dell'utente.

Vantaggi particolari:

- sono solitamente utilizzati come display per la testa oppure come dispositivo tattile di I/O per le mani.

2.1.5 Tracking ottico

I dispositivi che lo implementano utilizzano informazioni visive per tracciare l'utente. Esistono molti modi per farlo ma il più comune è quello di utilizzare una videocamera per "guardare" l'oggetto o la persona da tracciare. Normalmente questa è situata in una determinata locazione e grazie a delle tecniche visive è possibile determinare la posizione dell'oggetto in base a ciò che la videocamera rileva. Se si utilizza un singolo dispositivo è possibile tracciare l'obiettivo solo in 2 dimensioni, mancando quindi la profondità, mentre

se viene visualizzato da più sensori il sistema può triangolare la sua posizione e/o il suo orientamento nelle 3 dimensioni.

Figura 2.2: Videocamera del sistema ottico di Oculus Rift

Limitazioni distintive:

- la linea visiva tra l'oggetto tracciato e la videocamera deve sempre essere libera affinché funzioni;
- limitazione dei movimenti dell'utente dovuto alla linea visiva.

2.1.6 Tracking videometrico

Il tracciamento videometrico funziona esattamente nella maniera opposta a quello ottico. In questo caso è l'oggetto da tracciare che indossa il dispositivo sensore e guarda l'ambiente circostante. In questo modo il sistema VR analizza le immagini che acquisisce dallo spazio circostante e posiziona dei landmark grazie ai quali può calcolare la posizione della videocamera rispetto ad essi. Per esempio la videocamera può essere montata sulla testa dell'utente, come head-base display, e fornire l'input al sistema VR. Questi può determinare, grazie ai dati fornitigli, la posizione degli angoli della stanza circostante e di conseguenza ricavare la posizione dell'utente. Per far sì che funzioni è però necessario che la locazione dei landmark nello spazio sia conosciuta oppure separabile dal resto dei dati affinché sia possibile determinare la posizione assoluta del dispositivo. Si può ridurre la computazione necessaria rendendo i landmark distinti in base al colore o alla sagoma, in maniera tale che l'algoritmo di visione del computer possa tracciare velocemente punti multipli e distinguerli da altri

oggetti. Questi landmark agiscono come punti di riferimento conosciuti nel mondo.

2.1.7 Tracking ad ultrasuoni

Il tracciamento ad ultrasuoni utilizza dei suoni a tonalità molto alta emessi a intervalli regolari per determinare la distanza tra il trasmettitore (uno speaker) e il ricevitore (un microfono). Come per il tracciamento ottico, la combinazione di tre ricevitori e tre trasmettitori fornisce dati sufficienti ad un sistema per triangolare i 6 gradi di libertà di un oggetto.

Figura 2.3: Esempio di dispositivo Logitech

Svantaggi rilevanti:

- le performance di tracciamento calano drasticamente in ambienti rumorosi. Il suono deve avere una linea di propagazione libera tra speaker e microfoni affinché possa essere determinato correttamente il tempo di arrivo del segnale;
- i trasmettitori e i ricevitori devono essere posizionati a una distanza minima. E' un problema per i ricevitori poiché potrebbero captare suoni non voluti tra loro.

Vantaggi:

- basso costo dei dispositivi utilizzati (speaker, microfoni e computer);
- portata del tracciamento espandibile economicamente.

2.1.8 Tracking inerziale

Questa tipologia di tracciamento utilizza strumenti elettromeccanici per catturare il movimento relativo dei sensori misurando il cambiamento nelle forze giroscopiche, accelerazione e inclinazione [48]. Questi strumenti integrano accelerometri e inclinometri, i quali servono sia per misurare l'accelerazione, e possono essere utilizzati per determinare la nuova posizione di un oggetto in movimento conoscendone il punto di partenza, sia ad individuare l'inclinazione del suddetto oggetto. Il tracciamento inerziale opera con la stessa tecnica con la quale l'orecchio interno aiuta a determinare l'orientamento della testa. Un fluido tende a rimanere fermo mentre la struttura circostante ruota. I sensori all'interno della struttura trasmettono l'informazione sulla locazione della struttura relativa al fluido. Il cervello utilizza questa informazione per calcolare l'orientamento e i cambiamenti dello stesso nel tempo. Questi dispositivi, di piccolissime dimensioni, vengono posti sugli oggetti che si vogliono tracciare e devono essere accoppiati a dei computer. Grazie a giroscopio, accelerometro e inclinometro, presenti all'interno di ogni dispositivo, è possibile misurare cambiamenti di posizione all'interno di 6 gradi di libertà; bisogna tuttavia tenere conto che nel tempo la rilevazione della posizione potrebbe diventare errata a causa di un accumulo di errori, dovuti ad una misurazione relativa e non assoluta della posizione. Questa tipologia di dispositivo viene utilizzata negli Head-Mounted Display, per tracciare i movimenti della testa, in combinazione ad altre metodologie di tracking per la gestione dei movimenti. Il tracciamento magnetico, per esempio, può fornire la posizione dovuta al movimento, seppure ad un ritmo minore, e può essere utilizzato per correggere la deriva nel sistema inerziale.

Svantaggi distintivi:

- possibile necessità di ricalibrazione manuale dovuta alla degradazione della precisione nel tempo. Ciò avviene poiché gli accelerometri forniscono misurazioni relative, quindi non assolute, e nel tempo gli errori continuano ad accumularsi nel sistema;
- necessità di utilizzare un altro sistema di tracking per rilevare la posizione assoluta dell'utente.

Vantaggi peculiari:

- nessun limite di portata poiché questi dispositivi costituiscono unità autonome che non necessitano di altri componenti per assolvere la loro funzione;
- lavorano molto velocemente e introducono solo minimi ritardi nel sistema;

- possibilità di combinare il tracciamento inerziale con altre metodologie di tracking per migliorare le prestazioni.

2.1.9 Tracking neurale (o muscolare)

Il tracciamento neurale o muscolare è un metodo per rilevare il movimento relativo di singole parti rispetto ad altre sezioni del corpo. Questa metodologia di tracking non è adatta al tracciamento della posizione dell'utente nell'ambiente designato, tuttavia può essere molto utile per tracciare il movimento delle dita o altre estremità. Dei sensori molto piccoli sono attaccati alle dita o agli arti in modo tale da garantire il mantenimento della loro posizione nel tempo. Questi sensori misurano i cambiamenti dei segnali nervosi o le contrazioni muscolari e trasmettono al sistema VR informazioni rispetto alla posizione delle zone monitorate. Questa tipologia di dispositivi risulta ideale per tracciare attività sportive o ad esempio il suonare il violino virtuale, e in generale tutti quei contesti ove differenti movimenti del braccio, delle mani e delle dita possono portare a risultati molto diversi l'uno dall'altro.

2.2 Body tracking

Il body tracking cosituisce la capacità di un sistema VR di percepire la posizione e le azioni dei partecipanti. Le particolari componenti del movimento che vengono tracciate dipendono dalla parte del corpo presa in esame e da come il sistema viene implementato. Qualsiasi componente del corpo può essere tracciato in uno o più gradi di libertà, assumendo che sia disponibile un meccanismo di tracking, con una grandezza e peso appropriati, e attaccato al sistema. I requisiti di un'esperienza VR determinano la mole di tracciamento del corpo da coprire, sebbene le possibili limitazioni possano forzare lo sviluppatore verso implementazioni che sostituiscono il tracciamento di alcune parti del corpo con altre forme di interazione.

2.2.1 Le posture del corpo e i gesti

Nel monitorare l'utente, il sistema determina la posizione corrente dell'utilizzatore o quella di alcune delle sue parti del corpo. La posizione statica di una di queste parti o di un gruppo di queste, come per esempio una stretta di pugno, è detta *postura*. Il sistema può inoltre mantenere nel tempo le informazioni relative al movimento dell'utente. Gli spostamenti specifici più ricorrenti nel tempo vengono chiamati *gesti* e possono essere utilizzati in maniera molto efficace come metodi intuitivi di interfacciamento. Un esempio di gesto intuitivo è il movimento che si effettua per afferrare qualcosa, ad indicare

la volontà di raccogliere un oggetto all'interno del mondo virtuale. Le posture e i gesti contribuiscono ad espandere il repertorio di comandi di input al sistema anche se il loro grado di intuitività differisce enormemente da utente a utente. Le parti del corpo e le tecniche di tracking, relative ad esso, utilizzate comunemente nelle applicazioni VR includono:

- il tracking della testa;
- il tracking della mano e delle dita;
- il tracking degli occhi;
- il tracking dei piedi;
- il tracking di altre parti del corpo;
- il tracking indiretto.

2.2.2 Tracking della testa

La testa è tracciata in quasi tutti i sistemi VR, anche se non sempre in tutti e 6 i gradi di libertà, ed è attualmente la parte del corpo, insieme ai supporti per le mani, più importante da tracciare all'interno del mondo virtuale. La maggior parte dei display richiedono il tracciamento dell'orientamento della testa. Questo perché se l'utente ruotasse il capo e lo scenario non si adattasse in maniera appropriata al mutamento della visuale soggettiva, questi non si sentirebbe fisicamente immerso. Anche se non essenziale, il location tracking (tracciamento della locazione) migliora in modo rilevante la qualità delle esperienze VR poiché aiuta a fornire il senso di motion parallax (il senso con cui un oggetto cambia posizione in base al punto di vista da cui è osservato). Alcune esperienze VR evitano di effettuare location tracking incoraggiando o richiedendo all'utente uno spostamento continuo (virtuale) attraverso l'ambiente.

2.2.3 Migliorare il tracking

Ogni tipologia di tracciamento ha le sue limitazioni, ma queste possono essere ridotte, eliminate o evitate. Alcune strade per evitare queste problematiche includono l'utilizzo di analisi predittive, calibrazioni di sistema, una combinazione di metodi di tracciamento e auto-calibrazione utilizzando oggetti (supporti) nel mondo reale. L'*analisi predittiva* è un processo computazionale che può essere utilizzato per migliorare la precisione e ridurre la latenza. Effettuando delle analisi sul movimento delle unità di tracciamento è talvolta

possibile determinare il percorso che queste effettueranno nel tempo e quindi fornire dei valori riguardanti la posizione in cui saranno negli istanti futuri. Questo permette al sistema di avere una posizione plausibile in un punto preciso nel tempo anzichè limitarsi a conoscere l'ultima posizione riportata. Questa analisi è possibile solo se l'oggetto tracciato si muove in maniera predicibile e quando questo non avviene il sistema non sarà in grado di ottenere dei risultati curati e precisi nel tempo. *Calibrare* il sistema per operazioni all'interno di un ambiente specifico può aiutare a ridurre gli errori, come nel caso in cui ci siano degli oggetti metallici vicini ad un sistema con tracciamento magnetico. Indipendentemente dall'ambiente, per far sì che questi siano utilizzabili spesso si rende necessaria una calibrazione, operazione che può consistere nel comunicare al sistema dove si trovi il suo punto di origine all'avvio dell'esperienza VR. Altri sistemi permettono di effettuare una calibrazione a comando dell'utente oppure questa operazione viene effettuata al volo durante l'esecuzione dell'esperienza (on the fly). La *combinazione* di tipologie diverse di tracciamento in alcuni casi può fornire dei risultati favorevoli utilizzando i lati positivi di una metodologia per superare le limitazioni imposte da un'altra. Un esempio è l'implementazione di una videocamera a supporto di un head-mounted display per migliorare il tracciamento all'interno dell'ambiente virtuale

2.2.4 Altre metodologie di input

Altri metodi di input per l'utente includono pulsanti, joystick, supporti (anch'essi tracciati nel mondo virtuale) e piattaforme che aiutano a definire lo spazio che l'utente occuperà durante l'esperienza e a fornire un feedback coerente all'input all'utente. Occasionalmente un sistema VR potrebbe includere un ingresso vocale per fornire metodi di comunicazione naturali con il mondo virtuale. Altre tipologie di input possono essere utilizzate per monitorare il mondo reale o un mondo virtuale persistente, raccogliendo dati sulle qualità dinamiche del mondo atte a cambiare nel tempo, oppure possono raccogliere informazioni del mondo reale e trasformarle in segnali elettrici processabili dai computer (dispositivi chiamati trasduttori). Altri metodi prevedono invece di utilizzare i dati del mondo reale per creare porzioni del mondo virtuale. I dispositivi di input vanno scelti molto attentamente a seconda dell'applicazione poiché forniscono la base per definire come i partecipanti interagiranno con il sistema. Molti di questi dispositivi possono essere utilizzati per svariati scopi generici mentre altri nascono per compiti specializzati data la loro natura particolare. Alcuni di questi forniscono anche un meccanismo di output simultaneo a quello d'ingresso: si pensi ad esempio ad una mazza da golf tracciata, un dispositivo di input, che può anche fornire dei feedback tattili all'utente per aiutarlo a colpire la pallina.

2.3 Output di un sistema VR

L'altro aspetto chiave nell'esperienza di realtà virtuale, il come l'utente percepisce l'ambiente e la sua percezione fisica del mondo virtuale, è basata interamente su cosa viene mostrato dall'unità di elaborazione. Il sistema percettivo umano ha cinque sensi con cui fornisce informazioni al cervello. Tre di questi (la vista, l'udito e il tatto) sono comunemente sollecitati nell'utente con stimoli sintetici nell'esperienza VR. In questo modo i sistemi di realtà virtuale ingannano i sensi facendo percepire all'utente degli stimoli generati dalle unità di elaborazione al posto di quelli naturali. L'inclusione di ulteriori sensi, nella maggior parte dei casi, migliora l'immersione dell'utente nel mondo virtuale anche se allo stato dell'arte il gusto e l'olfatto non sono gestiti in maniera corretta, facendo sì che le applicazioni VR si limitino a gestire i tre sensi precedentemente citati. Esistono tre tipologie d'assetto per mostrare fornire informazioni a tutti i sensi interessati:

- assetto stazionario;
- assetto basato sulla testa (head-based);
- assetto basato sulla mano (hand-based).

Quando si parla di sistemi VR è di vitale importanza capire come gli occhi funzionano poiché le immagini in base alla tipologia di display vengono mostrate in maniera differente. Gli esseri umani percepiscono informazioni riguardo la distanza relativa degli oggetti in molte maniere; questi indicatori di distanza vengono detti *indici di profondità*. Esistono quattro varietà di indici di profondità, legati a dozzine di modi in cui è possibile percepire la distanza relativa. Questi sono:

1. *indici di profondità di immagini monoscopiche*: sono quelli che possono essere visti in una singola immagine statica di una scena, per esempio nelle fotografie o nei dipinti. Fanno parte di questa categoria l'interposizione, l'ombreggiatura, la dimensione, la prospettiva lineare, il gradiente di superficie delle texture, l'altezza nel campo visivo, gli effetti atmosferici e la luminosità.

2. *indice di profondità di immagini Stereoscopiche*: questo indice di profondità dipende quindi dal parallasse tra due immagini differenti ricevute dalla retina di ogni occhio (disparità binoculare), che non è altro che lo spostamento apparente di oggetti visti da posizioni differenti.

3. *indice di profondità di movimento*: questi derivano dal parallasse creato dal cambiamento relativo di posizione tra la testa e l'oggetto che viene osservato (uno dei due o entrambi possono essere in movimento). Questa informazione è legata al fatto che gli oggetti più vicini all'occhio vengono percepiti come più veloci rispetto a quelli distanti. Fondamentalmente questo cambio di prospettiva avviene in due modi: quando l'oggetto si muove o quando è l'osservatore a muoversi.

4. *indici di profondità fisiologici*: sono generati dai movimenti del muscolo oculare per mettere a fuoco un oggetto. Questi indici sono l'adattamento e la convergenza.

Non tutti questi indici hanno la stessa priorità, per esempio la stereoscopia è un indice di profondità molto forte che in caso entrasse in contatto con altri indici risulterebbe dominante, escludendo il movimento relativo rispetto al quale potrebbe risultare più debole. Gli indici fisiologici risultano generalmente più deboli.

2.3.1 Il campo visivo (Field of View)

Il campo visivo (Field Of View) è una caratteristica dei display visivi con cui viene misurata la larghezza angolare delle visuale dell'utente, cioè quella mostrata dal visore in un determinato momento. Il normale campo visivo di un essere umano è approssimativamente di 200 gradi, di cui 120 sono di sovrapposizione binoculare [49]. I display che forniscono 100-120 gradi di FOV ricoprono una porzione ragionevole del range visivo umano. Tuttavia questo solo indicatore può essere fuorviante e va tenuta in considerazione anche la sovrapposizione stereoscopica del campo visivo affinché possa essere ottenuto un buon risultato. Questo parametro può variare in base all'HMD poiché in ogni dispositivo le schermate degli occhi potrebbero essere separate in vari modi, nel caso in cui la sovrapposizione per entrambi gli occhi fosse più piccola di 30 gradi, sarebbe molto difficile percepire la stereoscopia. Disporre di un parametro più grande permette di ottenere una proiezione più ampia e quindi anche un campo visivo maggiore.

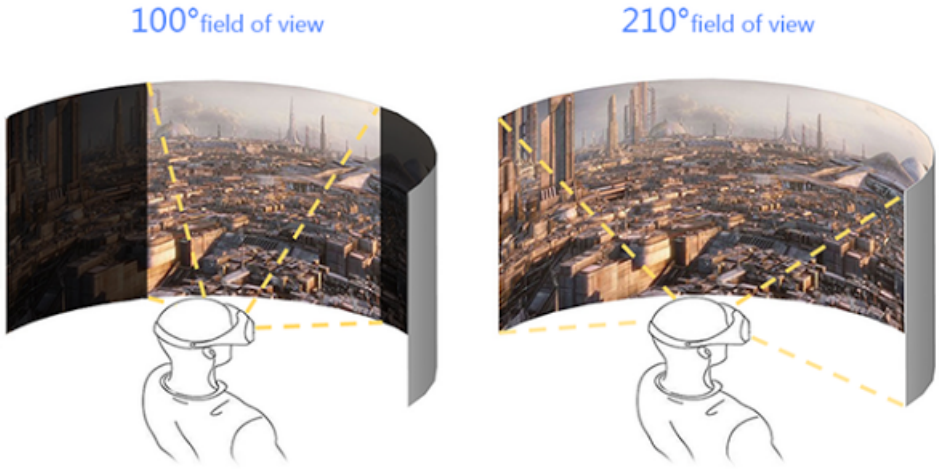

Figura 2.4: Esempio di FOV nelle applicazioni VR

2.3.2 Frame Rate

Il frame rate è la frequenza con cui le immagini vengono visualizzate ed è rappresentato dal numero di frame per secondo (FPS), anche se può essere misurato in Hertz. Genericamente il frame rate non dipende dal tipo di visore utilizzato ma dall'abilità di rendering (capacità di mostrare a schermo) grafico dell'hardware e del software, nonchè dalla complessità del mondo virtuale. Il frame rate può avere un grande effetto sull'immersione mentale e più è alto il numero di immagini per secondo mostrato e più risulterà immersiva l'esperienza. Esistono valori standard che i visori attualmente in commercio adottano: essi generalmente superano i 60 fps, nello specifico le app di Cardboard normalmente lavorano a questa frequenza mentre i visori più complessi come Oculus Rift e HTC Vive lavorano a una frequenza di 90 Hz. Mantenere un parametro alto permette di evitare effetti indesiderati come la nausea dovuta al movimento, o motion sickness (affrontato nel prossimo capitolo), e rendere l'esperienza il più fluida possibile agli occhi dell'utente.

2.4 Interazioni con il mondo virtuale

Interagire significa essere coinvolti in azioni reciproche tra un'entità e l'altra. L'interazione è una caratteristica chiave della realtà virtuale e che mag-

giormente la distingue dagli altri media. Quando il mondo virtuale risponde alle nostre azioni diveniamo sempre più coinvolti, aumentando il nostro senso di presenza all'interno del mondo virtuale. Purtroppo però la maggior parte delle interazioni possibili con le interfacce di sistemi tecnologici non sono naturali per un utente umano. Per esempio un'interfaccia povera può rendere complicata l'interazione e quindi interferire con l'abilità di concentrarsi sull'esperienza di un partecipante. Nel tempo gli utenti hanno acquisito famigliarità con le nuove tecnologie e ad oggi la sfida più grande all'interno di esperienze VR è quella di raggiungere la maggior naturalezza nelle interazioni tra utente e interfacce (sia grafiche che di comando). Un modo per raggiungere questo obiettivo è quello di costruire nuove interfacce utilizzando una metafora che mette al centro dell'attenzione la famigliarità d'interazione per l'utente. Un altra strategia utile è quella di inserire dei feedback nel momento in cui avviene un'interazione e quindi comunicare al partecipante il suo l'effettivo avvenimento. Di seguito sono elencate le forme di interazione possibili all'interno di un'esperienza di realtà virtuale:

- manipolazione;
- navigazione;
- collaborazione (interazione con altri utenti);
- comandi di sistema VR (o metacomandi).

2.4.1 Manipolazione

La manipolazione permette all'utente di modificare il mondo virtuale e gli elementi che lo occupano. Nella realtà virtuale la maggior parte delle manipolazioni avvengono in due fasi: la prima implica la selezione di qualcosa e la seconda lo svolgimento di un'azione. A volte queste due operazioni possono avvenire simultaneamente ed esistono una grossa varietà di metodi con cui è possibile manipolare gli oggetti del mondo virtuale. Principalmente si individuano quattro modi in cui sono svolte la maggior parte delle manipolazioni all'interno di un'esperienza VR:

1. *controllo diretto dell'utente*: è il metodo con cui il partecipante interagisce con gli oggetti del mondo virtuale nello stesso modo in cui lo farebbe nel mondo reale. Un esempio sono le interazioni "prendi con la mano" in cui l'utente chiudendo la mano esegue un blocco sull'oggetto virtuale e in caso uno spostamento. La maggior parte di interazioni di questo tipo utilizzano dei gesti o lo sguardo per effettuare una selezione.

2. *controllo fisico*: sono quegli oggetti che controllano il mondo virtuale tramite un apparato del mondo reale. Dispositivi di questo tipo possono essere pulsanti, interruttori e joystick.

3. *controllo virtuale*: è un controllo che si manifesta completamente all'interno del mondo virtuale. La maggior parte di questi controlli non sono altro che delle rappresentazioni virtuali di oggetti reali come per esempio dei pulsanti o degli interruttori.

4. *controllo di un agente*: sono quei controlli che permettono ad un utente di impartire comandi attraverso un intermediario come se questi fosse in comunicazione diretta con un altro agente "intelligente" che svolgerà l'azione richiesta. Questo agente può essere una persona o un'entità controllata dal computer e potrà interagire attraverso uno dei metodi di manipolazione appena citati. La comunicazione con l'agente può anche essere effettuata attraverso gesti o comunicazione vocale in maniera tale da aumentare il senso d'immersione dell'utente.

2.4.2 Navigazione

La navigazione descrive come l'utente si muove da un posto all'altro. Nel mondo reale per esempio noi navighiamo mentre camminiamo, guidiamo e voliamo. In un'esperienza VR queste sono solo alcune delle infinite possibilità di navigazione che possono essere rese disponibili nell'ambiente virtuale. Il processo di navigazione all'interno dello spazio è parte vitale dell'esperienza dell'utente e il come questi possa viaggiare attraverso il mondo può giocare un ruolo significativo nel come questo possa essere percepito dal partecipante. Esistono però alcuni aspetti critici della navigazione e una grande varietà di possibili implementazioni. La navigazione coinvolge due componenti ben distinti: il *viaggio* (come l'utente si muove nello spazio e nel tempo) e l'*orientamento* (come l'utente capisce dove si trova e dove sta andando). E' di vitale importanza che all'interno del mondo virtuale l'utente non si senta smarrito e capisca come muoversi in modo molto naturale per non diminuire il senso d'immersione dell'esperienza VR. Per questo spesso e volentieri è consigliato fornire degli aiuti all'interno dell'esperienza così da agevolare sia il viaggio che l'orientamento all'utente.

2.4.3 Le interazioni con altri utenti

Lavorare e condividere l'ambiente VR con altri utenti può giocare un ruolo importante ed esistono svariati approcci che permettono di realizzare questo

tipo di esperienza di realtà virtuale. Nel momento in cui lo scopo della condivisione dell'esperienza è il completare insieme ad altri un determinato compito allora si parla di *esperienza collaborativa*; questo non esclude che non possano essere realizzate anche *esperienze competitive* o di altro tipo. Questo tipo di esperienze condivise può generare alcune preoccupazioni legate a specifiche problematiche di implementazione. Il primo problema consiste nello scegliere i metodi con cui sono gestite le interazioni collaborative, un altro è il quanto concorrente debba essere l'esperienza; oppure chi ha il controllo sulle operazioni di manipolazione/comunicazione; come il mondo viene mantenuto concorrente; e, forse il più importante, come i vari compartecipanti possano comunicare l'uno con l'altro. Vari aspetti possono essere condivisi tra gli utenti come per esempio il mondo virtuale stesso che deve essere obbligatoriamente condiviso affinché vi sia l'interazione fra i partecipanti. Caratteristica fondamentale da considerare è che ogni utente possiede un proprio punto di vista del mondo virtuale e che anche questo può essere condiviso oppure risultare molto diverso da quello di un altro partecipante.

2.4.4 Le interazioni con il sistema VR (i metacomandi)

In alcune situazioni può sorgere il bisogno di interagire non solo con il mondo virtuale percettivo o con altri utenti all'interno del mondo, ma anche con simulazioni e strutture del mondo sottostante. Queste tipologie di interazioni includono operazioni che possono spaziare dal caricamento di nuovi dati scientifici per la visualizzazione al controllo di un agente surrogato all'interno del mondo. Questi comandi possono operare sotto la superficie del mondo virtuale e possono essere identificati come *metacomandi*. In molte esperienze VR qualcuno, esterno ai partecipanti immersi, deve manipolare il mondo affinché queste possano procedere nel modo corretto. Spesso le applicazioni VR sono progettate in maniera tale da permettere ai partecipanti immersi di specificare metacomandi, per esempio caricare modelli differenti o cambiare parametri globali o annullare un'operazione appena effettuata. L'abilità di influenzare il mondo in questa maniera può essere considerata come un modo per ridurre l'immersione mentale dell'esperienza, quindi normalmente i progettisti di applicazioni per la comunicazione sperimentale non permettono ai partecipanti di compiere questo tipo di operazioni.

Capitolo 3

Mobile VR: Google Cardboard

In questo capitolo vengono illustrati Google Cardboard, le problematiche introdotte dalla realtà virtuale e le linee guida fornite da Google per la gestione delle interazioni e del fenomeno del motion sickness.

3.1 Smartphone per la realtà virtuale?

Il dispositivo Google Cardboard prevede l'utilizzo di uno smartphone per poter funzionare, rendendo il dispositivo mobile l'elemento realmente fondamentale per realizzare l'esperienza VR. Fino a pochi anni fa sarebbe stato impensabile realizzare un visore di realtà virtuale a basso costo senza prevedere l'implementazione di un Head-Mounted Display appositamente sviluppato. Nell'arco di una decina di anni gli smartphone sono stati protagonisti di una crescita tecnologica esponenziale sia a livello di potenza di calcolo sia riguardo al numero e alla precisione dei sensori in essi integrati, necessari per la realizzazione del tracking. Ovviamente le differenze tecniche rispetto ai visori molto più costosi sono rilevanti ma la resa globale dell'esperienza risulta essere adeguata per permettere l'implementazione della realtà virtuale. Un altro fattore che ha portato ad utilizzare gli smartphone è stata l'esplosione delle app all'interno dei sistemi operativi Android ed iOS, scaricabili dai relativi PlayStore [50] ed App Store [51], che ha raggiunto il mercato mainstream con una tale forza da spingere le aziende di qualsiasi settore a realizzare applicazioni delle più varie tipologie. Nessun dispositivo di realtà virtuale ha mai avuto una base installata anche lontanamente paragonabile ad una di queste dimensioni.

3.2 La scelta di Google Cardboard

Come già accennato nel capitolo 1, Google Cardboard è stato presentato da Google al Google I/O 2014 ed è entrato immediatamente in commercio nella sua versione V1. Nel maggio del 2015 è stata immessa sul mercato una versione V2 che inserisce sul dispositivo un pulsante magnetico finalizzato a permettere interazioni senza l'ausilio di dispositivi aggiuntivi. Cardboard è un supporto su cui posizionare uno smartphone caratterizzato dal costo veramente esiguo nella sua versione base (dal prezzo inferiore ai 10 euro). Esistono svariati modelli sviluppati da terze parti che migliorano la qualità dei materiali e l'aderenza al volto ad un prezzo più alto. Il basso costo del dispositivo ha permesso a moltissimi utenti di provare per la prima volta delle applicazioni di realtà virtuale. La fascia di dispositivi VR immediatamente sopra al Cardboard comprende il Samsung Gear VR e Oculus Rift che costano rispettivamente 130 euro [40] e 700 euro [41] circa sui relativi siti ufficiali, ai quali vanno aggiunti i costi di uno smartphone Samsung Galaxy compatibile per il primo e un personal computer dalla potenza sufficiente per gestirlo per il secondo. Questo basso prezzo ha permesso un'*ottima espansione* del Carboard ed incoraggiato molti sviluppatori a costruirvi sopra delle esperienze di realtà virtuale più o meno complesse, cosa impensabile fino a poco tempo prima, considerando l'esigua base installata di tutti i dispositivi VR. Un'altra ragione che mi ha spinto verso l'impiego di questa tecnologia è il *grande supporto* fornito da Google nella produzione di applicazioni VR per i tool di sviluppo più utilizzati (Unity ed Unreal) e i sistemi operativi mobile (iOS ed Android). Creare delle applicazioni VR risulta quindi molto intuitivo e agevolato, garantendo agli sviluppatori la massima efficienza possibile all'interno del proprio progetto.

Un'altra peculiarità dell'SDK di Cardboard consiste nel fatto che l'app si *adatta automaticamente al visore utilizzato*, così come per le funzioni audio stereo, quelle per la correzione dell'immagine e per la distorsione. L'accoppiamento si effettua facendo scannerizzare al cellulare il codice QR presente sulla confezione del Cardboard e la procedura viene completata automaticamente.

La stessa casa di sviluppo ha reso disponibile sul proprio sito delle linee guida per realizzare applicazioni VR su Cardboard atte ad evitare gli errori di design più comuni agli sviluppatori. Nei prossimi paragrafi sono approfondite proprio queste tematiche.

3.3 Funzionamento di Google Cardboard

Come già accennato, Google Cardboard è un supporto su cui va inserito un hardware, per la precisione uno smartphone, il quale ha bisogno di un software ad esso dedicato per poter funzionare. Per questo è necessario che lo sviluppa-

tore abbia creato un applicativo pensato per questa tecnologia, che sfrutti al suo interno la visione stereoscopica in modo tale da rendere la percezione della profondità. L'immagine unica elaborata dal cervello in realtà viene formata da due immagini leggermente sfalsate generate dal dispositivo, ognuna preposta a coprire un occhio specifico in maniera non molto distante da quanto accadrebbe nel mondo fisico. Gli smartphone compatibili con questo dispositivo sono dotati di un giroscopio molto sensibile che segue i movimenti sui quattro assi della visuale (alto, basso, destra, sinistra) permettendo in questo modo di realizzare il tracciamento dei movimenti della testa in modo inerziale. Per quanto riguarda invece il tracking della posizione, questo è delegato ad altri dispositivi in quanto il Cardboard non lo prevede. Lo spostamento all'interno dello spazio virtuale dovrà essere quindi svolto mediante supporti come joystick e gamepad compatibili con lo smartphone utilizzato, oppure utilizzando degli escamotage a livello applicativo.

Figura 3.1: Come si presenta l'immagine sullo schermo dello smartphone

3.4 Problematiche della realtà virtuale

Costruire delle esperienze usufruibili dall'utenza all'interno di un mondo virtuale è molto differente dal modellare delle tradizionali applicazioni in 2 dimensioni (2D). A differenza di queste ultime, nel creare un mondo VR vi sono svariati aspetti della cognizione umana che vanno considerati proprio a causa dell'immersione dell'utente all'interno dell'esperienza che precedentemente potevano essere in parte trascurati.

Vi sono due specifiche problematiche che vanno affrontate e sono:

1. il motion sickness (o chinetosi);

2. la semplicità e la famigliarità delle interazioni.

3.5 Il motion sickness

I sintomi del motion sickness possono presentarsi durante un viaggio all'interno di qualsiasi mezzo di locomozione. A differenza di molti passeggeri d'auto che subiscono effetti di nausea se non guardano fuori dal finestrino dell'autoveicolo il pilota difficilmente ne soffre, poiché può anticipare la sensazione di movimento prima che questa avvenga. Il motion sickness (o chinetosi) non è altro che un disturbo neurologico causato dalla disparità tra quello che una persona percepisce e quello che si aspetta di percepire.

L'evoluzione ci da indizi sul perché questa disparità esista: per esempio la nausea indotta dall'ingerimento di cibo velenoso ha difeso i nostri antenati tramite la rimozione del veleno prima che questo potesse portarli alla morte. Quando si riceve un input dal sistema vestibolare che differisce dal sistema visivo allora si percepisce la nausea. Questo meccanismo difensivo si è evoluto come una capacità di sopravvivenza, nonostante nella società moderna costituisca più una fonte di disturbo che un beneficio.

3.6 Come evitare il motion sickness

Gli sviluppatori e i designer di applicativi per la realtà virtuale hanno la responsabilità di seguire alcune linee guida per ridurre le possibilità di sviluppo di motion sickness da parte dell'utenza. Di seguito sono riportate le linee guida per evitare questo fenomeno fornite da Google [52].

- **Mantenere sempre l'head tracking**: fermare il tracciamento dell'utente anche solo per un brevissimo lasso di tempo potrebbe causargli la nausea. E' la più importante regola nel creare applicazioni di realtà virtuale.

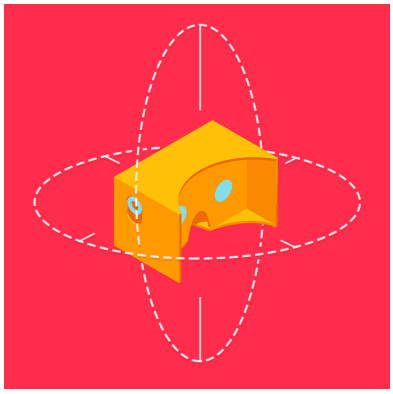

Figura 3.2: Immagine esplicativa dell'head tracking con Google Cardboard

- **Renderizzare immagini 2D nello spazio 3D**: mostrare alla vista dell'utente delle schermate fisse e disabilitare l'head tracking può causare dei disagi nell'utente stesso. Viene invece suggerito di renderizzarle senza bloccare il tracciamento e lasciando all'utente uno o più gradi di libertà nel movimento della testa (rotazione, inclinazione e imbardata) per evitare il verificarsi della nausea. Nel caso in cui si presentassero dei blocchi non intenzionali nel tracking (per esempio dutante caricamenti o carichi computazionali molto elevati) è sempre preferibile oscurare lo schermo o lasciare uno sfondo monocromatico di colorazione variabile con del feedback audio così da evitare problemi di motion sickness causati dalle interruzioni ed allo stesso tempo far capire all'utente che l'applicativo è ancora in esecuzione.

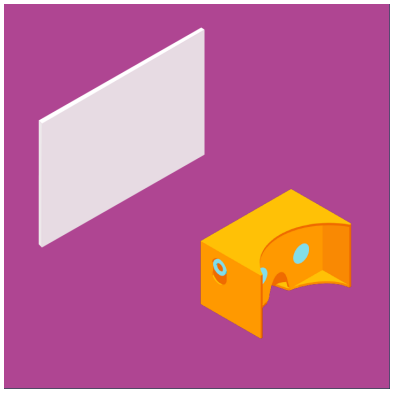

Figura 3.3: Immagine esplicativa di una schermata fissa con head tracking disabilitato

- **Evitare qualsiasi tipologia di blocco nel tracciamento della testa**: per quanto accennato nel paragrafo precedente è di estrema impor-

tanza che l'utente non si senta un "passeggero" all'interno dell'applicazione ma abbia il pieno controllo del movimento in maniera tale da poter attivamente prevedere ciò che sta per percepire. Esistono delle eccezioni a questa linea guida come le applicazioni di montagne russe, anche se non tutti gli utenti riescono ad affrontarle senza nausea. Conviene sempre lasciare attiva una qualsiasi tipologia di movimento all'utente.

- **Effettuare movimenti a velocità costante**: nella vita reale noi sentiamo accelerazione e decelerazione ma non percepiamo la velocità: ad esempio noi non percepiamo la velocità costante quando siamo all'interno di un'auto ma ne rileviamo i cambiamenti. Purtroppo questo non avviene all'interno del mondo virtuale e quando l'utente accelera o decelera ne risulta una disparità tra quello che il nostro corpo percepisce e quello che si aspetterebbe di percepire. Questa disparità può causare la nausea e può essere ridotta cercando di mantenere una velocità costante per l'utente quando questi si muove all'interno dell'applicazione.

- **Replicare l'assetto dell'utente del mondo reale nel mondo virtuale**: se l'utente è seduto mentre utilizza un'applicazione VR allora posizionarlo in un ambiente virtuale in cui sta fermo nella stessa posizione lo aiuterà a riconciliare i movimenti della realtà virtuale con quelli della realtà fisica. Egli può essere seduto anche all'interno di oggetti in movimento (abitacoli per esempio) ma va tenuto conto che se oggetti di grosse dimensioni sono in movimento di fianco a lui potrebbe verificarsi nausea dovuta ad una percezione illusoria di movimento creata da questi oggetti.

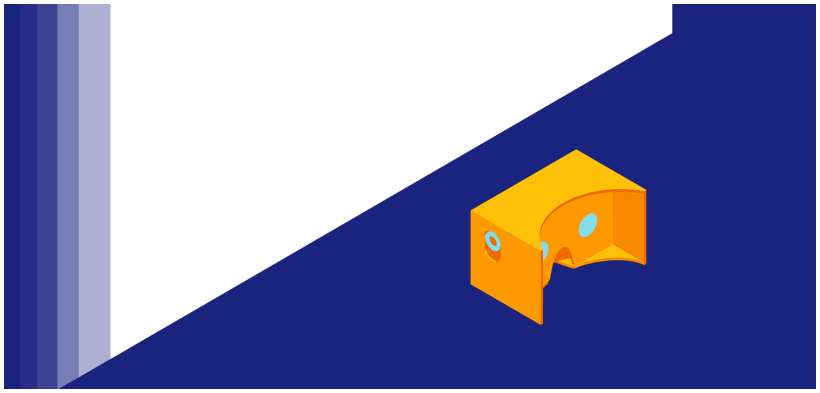

Figura 3.4: Immagine esplicativa del movimento di un grande oggetto vicino all'utente

- **Evitare cambiamenti di luminosità molto accentuati**: effettuare dei cambi di luminosità molto accentuati viene sconsigliato poiché con un display a breve distanza dagli occhi una transizione da oscurità a luce può causare sconforto fino a quando non ci si abitua alla nuova luminosità, esattamente come accade nel mondo reale.

3.7 Semplicità e famigliarità nelle interazioni

Trattandosi di un nuovo medium, quello della VR, l'utente potrebbe non avere ancora famigliarità con le interazioni all'interno del mondo virtuale. Guidare l'attenzione dell'utente in questo tipo di applicazioni presenta delle sfide uniche non presenti nelle applicazioni in due dimensioni.

Di seguito sono illustrati alcuni pattern di successo e come evitare errori comuni nelle interfacce VR che confondono l'utente.

3.7.1 Adattare l'esperienza all'utente

Abituarsi alla VR potrebbe richiedere tempo. L'utente potrebbe per esempio dover regolare un headset oppure inserire il telefono in un dispositivo simile al Cardboard. E' consigliato fare in modo che l'esperienza inizi solo quando l'utente indicherà di essere pronto, per questo motivo conviene sempre far cominciare l'esperienza con un input da parte dell'utente.

3.7.2 Intuitività dei controlli

Quando viene lanciata un'applicazione i comandi dell'interfaccia grafica devono comparire all'interno del campo visivo dell'utente poiché se ciò non accadesse egli potrebbe confondersi o rimanere disorientato cercando di capire i controlli all'interno dell'esperienza. Nel caso in cui l'app preveda il movimento è quindi consigliato aggiornare la posizione dei controlli dell'interfaccia affinché risultino sempre nel campo visivo dell'utente. Dispositivi come Cardboard presentano dei pulsanti sul dispositivo per cliccare sugli obiettivi ma nel momento in cui si debba rendere necessario creare dei controlli più sofisticati risulta molto difficile implementarli data l'assenza di altre forme di input. Per permettere ciò si possono creare dei pulsanti che permettono di fondere svariate funzioni e che cambiano comportamento in base al tempo in cui vengono evidenziati o premuti; tuttavia in certe situazioni possono risultare frustranti per l'utente poiché deve obbligatoriamente "perdere" del tempo per far sì che questi si attivino. Nel momento in cui si decida di implementare tali tipologie di soluzioni conviene sempre accompagnare il pulsante ad un timer sul qua-

le mostrare il tempo di attivazione al partecipante in maniera tale che possa comprendere cosa sta accadendo.

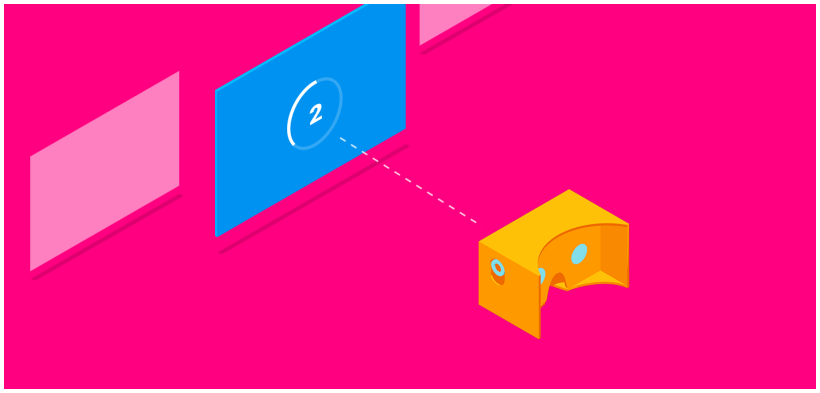

Figura 3.5: Immagine esplicativa del cambiamaneto di funzione di un elemento dell'interfaccia grafica

3.7.3 Utilizzare i feedback

All'interno delle esperienze di realtà virtuale è altamente consigliato utilizzare l'audio per comunicare le informazioni all'utente in modo tale da non generare un'alta mole di testi che in certe situazioni potrebbero risultare difficoltosi da leggere. Per rendere ancora più immersiva l'esperienza è sempre altamente consigliato inserire delle musiche di sottofondo accompagnate da effetti sonori per guidare il giocatore nelle aree di interesse dell'applicazione. Google consiglia anche di utilizzare il feedback aptico degli smartphone per migliorare l'esperienza dell'utente all'interno dell'esperienza VR. Per esempio eventi come il tocco di un oggetto o l'interazione con controlli da parte del partecipante possono beneficiare tantissimo di questo responso garantendo un livello maggiore di immedesimazione con il mondo virtuale.

Figura 3.6: Immagine esplicativa dei feedback audio

3.7.4 Mostrare un reticolo di puntamento

Molto frequentemente potrebbe succedere che l'utente decida di guardare accuratamente dei particolari oggetti, di qualsiasi dimensione, posti all'interno del mondo virtuale. Questa operazione avviene semplicemente tramite il movimento del suo sguardo ma potrebbe risultare estremamente difficoltosa senza alcun tipo di supporto nella fase di puntamento. Per semplificargli il compito sarebbe conveniente fornire all'utente un cursore per puntare gli obiettivi oppure mostrargli un reticolo quando sta cercando di mirare accuratamente. Nel caso vi fosse l'esigenza di mantenere un alto livello di realismo nella propria applicazione si potrebbero usare dei piccoli escamotage come l'utilizzo di un fascio di luce per indicare gli oggetti con i quali l'utente potrebbe interagire oppure mostrare il reticolo solo sugli oggetti con cui questi può effettivamente realizzare un'interazione. Il reticolo dovrà essere renderizzato in maniera stereoscopica (creando l'illusione della profondità) così che si proietti spazialmente sugli oggetti. Questi potrà poi aumentare o diminuire le proprie dimensioni con la profondità del movimento oppure rimanere di una grandezza fissa così che risulti più semplice da vedere per tutta la durata dell'esperienza.

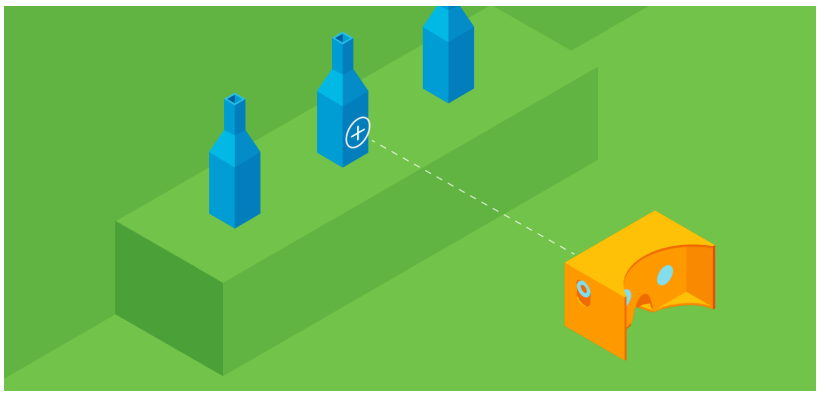

Figura 3.7: Immagine esplicativa del puntamento con mirino

Capitolo 4

Ambienti di sviluppo e development kit per Google Cardboard

In questo capitolo vengono illustrati alcuni degli ambienti di sviluppo e il Software Development Kit (SDK) fornito da Google per la realizzazione di applicazioni di realtà virtuale.

4.1 Ambienti di sviluppo

Esistono svariati ambienti di sviluppo, gratuiti e non, che permettono la realizzazione di applicazioni di realtà virtuale. Questi vengono chiamati engine e permettono di modellare esperienze di qualsiasi tipologia, dai simulatori ai videogiochi, fornendo un supporto nativo o tramite plugin, dipendente dalla tipologia di visore utilizzata, allo sviluppo di applicazioni di realtà virtuale. Tra questi vi sono Lamberyard [53], CooperCube [54], CryEngine [55], Fuzor [56], Autodesk Stingray [57], Unity3D [58], Unreal Engine [59], solo per citare i più conosciuti. Alcuni di questi engine sono reperibili in forma totalmente gratuita mentre altri debbono essere acquistati, alcuni non prevedono il pagamento di royalties al rilascio dell'applicazione sviluppata mentre altri solo al superamento di una determinata quota di vendita. Gli ambienti di sviluppo supportanti Google VR, tramite l'utilizzo del relativo SDK, sono quattro: iOS, Android, Unreal Engine e Unity. Le possibilità di scelta risultano quindi molto varie anche se i due engine più utilizati per la realizzazione di esperienze VR risultano essere Unity ed Unreal Engine, entrambi disponibili in forma gratuita.

4.1.1 Unreal Engine

Unreal Engine è un motore grafico sviluppato da *Epic Games* [60]. La prima versione è stata realizzata nel 1998 per il gioco Unreal pubblicato su Mac OS, Linux e Microsoft Windows. Nel susseguirsi degli anni ne sono state realizzate svariate versioni che andavano a migliorare le potenzialità del software sugli hardware correnti e ad estendere il numero di piattaforme supportate. Dal 2015 Unreal 4 è disponibile gratuitamente e prevede un versamento di una royalty pari al 5% sul reddito lordo solo nel caso in cui il prodotto sviluppato incassi più di 3000$ dalla sua immissione sul mercato. Grazie a questa caratteristica questo engine è tornato ad essere molto utilizzato anche dalle software house più piccole poiché permette di avere un motore versatile e molto performante, soprattutto a livello grafico, ad un costo irrisorio e dipendente solo dal successo del proprio applicativo. La versione corrente di Unreal Engine è la 4.16 [65] uscita a maggio 2017 e offre molte migliorie, tra le altre cose, ad alcune funzionalità utilizzate per realizzare applicazioni di realtà virtuale.

4.1.2 Unity

Unity è un ambiente di sviluppo prodotto da *Unity Technologies* [61] che permette di creare videogiochi 3D/2D multipiattaforma, visualizzazioni architettoniche, animazioni 3D e molti altri contenuti interattivi. Inizialmente era stato annunciato solo per sistemi operativi Macintosh, OS X, alla Worldwide Developers Conference di Apple del 2005 e da allora ha ampliato il numero di piattaforme supportate, portandolo a 27. Fino ad oggi sono state rilasciate 5 versioni principali del software e dal 2016 Unity Tecnologies ha deciso di cambiare il sistema di numerazione per le versioni beta delle release portandolo a una numerazione annuale, così da allinearle con il loro rilascio frequente. La versione corrente è la 5.6.1 rilasciata l'11 maggio del 2017. Il software è totalmente gratuito nella sua versione *Personal* e permette una monetizzazione massima di 100000$ annuali prima che sia obbligatoria una sottoscrizione mensile per una licenza *Plus* o *Pro*.

4.2 Unity come ambiente di sviluppo

Unity è un software che vanta una notevole semplicità d'approccio allo sviluppo rispetto a molti altri applicativi della stessa tipologia. L'interfaccia grafica permette in pochissimi minuti di creare qualcosa di funzionante con estrema intuitività e la nutrita community online, ad esso associata, è sempre molto attiva nell'aiutare gli utenti a risolvere problematiche di qualsiasi tipologia. Tali caratteristiche, unite allo sviluppo nativo multipiattaforma, hanno

portato questo software ad essere uno dei più utilizzati per la creazione di applicazioni interattive. Nei prossimi paragrafi verranno illustrati svariati aspetti di questo ambiente di sviluppo.

4.2.1 Principali caratteristiche

La più grande peculiarità di Unity è sicuramente quella di essere un ambiente di sviluppo multipiattaforma, quindi con cui è possibile creare applicazioni che possono funzionare su svariate piattaforme. Tra queste vi sono: Mac OS X [62], Linux [63], Microsoft Windows [64], Android [65], iOS [66], Adobe Flash Player [67], Nintendo WiiU [68], Nintendo Switch [69], Nintendo 3DS [70], la famiglia Microsoft Xbox One [71], Xbox 360, Sony PlayStation 3, Sony PlayStation 4, e svariate altre. Un'altra caratteristiche che l'ha reso famoso è l'integrazione nativa con tutti i software di grafica più utilizzati, come per esempio: Blender, Maya, Modo, 3DStudioMax, Photoshop, grazie alla quale è possibile importare modelli 3D realizzati con questi software.

Unity supporta tre linguaggi di programmazione per script: C#, JavaScript e Boo.

Questo ambiente di sviluppo utilizza librerie diverse in base all'ambiente di lavoro:

- le DirectX sono utilizzate esclusivamente in ambiente Windows;
- le Open GL in ambiente Mac OS X, Linux e Windows;
- Open GL ES in iOS e Android.

Unity fornisce inoltre la possibilità di recuperare ogni tipologia di asset, anche in forma gratuita, direttamente dall'interno dell'applicazione tramite un negozio denominato *Asset Store* (illustrato successivamente). Quelle appena elencate sono anche le motivazioni che mi hanno portato a scegliere Unity come engine ed ambiente di sviluppo per l'elaborato allegato a questa tesi.

Un altro aspetto molto importante da considerare quando si utilizza questo ambiente di sviluppo è la programmazione *multi-thread*. Unity supporta la concorrenza tra thread ma in maniera molto particolare. Il software consente di istanziare thread *Worker* per alleggerire il carico computazionale a quello principale (*MainThread*) ma a una sola condizione, ovvero che questi non utilizzino le API di Unity, poiché non Thread Safe, o che non interagiscano con l'ambiente di sviluppo. Per impedire le chiamate delle API il software implementa un *Thread-Check* atto ad impedire e bloccare l'esecuzione di chiamate da thread differenti dal MainThread. Questa è una grossa limitazione

del software che porta a lavorare principalmente con un solo thread lasciando comunque la possibilità di alleggerimento per le operazioni di calcolo più complesse di valori.

4.2.2 Design pattern

Le fondamenta della struttura di Unity sono basate su un pattern: il *Component Pattern*. Esso definisce ogni GameObject all'interno dell'ambiente di sviluppo come composizione di *Components* (componenti), ognuno con il proprio dominio, separanti le differenti funzioni. In questo modo le entità vengono ricondotte a dei semplici contenitori di componenti [72].

Altro pattern implementato direttamente all'interno di questo ambiente di sviluppo è il *Prototype Pattern* con cui è possibile clonare degli oggetti all'interno della nostra applicazione da un elemento di partenza chiamato Prototipo. In Unity questa funzione di prototipo la forniscono i *Prefab* (affrontati più approfonditamente nei paragrafi seguenti) che possono essere istanziati più e più volte, risparmiando delle istanziazioni statiche all'interno dell'applicazione. Anche il *Command Pattern* è integrato a livello di ambiente di sviluppo tramite l'assegnazione dei vari comandi di gioco nelle impostazioni degli Input.

Altri due pattern implementati direttamente all'interno della struttura di Unity sono il *GameLoop Pattern* e l'*Update Method Pattern*. Il primo implementa un ciclo di gioco (Game Loop) che viene eseguito in maniera continua durante l'esecuzione dell'applicazione e ad ogni ripetizione di questo il programma processa gli input senza bloccarsi, aggiorna lo stato interno dell'applicazione e successivamente renderizza ciò che deve essere visto a schermo, tracciando il tempo impiegato per mantenere un FrameRate costante. L'Update Method Pattern invece implementa, all'interno di ogni singolo oggetto facente parte dell'applicazione, un metodo chiamato Update in cui ognuno di questi simula il proprio comportamento all'interno del frame corrente. Ad ognuno di questi frame l'applicazione aggiorna tutti gli elementi che la compongono chiamando il metodo update su ciascun elemento. Il Game Loop non è visibile durante la stesura del codice ma è implementato nelle API di Unity mentre l'Update Method viene creato alla generazione di ogni script (anche se un oggetto può non utilizzarlo).

Unity permette anche la gestione di eventi direttamente all'interno dell'ambiente di sviluppo permettendo di disaccoppiare un messaggio o l'invio di un evento dal momento in cui questo si aziona a quello in cui viene processato. Questo tipo di soluzione è chiamata *Event Queue Pattern*. Ne è un esempio la funzione *SendMessage* presente all'interno della classe GameObject (da cui si origina ogni altra classe) con cui si invia un messaggio di esecuzione di un metodo presente in un'altro oggetto, disaccoppiando la chiamata della funzione

dal momento in cui l'oggetto ricevente la processa. Questo pattern può essere implementato anche nella gestione delle interfacce grafiche utente. Nella nuova versione introdotta dalla versione 5 del software gli eventi possono essere generati direttamente dal componente *Event Trigger* all'interno dell'oggetto UI e catturati, o controllati, da degli script Osservatori (integrazione dell'*Observer Pattern*).

Un altro pattern integrato nell'ambiente di sviluppo è lo *State Pattern*. Questi permette a un oggetto di alterare il proprio comportamento, anche in modo totalmente differente, al cambiamento del proprio stato interno. L'introduzione delle *Finished State Machine* avviene in Unity, per esempio, tutte le volte che si implementa un Animator all'interno di un oggetto. Il proprio comportamento a livello di animazione cambia durante la transizione da uno stato all'altro. Questo pattern può essere implementato in qualsiasi oggetto anche a livello di script. Un altro pattern integrato, e implementabile in certe situazioni per migliorare l'efficienza, è l'*Object Pool Pattern* (a sua volta esempio di *Flyweight Pattern*) con cui è possibile gestire numerose istanziazioni e distruzioni di oggetti senza consumo eccessivo di memoria. All'interno di Unity viene seguita una logica molto simile a quella dell'Object Pool Pattern quando a più oggetti sono assegnate le medesime texture e mesh, queste tipologie di componenti vengono allocate in memoria una sola volta e poi rimosse solo al momento dell'effettiva distruzione da parte del sistema. Un esempio in cui risulta molto utile implementare questo pattern si verifica nel momento in cui molti oggetti della stessa tipologia devono essere generati e distrutti conseguentemente in un breve periodo, come per gli effetti particellari per esempio. Con questo pattern il sistema mantiene allocata in memoria una parte degli oggetti distrutti definendoli semplicemente come inattivi e riattivandoli con le dovute caratteristiche al momento di una nuova istanziazione del medesimo oggetto. Questo pattern tuttavia presenta un difetto, nel momento in cui vengono istanziati molti oggetti nello stesso momento e successivamente vengono distrutti il sistema potrebbe mantenere allocati in memoria degli oggetti che non riutilizzerà per diverso tempo sprecandone una parte, quando invece il pattern nasce per lo scopo opposto e per evitare la frammentazione. Nel momento in cui lo si utilizza per questa funzione bisogna essere molto attenti a gestire in maniera corretta il numero adeguato di elementi.

Unity permette di implementare molti altri pattern, anche in maniera personalizzata, ma devono spesso essere rimodellati per non entrare in collisione con la struttura dell'architettura del software.

4.2.3 L'interfaccia grafica di Unity

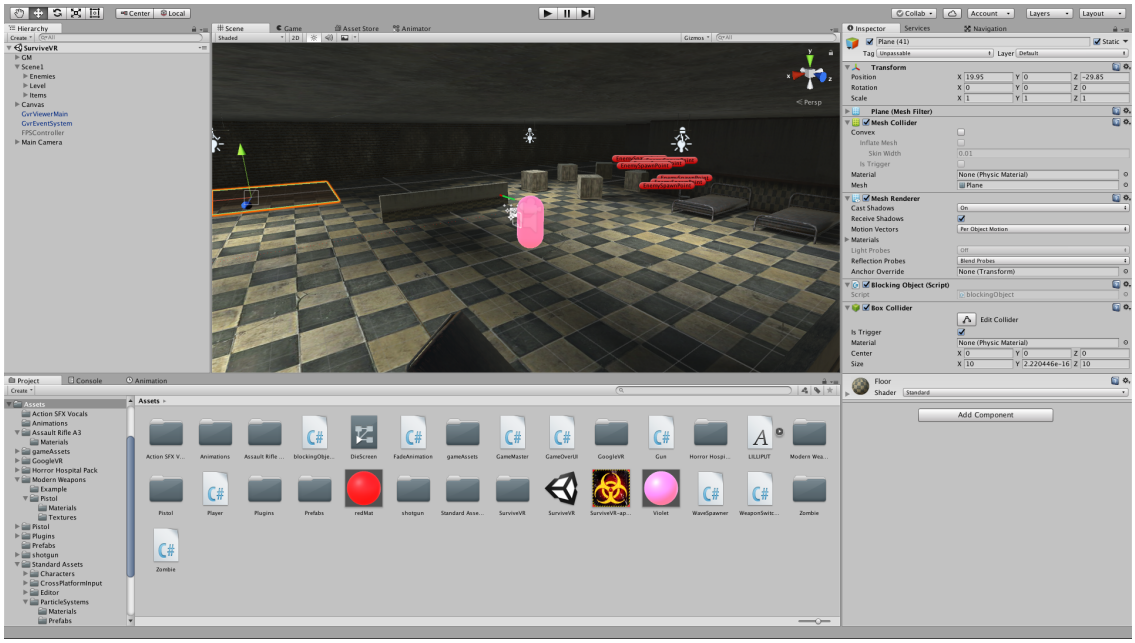

Figura 4.1: Finestra di lavoro di Unity

Come mostrato sopra nell'immagine 4.1 l'interfaccia di Unity è suddivisa in svariate aree:

- **Hierarchy** : questa area mostra tutti gli elementi che compongono il gioco all'interno della scena corrente. Ogni oggetto può essere padre o figlio, creando quindi una relazione gerarchica in cui il figlio eredita caratteristiche e componenti del padre. Questa relazione si crea trascinando un oggetto su un altro, relativamente il primo diviene figlio e il secondo il padre, in maniera molto intuitiva. Qui possono anche essere inseriti nuovi elementi di qualsiasi tipo (si va da modelli 3D/2D, a elementi dell'interfaccia grafica per esempio).

- **Scene** : in questa zona viene mostrata in tempo reale la scena e si possono posizionare, ruotare ed aggiungere nuovi oggetti. Trascinando oggetti nella schermata è possibile inserirli direttamente nella posizione desiderata.

- **Game** : in questa finestra viene mostrata la scena inquadrata dalla telecamera principale selezionata. In questo modo è possibile vedere in tempo reale la scena dall'inquadratura di gioco.

- **Asset Store** : finestra tramite cui è possibile raggiungere il negozio degli asset di Unity. Da questo store è possibile scaricare elementi ed utilizzarli immediatamente all'interno del proprio progetto. Gli elementi scaricabili possono essere di qualsiasi natura, si può andare da elementi audio a modelli grafici più o meno complessi per esempio.

- **Project** : in questa zona sono mostrate tutte le cartelle e i file all'interno del progetto. E' possibile navigare esattamente come nel Finder di OS X, nell'Explorer di Windows o nel Nautilus di Ubuntu (Linux). E' anche possibile creare degli elementi prefabbricati che possono poi essere riutilizzati all'interno del progetto, in inglese prefabs, trascinando gli oggetti dall'area Hierarchy a quella Project. In questo modo è possibile creare dei modelli prefabbricati di qualsiasi elemento del progetto.

- **Inspector** : nell'Inspector sono contenute tutte le informazioni relative all'oggetto selezionato. Qui sono mostrati i componenti (components) che caratterizzano tale oggetto o il suo comportamento.

- **Console** : come in molti altri software di sviluppo in quest'area sono riportati i messaggi di errore, di warning e di stampa dell'applicazione.

4.2.4 I progetti in Unity

Tutto ciò che è necessario per il funzionamento di un'applicazione sviluppata con Unity deve essere situato all'interno dell'area Project e quindi contenuto all'interno della cartella Assets del progetto in questione. All'interno di questa vanno salvate tutte le animazioni, gli script, gli effetti audio, le musiche, i materiali, i prefabbricati, ecc. Tutte le scene sono composte da GameObject [73], di questi fanno parte tutte le tipologie di oggetti appartenenti alla suddetta scena come personaggi, telecamere, figure dell'ambiente, ecc. Gli oggetti di questa tipologia inseriti all'interno del "Project" vengono detti prefabs [74] (prefabbricati) e sono degli oggetti riutilizzabili più volte all'interno del progetto, da cui il nome prefab. Ogni volta che un oggetto di questo tipo viene inserito in una scena si dice che questo è un'istanza del prefab stesso, quindi ogni modifica apportata all'oggetto originale/padre si ripercuote su ogni singola istanza figlia. Ogni GameObject facente parte del progetto ha delle proprietà visibili all'interno dell'*Inspector*, una volta che questi è selezionato, disposto nell'area di lavoro sulla destra nella schermata principale del software. Tali caratteristiche sono definite come *component* [75]. Esistono tantissimi tipologie di componenti che variano in base alla caratteristica implementata. Alcuni esempi sono:

- **Transform** : componente che determina la posizione nello spazio 3D all'interno della scena corrente;

- **Script** : componente che inserisce all'interno dell'attuale GameObject uno script per implementare nuove funzioni o comportamenti non definiti;

- **Collider** : componente che permette di gestire le collisioni tra il GameObject corrente e tutti gli altri presenti in scena;

- **Animation** : componente che associa un'animazione al GameObject corrente;

- **Animator** : componente che associa un animatore al GameObject corrente. Un animatore è composto da una sequenza data da una macchina a stati finiti che gestisce il susseguirsi delle animazioni o eventi sull'oggetto stesso.

- **Audio Source** : componente che associa una risorsa audio a un GameObject.

4.2.5 Le interfacce grafiche

Come è stato accennato nel capitolo precedente le interfacce grafiche giocano un ruolo importantissimo all'interno delle applicazioni VR poiché consentono all'utente di sperimentare le prime interazioni nel mondo virtuale. Queste dovranno essere realizzate all'interno dello spazio 3D, quindi dovranno fornire un senso di profondità all'utente, in maniera tale da essere facilmente individuabili e permettere interazioni intuitive e naturali per il partecipante.

Unity mette a disposizione degli sviluppatori un sistema per la realizzazione di interface grafiche utente, *User Interfaces* o *UI*, da inserire all'interno delle applicazioni. Esso costituisce un insieme di strumenti molto flessibili che permettono la realizzazione di animazioni ed effetti sulle interfacce con estrema semplicità. Successivamente sono illustrati nel dettaglio tutti gli strumenti utilizzati all'interno dell'elaborato presentato nell'ultimo capitolo:

- **Canvas** [76]: questo strumento può essere identificato come il contenitore degli elementi che compongono l'interfaccia utente. Questi non è altro che un GameObject con componente Canvas e tutti gli oggetti inseriti all'interno di esso, quindi facenti parte dell'UI, ne sono figli. Il Canvas non è altro che un elemento rettangolare nel quale vengono disegnati tutti gli elementi dell'interfaccia utente ad esso associati in un ordine

ben specifico, l'ultimo elemento figlio viene disegnato per ultimo (quindi rimarrà sovraesposto a quello precedente) e a ritroso tutti gli altri. Questo strumento possiede tre differenti modalità di rendering: screen overlay, screen space camera e world space. Queste non fanno altro che definire il comportamento di questo oggetto all'interno dell'applicazione. Se un Canvas è in modalità screen overlay pone tutti gli elementi ad esso associati sullo schermo renderizzato sopra la scena, adattando la propria risoluzione e dimensione in base ad esso. In modalità screen space camera il Canvas funziona in maniera molto simile a quella precedente ma è posizionato ad una distanza specifica davanti alla telecamera selezionata con un aspetto variabile in base alle impostazioni della suddetta telecamera. In modalità world space invece il Canvas si comporta come un qualsiasi oggetto della scena, con dimensioni e proprietà definite, ed è renderizzato all'interno dello spazio 3D come qualsiasi altro elemento. Questo è utile per creare interfacce posizionate sopra ad elementi specifici o per guidare l'utente all'interno del mondo virtuale.

- **Button** [77] : è il componente che permette di rendere interattiva l'interfaccia e non fa altro che inserire all'interno di un oggetto o del Canvas stesso un pulsante, su cui Unity prevede la possibilità di generazione di eventi, ad esempio quando questi è premuto (OnClick Listener)

- **Text** [78] : è un componente che può essere inserito nel Canvas o in un suo qualsiasi figlio. Questo contiene un'area di testo che può essere formattata all'interno del componente stesso.

- **Image** [79] : componente che se assegnato al Canvas o ad un suo figlio mostra un'immagine selezionata su di esso nella posizione specificata.

Gli strumenti sopra elencati sono solo alcuni di quelli presenti in Unity e come accennato si è ritenuto opportuno riportare solamente gli elementi utilizzati nell'elaborato allegato. Un elenco completo di tutti gli elementi dell'UI è consultabile nella documentazione di Unity (Unity Documentation) disponibile sul sito [80].

4.2.6 Gli script in Unity

Unity permette di generare ed utilizzare degli script scritti con tre linguaggi di programmazione: *Boo*, *C#* e *JavaScript*. In questa tesi sarà considerato C# poiché è il linguaggio di programmazione scelto per l'elaborato. Ad ogni GameObject presente all'interno del progetto possono essere associati uno o più script, che permettono di renderlo interattivo e di definirne un comportamento all'interno dell'esperienza. Questo è possibile inserendo dall'*Inspector*

il componente *Script*. Rendere un GameObject interattivo significa fare in modo che il suo comportamento muti in base al verificarsi di eventi previsti dallo sviluppatore. Essendo ogni script definito all'interno di uno script file, questo può essere riutilizzato anche all'interno di più progetti importandolo all'interno dell'area di lavoro corrente.

Nel momento in cui si crea uno script, Unity genera una classe derivante da *MonoBehaviour* [81] che contiene tutte le funzioni più utilizzate nella creazione di applicazioni. Di seguito viene illustrato un esempio di classe alla sua generazione con 2 funzioni utilizzate da svariati script dell'elaborato:

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

code.

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

public class NewScript : MonoBehaviour {

   // Use this for initialization
   void Start () {

   }

   // Update is called once per frame
   void Update () {

   }
}
```

Listato 4.1: Nuovo script MonoBehaviour

Due script possono comunicare in Unity se si presenta l'esigenza ed esistono più modi per realizzare questo scambio di messaggi. Può essere utilizzata una funzione chiamata *SendMessage("")* [82] che non fa altro che inviare un messaggio ad un GameObject specifico passandogli come parametro la stringa contenente il nome della funzione da richiamare. Se questa funzione non esiste all'interno di tale GameObject semplicemente non viene eseguita alcuna operazione. Per determinare a quale GameObject va inviato il messaggio è necessario utilizzare la funzione *Find("")* [83] che permette di trovare un GameObject in base al suo nome, contenuto nel testo della stringa passata come

parametro alla funzione, oppure nel caso in cui si voglia inviare un messaggio a più oggetti viene utilizzata la funzione *FindObjectWithTag("")* [84], con cui vengono individuati tutti gli oggetti che hanno come etichetta (*tag*) quella specificata nel testo della stringa passata alla funzione. L'etichetta in questo caso deve essere assegnata o creata agli oggetti all'interno dell'*Inspector*.

SCRIPT_A.cs — GameObject A

```
public class NewBehviourScript : MonoBehaviour{

    void Start() {}

    void Update{
        GameObject.Find("GameObjectB").SendMessage("Make");
    }
}
```

SRIPT_B.cs — GameObject B

```
public class NewBehaviourScript : MonoBehaviour{

    void Star(){}

    void Update(){}

    public void Make(){
        //    Conenuto della funzione Make
    }
}
```

Figura 4.2: Un esempio di comunicazione tra script in Unity

All'interno dell'installazione di Unity è presente una versione proprietaria di MonoDevelop [85] per permettere agli sviluppatori di lavorare e scrivere il codice senza utilizzare software di altro tipo, a parte per la modellazione grafica ed audio. Questo applicativo è un'implementazione opensource del .NET Framework [86] di Microsoft.

4.2.7 Le musiche e gli effetti audio

In Unity è possibile inserire dei suoni o delle musiche all'interno dell'applicazione semplicemente aggiungendo un componente chiamato *AudioSource* [87] all'interno di un GameObject qualsiasi. Grazie a questo è possibile associare uno o più file audio, ubicati all'interno della cartella *Assets* del progetto,

ad un oggetto qualsiasi che si occuperà di gestirne l'emissione del suono al momento giusto, che spesso avviene all'interno di script.

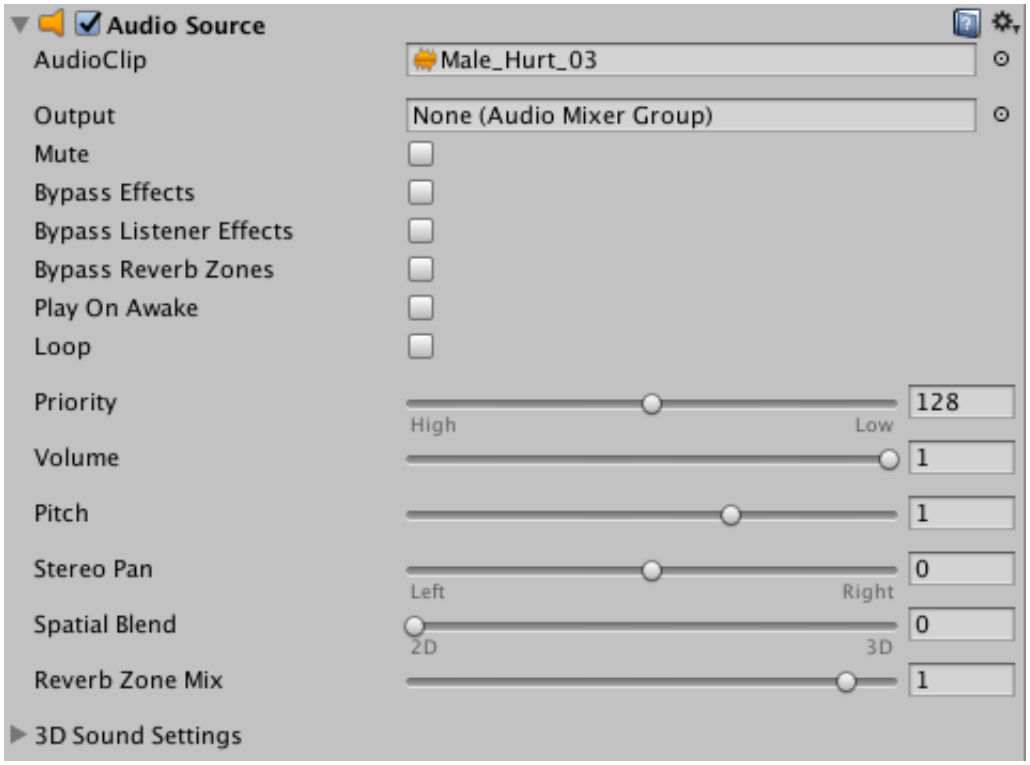

Figura 4.3: Area dell'Inspector inerente ad una sorgente audio

4.2.8 Le animazioni e gli Animator

Un'animazione in Unity è rappresentata da un *Animation Clip* [88] (clip di animazione) rappresentante un oggetto che memorizza tutte le proprietà di un GameObject nel tempo, quindi ad ogni *Frame*. Unity permette di creare delle *Animation Clip* che possono essere associate a svariati GameObject semplicemente inserendole come componenti al loro interno e gestendole tramite uno script. Le animazioni possono essere gestite anche tramite un componente *Animator* [89] che permette di modellare il susseguirsi di animazioni, azionate da qualsiasi tipologia di evento, tramite una *macchina a stati finiti*.

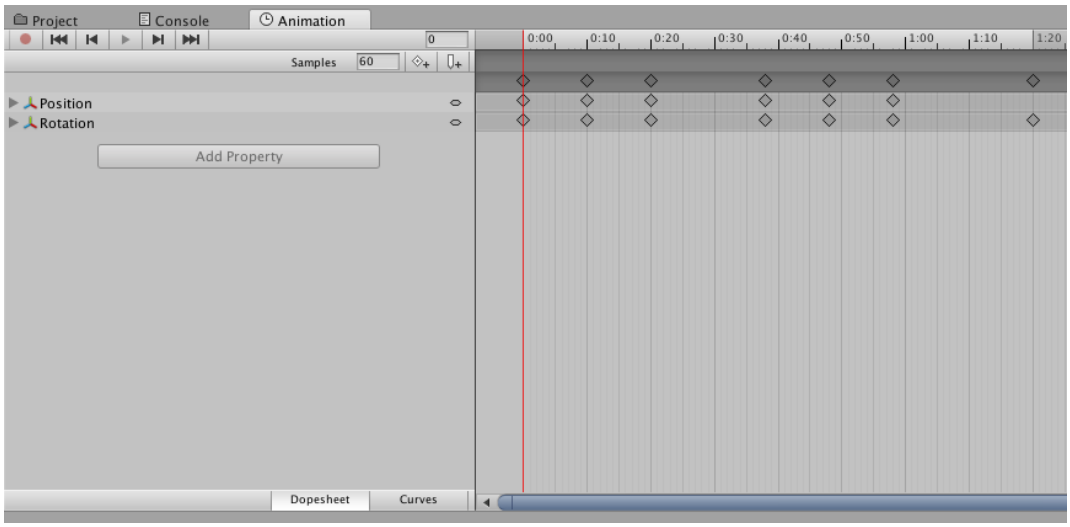

Figura 4.4: Animazione di un elemento composta da una traslazione e una rotazione

L'ambiente di sviluppo permette al suo interno di creare direttamente delle animation in un'area di lavoro ad essa dedicata chiamata *Animation*. Per creare una clip è sufficiente selezionare l'oggetto desiderato, posizionarsi con il puntatore del mouse sull'istante di tempo su cui si vuole modificare dei parametri e successivamente andare ad editare le caratteristiche desiderate direttamente all'interno del GameObject selezionato. Tutte le modifiche effettuate nel tempo formano un'animazione che può quindi comprendere qualsiasi tipologia di parametro all'interno della scena. Assegnando più animazioni allo stesso GameObject è possibile creare delle sequenze di queste dipendenti dallo stato in cui l'oggetto si trova in quell'istante, che si ripetano per un certo numero di volte e non si sovrappongano l'un l'altra. E' possibile gestire queste sequenze utilizzando un *Animator Component* che permette di associare un GameObject ad un *Animator Controller*. Un esempio di oggetto che deve seguire questo comportamento è un personaggio all'interno di un videogioco. Questo deve eseguire delle animazioni in base al movimento effettuato, quindi associa l'animazione corretta alla corsa, la camminata, il salto e l'attesa. Una volta creato, l'*Animator Controller* verrà salvato tra le risorse dell'attuale progetto.

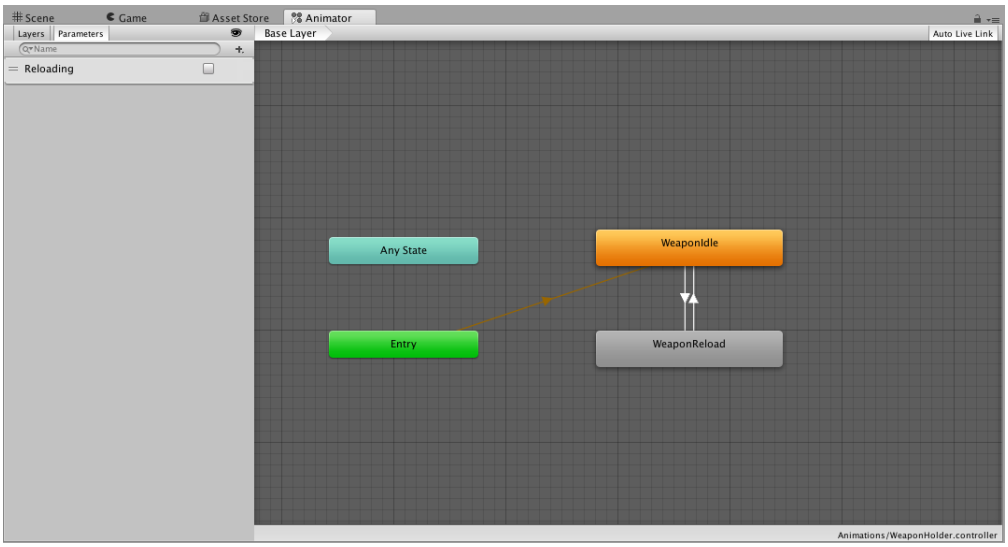

Figura 4.5: Esempio di machina a stati finiti di un Animator

4.2.9 Gli asset e l'Asset Store

Gli *asset* all'interno di Unity sono file di qualsiasi tipologia che posso risultare utili all'interno del progetto. Possono rappresentare per esempio suoni, musiche, modelli 3D, sprite 2D, texture, script, prefabs, scene realizzate con altri software di modellazione o shader. Tutto ciò che è situato all'interno delle cartelle del progetto può essere considerato un asset. Unity permette inoltre di importare nuovi asset da un negozio integrato all'interno del software, chiamato *Asset Store* [90], all'interno del quale è disponibile ogni tipologia di risorsa utile allo sviluppo di applicazioni. Questi asset possono essere creati e caricati dagli utenti che sono registrati sul sito di Unity e successivamente venduti al prezzo desiderato, ciò ha permesso di aumentare il numero di risorse disponibili in modo esponenziale nel corso degli anni fornendo agli sviluppatori un numero sempre crescente di modelli, audio, script, ecc. a pagamento o in forma gratuita.

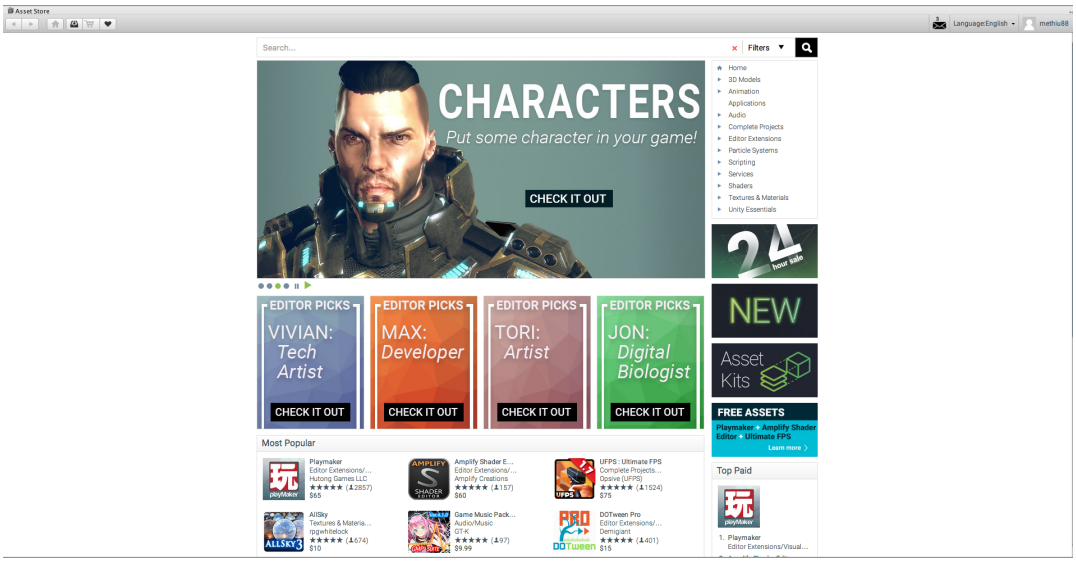

Figura 4.6: Asset Store all'interno di Unity

4.3 Il package Google VR SDK

Per utilizzare un qualsiasi visore in un progetto Unity è necessario *importare all'interno del proprio progetto* un package contenente gli asset necessari per il suo utilizzo. Il package da importare per utilizzare Google Cardboard viene fornito da Google sul sito ufficiale di *Google VR* insieme al Software Development Kit (*SDK*) di ogni sistema operativo (per dispositivi mobile) o per i due ambienti di sviluppo supportati [91].

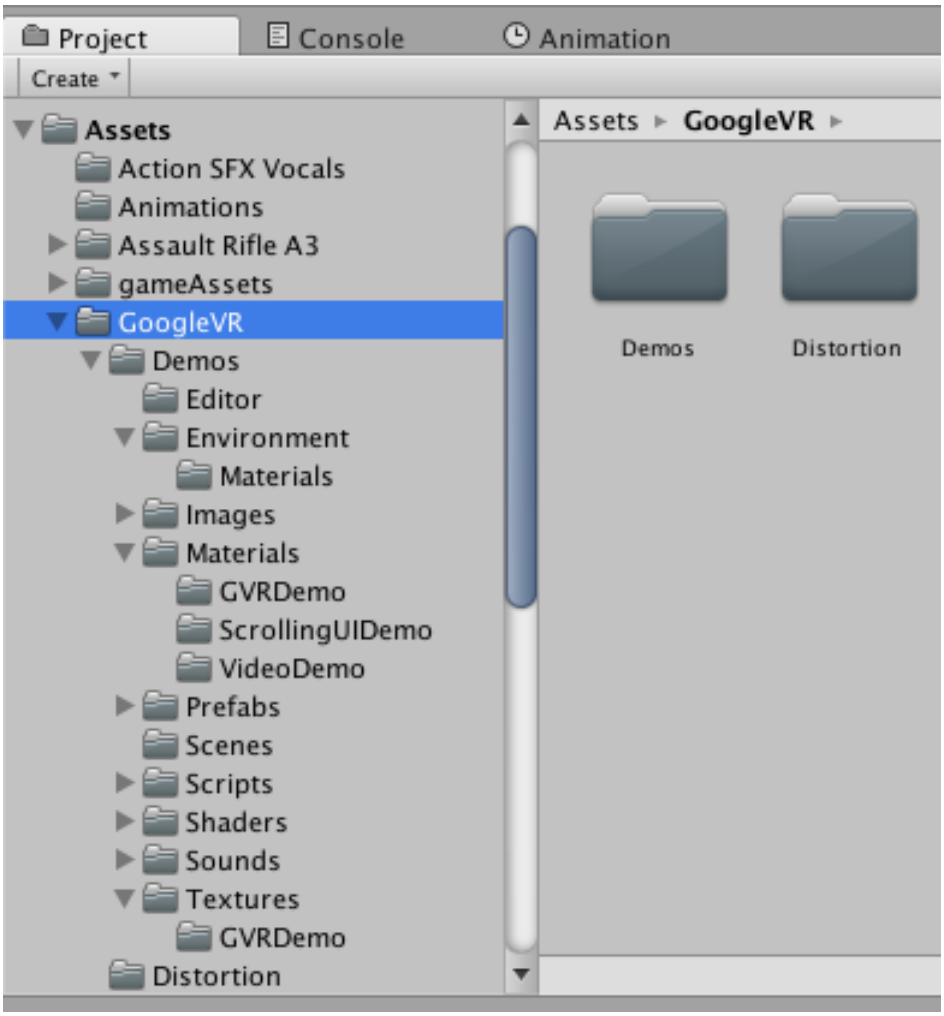

Figura 4.7: Package GoogleVR all'interno di Unity

All'interno di questo package sono contenuti tutti gli strumenti necessari per integrare i dispositivi Google Cardboard e Daydream View all'interno dell'applicativo. Tra questi spiccano prefab, script, texture, materiali, demo, ecc. Per poter implementare Cardboard è sufficiente inserire all'interno della scena del progetto il prefabbricato *GvrViewerMain* contenuto nella cartella Prefabs di *Google VR*. Questi implementa uno script che realizza le due telecamere preposte a mostrare agli occhi dell'utente le immagini del mondo virtuale, durante la sua esecuzione, nel momento in cui risulta abilitata la modalità VR dall'*Inspector* dell'oggetto. Le due immagini vengono mostrate dai figli *CameraMainLeft* e *CameraMainRight* di *GvrViewerMain* generati a runtime.

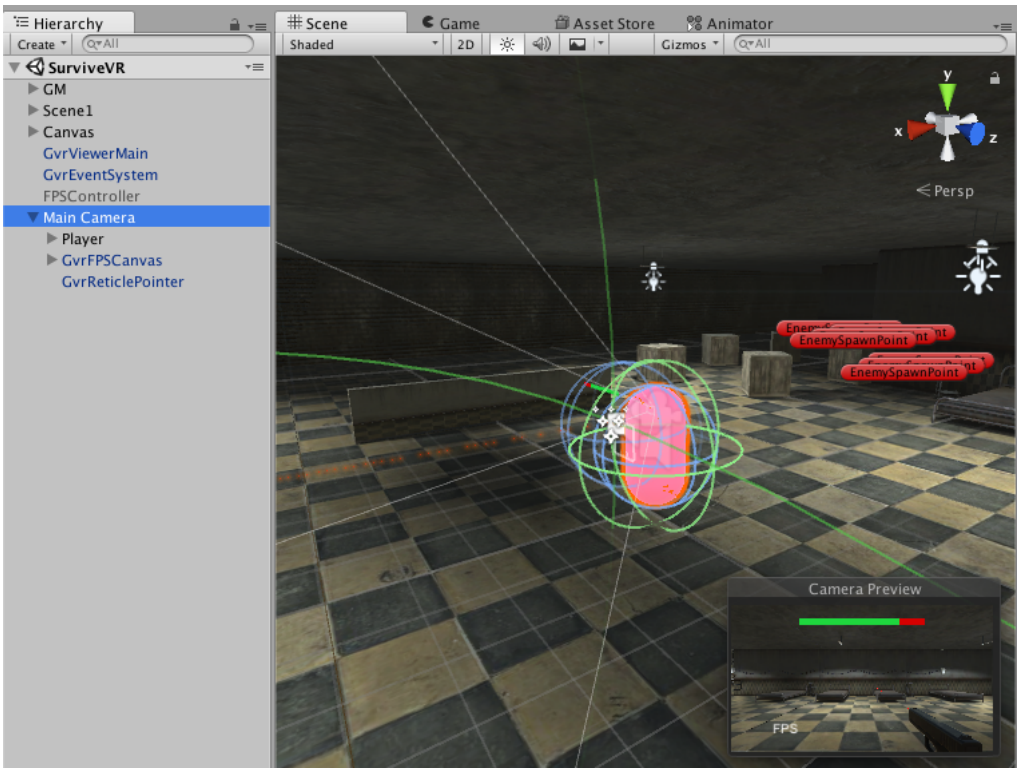

Figura 4.8: GvrMainViewer al'interno della scena

Capitolo 5

Un caso di studio: SurviveVR

Dopo aver definito la realtà virtuale, parlato dei suoi campi di sviluppo e di applicazione, elencati gli strumenti e le tecnologie che ci portano ad ottenerla, illustrato gli aspetti più rilevanti delle interazioni tra utente e mondo virtuale ed alcune delle problematiche più rilevanti da affrontare, si ritiene opportuno introdurre l'applicazione VR sviluppata a corredo di questa tesi.

L'elaborato, come precedentemente accennato, è stato realizzato con Unity, software dedicato allo sviluppo di videogiochi. E' stato scelto questo ambiente perché supporta nativamente lo sviluppo di applicazioni di realtà virtuale tramite l'importazione del SDK del visore utilizzato e per la sua diffusione, per cui è stato molto semplice trovare la documentazione necessaria per imparare ad utilizzarlo.

Al seguente link è possibile guardare e scaricare il codice sorgente relativo a tutti gli script realizzati per l'applicativo :

`https://github.com/methiu88/SurviveVR`.

5.1 Descrizione dell'applicazione

Uno degli obiettivi principali di questa tesi è di utilizzare la tecnologia Google Cardboard e il sistema Unity3D 5.5.1f1 per realizzare un'applicazione di realtà virtuale, valutando di conseguenza le possibilità e le limitazioni collegate alla piattaforma di sviluppo e al dispositivo.

L'applicazione prodotta, chiamata SurviveVR, non è altro che un prototipo in cui sono implementate alcune caratteristiche di interazione illustrate nel capitolo 3 e delle soluzioni atte ad evitare ogni tipo di problematica nel giocatore. L'esperienza inizia con la comparsa di una schermata principale mostrata nell'ambiente in 3 dimensioni. Qui vi sono spiegati i controlli, viene mostrato il miglior punteggio finora ottenuto ed è possibile cominciare la partita o uscire dal gioco cliccando su un apposita finestra. Una volta avviata la sessione di

gioco l'utente dovrà difendersi utilizzando delle armi da fuoco, in combinazione con il sistema di mira, dai nemici che cercheranno di colpirlo. Questi vengono generati ad *ondate*, ovvero arrivano a gruppi sempre più numerosi e alla loro morte lasceranno cadere oggetti utili al giocatore per sopravvivere il più a lungo possibile. Gli elementi principali nelle meccaniche dell'applicazione sono il puntamento ed il sistema di interazione con gli oggetti. Questi sono effettuati, rispettivamente, tramite lo sguardo del giocatore e il pulsante NFC situato sul Cardboard.

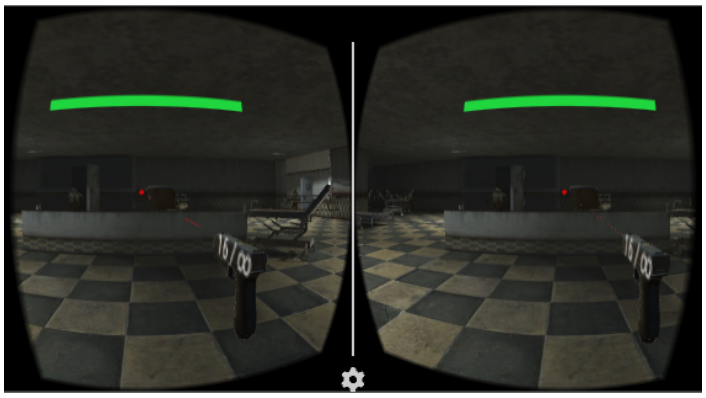

Figura 5.1: Applicazione in esecuzione sull'edtitor di Unity

5.2 Scelte di design nelle interazioni

La scelta attorno alla quale ruota tutta la progettazione è scaturita durante lo studio della tipologia di interazioni da implementare all'interno dell'applicativo. Essendo Cardboard V2 un dispositivo provvisto di un solo pulsante, tutte le interazioni presenti all'interno dell'app dovevano per forza scontrarsi con questa grossa limitazione durante la loro implementazione. Partendo da questo presupposto si è ritenuto opportuno rendere questa sua semplicità il punto di forza dell'applicativo. Da ciò è nata l'idea di creare tutto il software attorno al punto debole del Cardboard. La raccolta di oggetti, l'uccisione dei nemici e le interazioni con l'interfaccia grafica passano tutte attraverso la singola pressione del pulsante NFC, nonostante sia stata valutata l'idea di implementare la funzione di cambio arma con una pressione prolungata del bottone stesso. Questa è stata successivamente scartata poiché inficiava l'immersione all'interno dell'esperienza traviando le meccaniche di gioco stesse, dato che le interazioni con i nemici all'interno dell'applicativo ruotano attorno alle armi da fuoco. Di conseguenza ad una pressione prolungata del pulsante di fuoco l'utente si aspetterebbe di emanare una raffica di colpi e non di abilitare un'interfaccia grafica.

Un'altra scelta è stata quella di non permette il movimento del giocatore all'interno della scena per tre motivazioni:

1. Cardboard è provvisto solo di un pulsante per l'inserimento dei comandi;

2. sono stati studiate svariate tipologie di movimento, tra cui anche uno prefissato, ma sarebbero risultate molto confusionarie agli occhi del giocatore, a meno che non si fosse creato un FirstPersonShooter su binari. Per le meccaniche dell'app far correre il personaggio in una direzione e farlo guardare in quella opposta per colpire i nemici avrebbe potuto portare alla nausea gli utenti più sensibili per cui è stato evitato il movimento in assenza di un controllo aggiuntivo;

3. si voleva dare maggior rilevanza alle interazioni con gli oggetti e le interfacce grafiche all'interno del mondo virtuale.

E' stata valutata anche l'introduzione di un gamepad aggiuntivo per implementare i movimenti del giocatore ma a livello di mercato solo una piccolissima fetta di utenti usufruisce di tale supporto su dispositivi mobile.

Infine per tendere una mano all'utente sono stati aggiunti degli effetti sonori per restituirgli un feedback ad interazione avvenuta e dei piccoli elementi grafici per aiutarlo a comprendere dove si trovano gli elementi dell'interfacia grafica. Alcuni esempi sono il puntatore a schermo che cambia forma con gli elementi interagibili, le frecce che indicano la posizione delle varie tipologie di pannello nel menu del gioco e le indicazioni che appaiono alla fine della partita per indirizzare lo sguardo dell'utente verso il pannello di fine partita.

Tutte le interazioni all'interno di SurviveVR sono state studiate affinché fossero il più intuitive possibili.

5.3 Oggetti interattivi all'interno della scena

All'interno di SurviveVR vi sono svariate tipologie di oggetti interattivi che possono essere raggruppati in alcune categorie:

1. **oggetti appartenenti all'interfaccia grafica**: sono oggetti che mostrano il punteggio al giocatore o pulsanti, ubicati in una determinata posizione di fianco al giocatore, con cui questi interagisce per muoversi nel menu principale, iniziare la partita, uscire dall'applicazione, cambiare arma o ricaricarla;

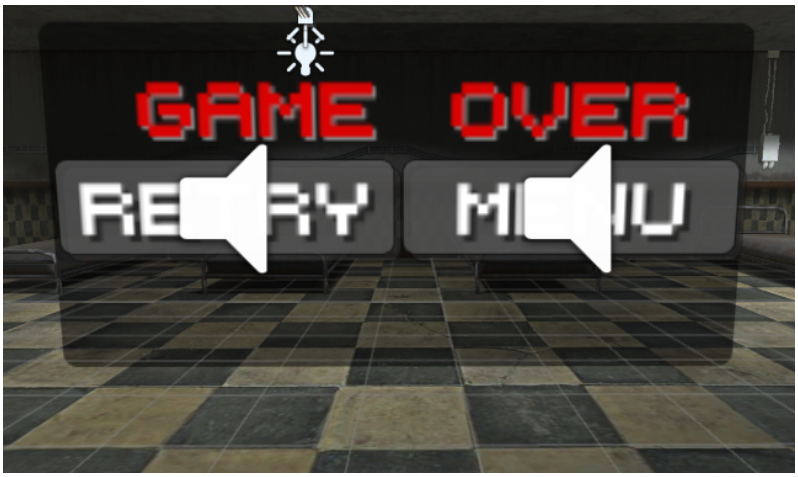

Figura 5.2: Elementi dell'interfaccia grafica

2. **nemici**: sono oggetti che si muovono all'interno della scena e cercano di uccidere l'utente attaccandolo in svariati modi. Con questi non è possibile interagire direttamente ma solo tramite i colpi delle armi da fuoco;

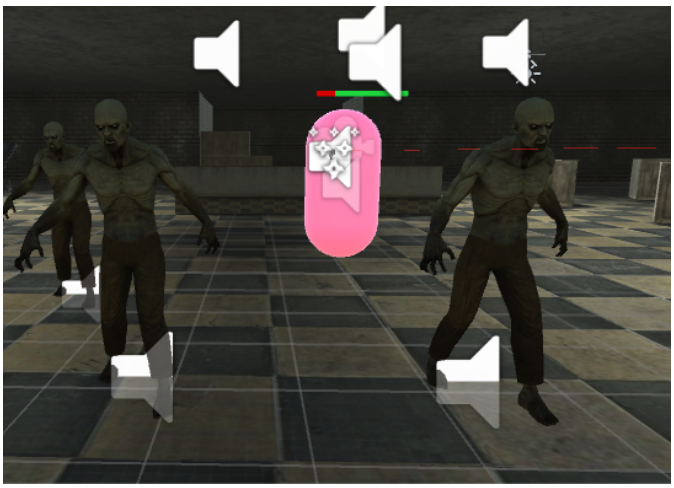

Figura 5.3: Nemici

3. **armi e kit medici**: oggetti che permettono il cambio di arma (in caso in cui si effettui la prima interazione con essi), l'incremento di munizioni relativo a quella raccolta o che ripristinano parte della salute del giocatore.

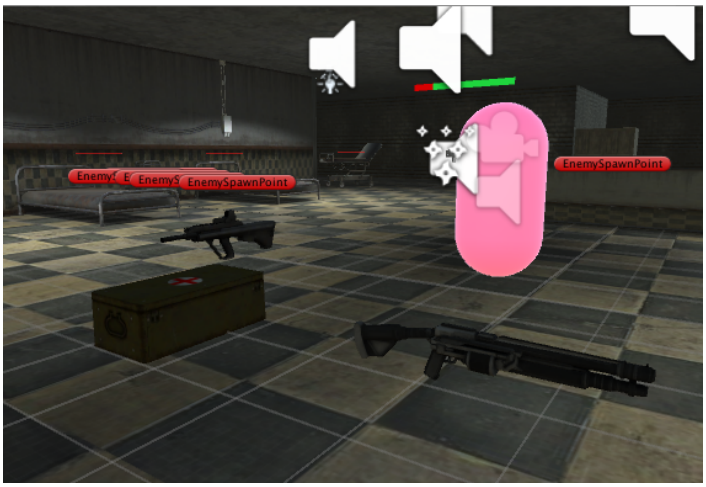

Figura 5.4: Oggetti da raccogliere

5.4 L'ambiente grafico

In un applicazione VR è di fondamentale importanza avere un ambiente grafico coerente agli occhi del giocatore per far sì che il partecipante si senta effettivamente immerso all'interno del mondo virtuale sia a livello visivo che fisico. Si parla di coerenza proprio perché la visione dell'ambiente virtuale deve essere simile o riconducibile in parte, per il giocatore, a quella del mondo fisico. Questa coerenza passa quindi attraverso il sistema di illuminazione, i modelli poligonali utilizzati e il sistema fisico. L'ambiente grafico è stato modellato utilizzando quasi esclusivamente gli asset scaricati dall'Asset Store di Unity (a partire dall'audio fino alle texture di qualsiasi modello poligonale). L'illuminazione all'interno dell'ambiente è calcolata dinamicamente e per far sì che questi siano illuminati in maniera realistica è sufficiente inserire dei punti luce all'interno della scena (che sono rappresentati dalle lampade appese sul soffitto dell'ambiente di gioco).

5.5 Le scene dell'app

Una scena non è altro che un ambiente al cui interno si svolgono le azioni di gioco. Queste possono essere composte da qualsiasi tipologia di elemento e vanno caricate per poter cambiare sezione nell'applicazione o mostrare elementi differenti non presenti nella scena corrente.

Survive VR è composto da due scene:

- MainMenu;

- GameScene.

5.5.1 MainMenu

MainMenu è la scena di partenza dell'applicazione e rappresenta una schermata principale, comune a qualsiasi videogioco, circondante l'utente all'interno dell'ambiente 3D. Negli svariati pannelli sono mostrati i pulsanti di gioco, i controlli, il miglior punteggio ottenuto ed i crediti del gioco. La scena è composta da molteplici elementi, tra cui il giocatore con la sua telecamera e il suo puntatore, che insieme formano l'esperienza di gioco. All'interno di MainMenu è possibile avviare l'esperienza e transitare alla scena GameScene, cliccando sul pulsante *Start*, oppure uscire dall'applicazione cliccando sul pulsante *Exit*. Queste operazioni avvengono grazie a degli script attaccati ai pulsanti che verranno spiegati nel dettaglio nel paragrafo seguente insieme a tutti gli altri elementi dell'applicazione.

5.5.2 GameScene

GameScene è la scena su cui si svolge l'azione di gioco. In questa compaiono l'ambiente, i nemici, il giocatore e le interfacce grafiche di fine partita o di selezione dell'arma. Al termine della sessione viene permesso di tornare al menu principale, cliccando sul pulsante *Menu*, oppure ricominciare la sessione di gioco premendo sul pulsante *Retry*. Come nella scena principale ogni elemento interattivo o che muta il proprio comportamento durante l'azione di gioco ha attaccato uno script, atto a realizzarne questa caratterizzazione.

5.6 Architettura dell'applicazione

L'applicazione è strutturata in 5 macro blocchi all'interno della Hierarchy del progetto (4 nella MainMenuScene). Questi sono il *GameMaster* (GM), il *GameSceneLevel*, il *Player*, la *UI* e i *Poolers*. All'interno di questi tutti gli elementi sono raggruppati in base alle funzionalità o ad un collegamento logico con l'elemento padre. Ogni blocco è indipendente da altri elementi per il funzionamento, a parte alcuni riferimenti ad oggetti necessari per effettuare transizioni o interazioni al momento giusto. Nello sviluppo dell'applicazione è stato preferito un approccio il meno possibile basato sulle funzioni *SendMessage()* e *FindObject()*, in questo modo è stato possibile semplificare le operazioni di personalizzazione/modifica e di comprensione senza dover stravolgere intere parti di codice. Di seguito vengono illustrate le peculiarità di ciascuno di questi blocchi con le relative scelte di design.

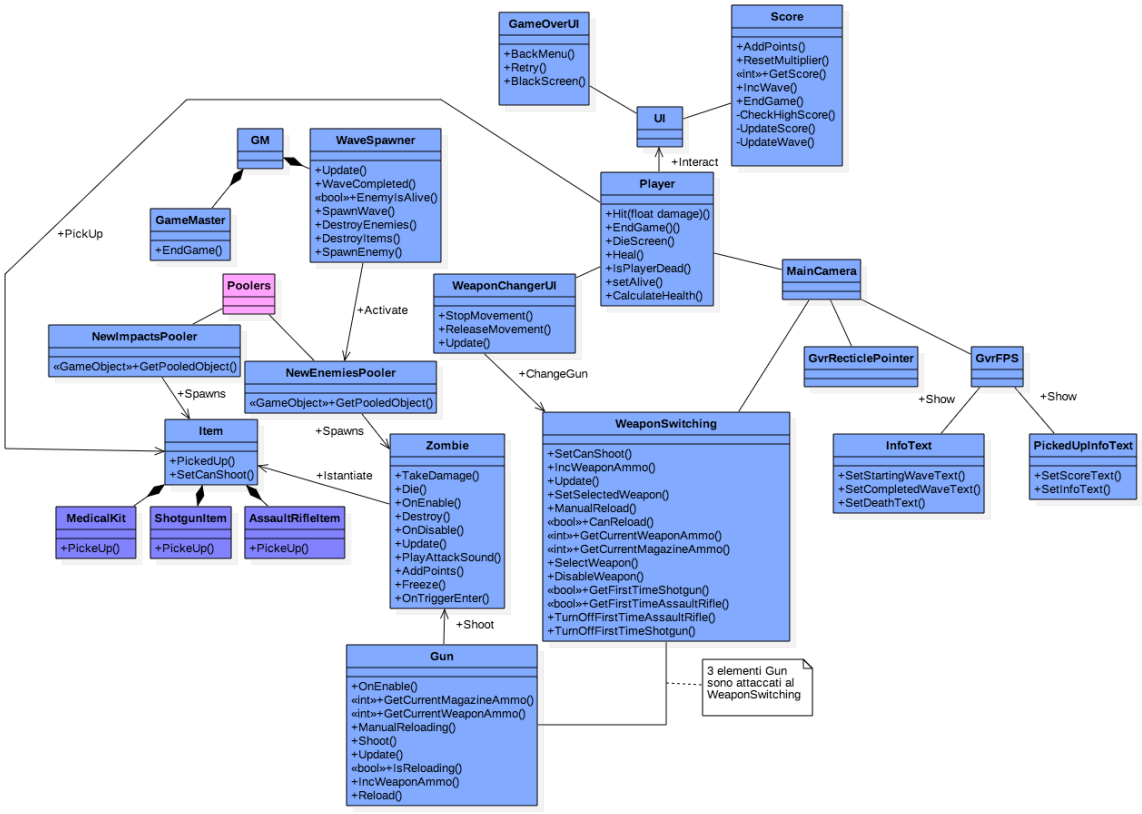

Figura 5.5: Diagramma logico di SurviveVR

5.6.1 Il GameMaster (GM)

Il GM è l'elemento atto a generare le ondate di nemici, distruggerli alla morte del giocatore (funzioni implementate nello script *WaveSpawner*) e conseguentemente mostrare all'utente la schermata di Game Over (realizzata nello script *GameMaster*). Il GM è padre di tutti gli *SpawnPoint* (punti di nascita) presenti nella scena e li utilizza come luoghi di generazione dei nemici. Il *WaveSpawner* opera in svariati stati interni tramite cui definisce in quali situazioni si trova la scena di gioco. Per esempio è di fondamentale importanza non generare nemici fino a quando non è terminata un'ondata, quando il giocatore è morto o nel periodo di attesa tra un'ondata e l'altra. Per far ciò è stato implementato all'interno di tale script lo *State Pattern* per poter gestire in maniera corretta il susseguirsi di queste differenti tipologie di stato. All'interno di questo script è stato anche implementato un esempio di *Prototype Pattern* combinato con *Object Pool Pattern* per l'istanziazione dei nemici nei vari SpawnPoint sparsi per la scena, questo avviene grazie ai

Poolers presenti nel quinto blocco dell'architettura. Ogni nemico istanziato viene quindi generato come copia di un elemento prefab (Prototipo) presente all'interno della cartella Prefabs. La scelta di implementare lo State Pattern è nata dall'esigenza di attribuire un comportamento diverso al WaveSpawner in base alla circostanza di gioco così che potesse controllare l'avanzamento dell'azione nell'applicazione, mentre la scelta di utilizzare l'Object Pool Pattern è scaturita dalla volontà di diminuire al minimo la mole di memoria allocata per i nemici. Se non fosse stato utilizzato questo pattern sarebbe insorto un consumo di memoria dovuto alla continua allocazione effettuata dal sistema nel caso in cui il numero di nemici fosse stato molto elevato (anche se Unity è ottimizzato per sprecarne il meno possibile).

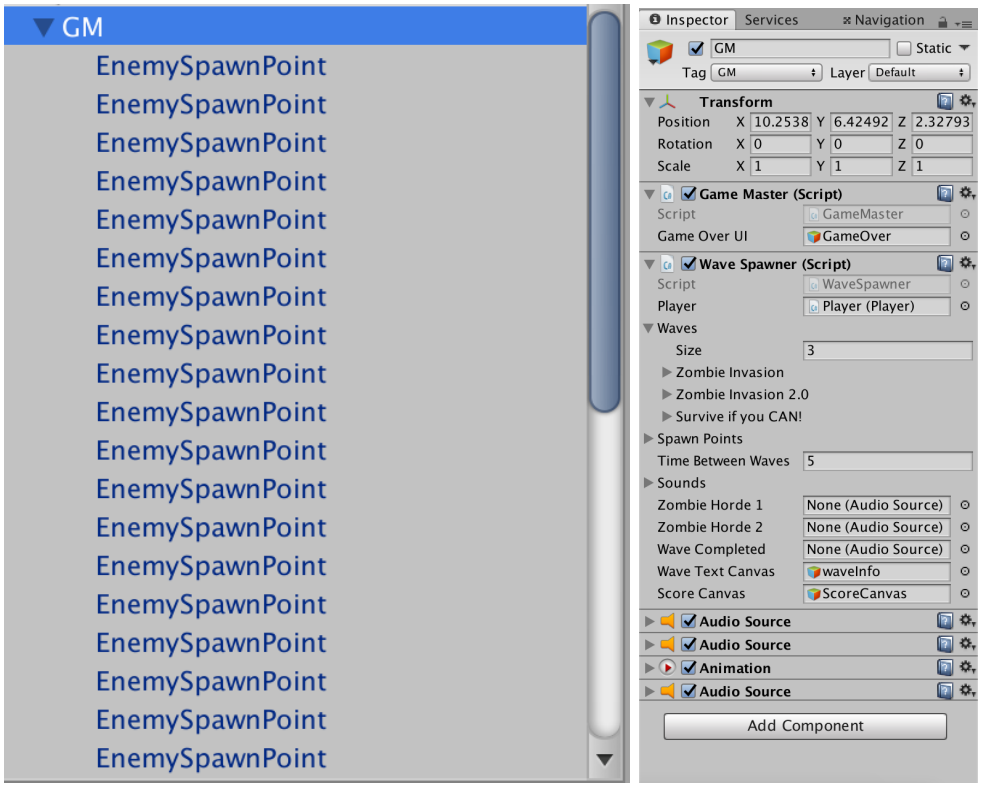

Figura 5.6: Il GameMaster

code.

```
/* Parte di codice per l'attivazione dei nemici all'interno del
    Pooler inerente */
void SpawnEnemy(Transform enemy){
   // Spawn Enemy
```

```
   Debug.Log("Spawning Enemy: " + enemy.name);

   if (spawnPoints.Length == 0) {
      Debug.LogError ("No spawn points referenced.");
   }

   Transform sp = spawnPoints[Random.Range(0, spawnPoints.Length)];

   GameObject obj = NewEnemyPoolerScript.current.GetPooledObject();
   if (obj == null) return;
   obj.transform.position = sp.position;
   obj.transform.rotation = sp.rotation;
   obj.SetActive(true);
}

         ...   ...   ...
         ...   ...   ...
         ...   ...   ...

   /* Parte di codice per i differenti stati nell'Update.
      Questi vengono cambiati all'interno di altre funzioni dello
          script */
void Update(){
   if (!player.IsPlayerDead ()) {

      if (state == SpawnState.WAITING) {
         // Verifica di nemici in vita nell'ondata corrente
         if (!EnemyIsAlive ()) {
            // Inizia un nuovo round
            WaveCompleted ();
            return;

         } else {
            return;
         }
      }

      if (waveCountdown <= 0) {
         if (state != SpawnState.SPAWNING) {
            // Inizio della fase di spawn dell'ondata
            if (Random.Range (0f, 2f) <= 1f) {
               zombieHorde1.Play ();
               anim.Play ();
            }
```

```
            else
               zombieHorde2.Play ();
            StartCoroutine (SpawnWave (waves [nextWave]));
         }
      } else {
         waveCountdown -= Time.deltaTime;
      }
   } else {
      if (!doneEndGame) {
         waveText.SetDeathText ();
         doneEndGame = true;
      }
      StartCoroutine (DestroyEnemies ());
      DestroyItems ();
   }
}
```

Listato 5.1: Design pattern nello script WaveSpawner.cs

5.6.2 GameSceneLevel / MainMenuLevel

In GameSceneLevel / MainMenuLevel sono raggruppati tutti gli elementi grafici che compongono la scena. Tra questi vi sono tutti gli elementi strutturali, gli oggetti d'arredamento e i punti luce.

5.6.3 Il Player

Il Player è l'elemento più complesso di tutta l'applicazione e come si può vedere dalla figura 5.7 è composto da molteplici oggetti dotati di script, oltre che essere parzialmente collegato con svariati elementi di gioco o dell'interfaccia grafica. Questi implementa le armi da fuoco con relativa gestione, l'interfaccia grafica che permette di cambiare bocca di fuoco o ricaricare quella equipaggiata, la barra della salute, i testi informativi che compaiono davanti all'utente e la schermata di morte del giocatore.

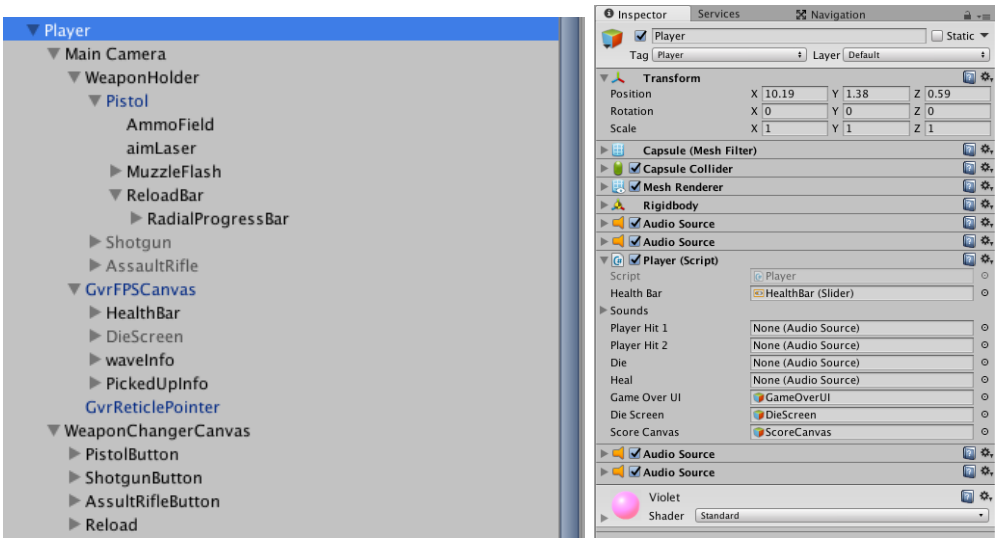

Figura 5.7: Il Player

Lo script *Player* attaccato al GameObject Player permette di effettuare la gestione delle statistiche relative al giocatore, capire quando questi è morto e mostrare le finestre relative quando ciò avviene. Questi aggiorna anche gli elementi grafici ad esso collegati, come la HealthBar, o richiama il punteggio per comunicare l'azzeramento del moltiplicatore di punteggio una volta che è stato colpito. All'interno del Player vi sono altri blocchi che ne identificano due aree ben distinte: il *WeaponChangerCanvas* e la *MainCamera*.

WeaponChangerCanvas è un semplice selettore di armi che compare solo quando il giocatore ruota il proprio capo verso l'alto, ruotando quindi insieme alla visuale. Da questo oggetto vengono generati degli eventi che effettuano la ricarica dell'arma selezionata o la cambiano con quella selezionata. Tali eventi vengono generati dall'ambiente di sviluppo (implementazione del pattern *Observer*) alla pressione del pulsante.

La MainCamera è la telecamera attraverso cui il giocatore vede il mondo di gioco. Dal momento in cui viene si effettua l'importazione del SDK di GoogleVR nel progetto, all'interno della MainCamera viene inserito uno script chiamato *GvrPointerPhysicRaycaster* atto ad individuare gli oggetti posti entro una data distanza dallo sguardo dell'utente. E' proprio grazie a questo *Raycaster* [92] che è possibile capire quando il giocatore sta puntando verso un elemento con cui vuole interagire. Questi non è altro che un raggio uscente dalla posizione della telecamera nella direzione ad essa frontale e che prosegue per una distanza range. Nel momento in cui questo raggio intersecasse un oggetto interagibile, il puntatore *GvrRecticlePointer* (prefab del package GoogleVR che implementa un cursore sullo schermo) cambierebbe forma per mostrare all'utente che è possibile effettuare un'interazione con tale elemento

dell'applicativo. Per esempio se il giocatore mirasse verso un nemico intersecante il *Raycast* e premesse il pulsante sul Cardboard, l'azione che ne conseguirebbe sarebbe quella relativa ad un oggetto *Enemy* (solo gli *Zombie* nella versione attuale dell'applicazione), con conseguente esplosione di un colpo di arma da fuoco nella direzione del Raycast. Alla MainCamera sono attaccati un *WeaponHolder* per la gestione delle armi, il *GvrRecticlePointer* accennato precedentemente e un altro prefab chiamato *GvrFPSCanvas* che non è altro che un Canvas su cui vengono mostrati i frame per secondo (FPS) tenuti dall'applicazione sullo smartphone. Al GvrFPSCanvas sono stati attaccati tutti gli elementi da visualizzare davanti allo sguardo del giocatore con i relativi script, quali la schermata da attivare alla sua morte *DieScreen*, la sua barra della salute *HealthBar*, e due pannelli di testo dedicati alle ondate e agli oggetti raccolti, relativamente *WaveInfoText* e *PickedUpInfoText*.

Il *WeaponHolder* è un oggetto a cui è associato uno script chiamato *WeaponSwitching*. Questi permette di gestire il cambio di armi, l'incremento delle munizioni e le ricariche di tutte le quelle presenti all'interno del WeaponHolder. A questi sono attaccate 3 tipologie differenti di armi a cui sono associati gli stessi componenti e lo stesso script, *Gun*, al cui interno sono implementati tutti gli elementi che caratterizzano le armi da fuoco: la loro ricarica, l'incremento delle munizioni, la cadenza di fuoco, le munizioni massime e la capacità del caricatore.

5.6.4 L'UI

All'interno di UI vi sono tutti i componenti dell'interfaccia grafica scollegati dalla visuale dell'utente. Questi sono renderizzati nel mondo virtuale e servono per mostrare la schermata di fine partita alla sua morte, le schermate nere di transizione e il punteggio all'utente. Queste vengono attivate attraverso delle chiamate del Player o all'attivazione di un evento generato da un pulsante (la schermata nera, per esempio, è la parte di *View* generata dalla pressione di un pulsante).

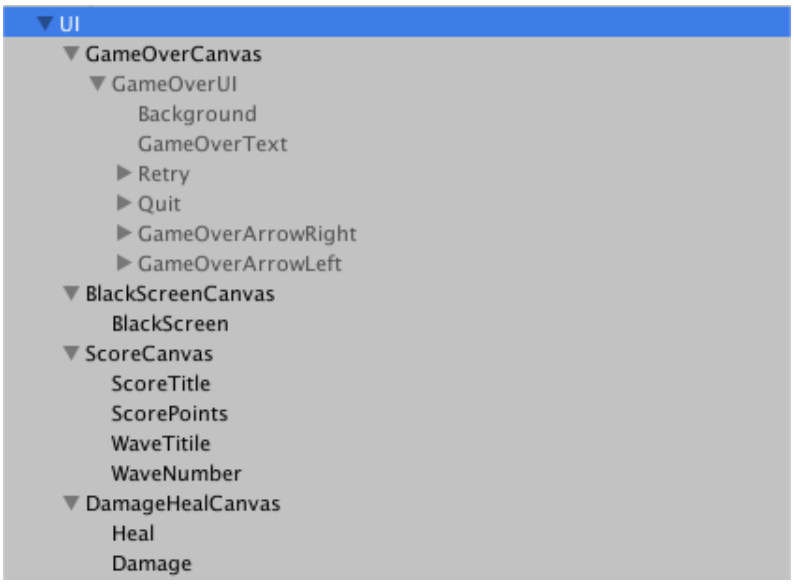

Figura 5.8: Hierarchy di UI

5.6.5 I Poolers

Come accennato precedentemente all'interno di *Poolers* sono contenuti i due Pool che permettono la generazione dei nemici e degli effetti particellari istanziati dalle armi da fuoco. In questo modo è possibile non sprecare memoria allocandone solamente una quantità ridotta. Questi nascono dall'implementazione, come già accennato, dell'*Object Pool Pattern* rappresentante una soluzione di alleggerimento del carico. Tale implementazione avviene creando un arraylist per ciascun pooler contente tutti gli elementi da gestire e al momento dell'istanziazione o della distruzione di un oggetto semplicemente lo attiva/disattiva all'interno di tale lista associata e lo riutilizza non appena si presenta la possibilità. In questo modo non vengono distrutti o creati oggetti aggiuntivi inutili. Un altro pattern utilizzato per questo oggetto è il *Singleton Pattern* in modo tale da rendere unico ognuno dei due elementi all'interno dell'ambiente, ciò li rende accessibili da chiunque all'interno dell'ambiente ma nell'architettura proposta solo un elemento accede ai Poolers.

Figura 5.9: Hierarchy di Poolers

code.

```
public class NewObjectPoolerScript : MonoBehaviour {

   public static NewObjectPoolerScript current;
   public GameObject pooledObject;
   public int pooledAmount = 20;
   public bool willGrow = true;

   public List<GameObject> pooledObjects;

   void Awake(){
      current = this;
   }

   // Use this for initialization
   void Start () {

      pooledObjects = new List<GameObject> ();
      for (int i = 0; i < pooledAmount; i++) {
         GameObject obj = (GameObject)Instantiate (pooledObject);
         obj.SetActive (false);
         pooledObjects.Add (obj);
      }

   }
   /* Funzione per ritornare l'oggetto che pu essere utilizzato per
       attivazione */
   public GameObject GetPooledObject(){
      for(int i = 0; i < pooledObjects.Count; i++){
         if (!pooledObjects [i].activeInHierarchy) {
            return pooledObjects [i];
```

```
        }
      }
      if (willGrow) {
        GameObject obj = (GameObject)Instantiate (pooledObject);
        pooledObjects.Add(obj);
        return obj;
      }
      return null;
    }
}
```

Listato 5.2: Applicazione di Object Pool Pattern nello script NewObjectPoolerScript.cs

5.7 Applicazione in esecuzione

Una volta avviata l'applicazione ed indossato il dispositivo Google Cardboard l'utente si trova immerso in un ambeiente virtuale che simula una vecchia ala di ospedale. Davanti a lui si paleseranno tutti gli elementi facenti parte del menu principale dell'applicazione.

Figura 5.10: Ambiente mostato all'utente all'avvio dell'app

L'utente a questo punto può guardarsi attorno per leggere i risultati relativi al miglior punteggio ed imparare i controlli all'interno dell'applicativo.

Figura 5.11: Scena della schermata principale dell'applicazione

Una volta avviata la sessione di gioco l'utente si ritrova nell'ambiente a difendersi dalle ondate di nemici e a raccogliere gli oggetti forniti per la propria sopravvivenza.

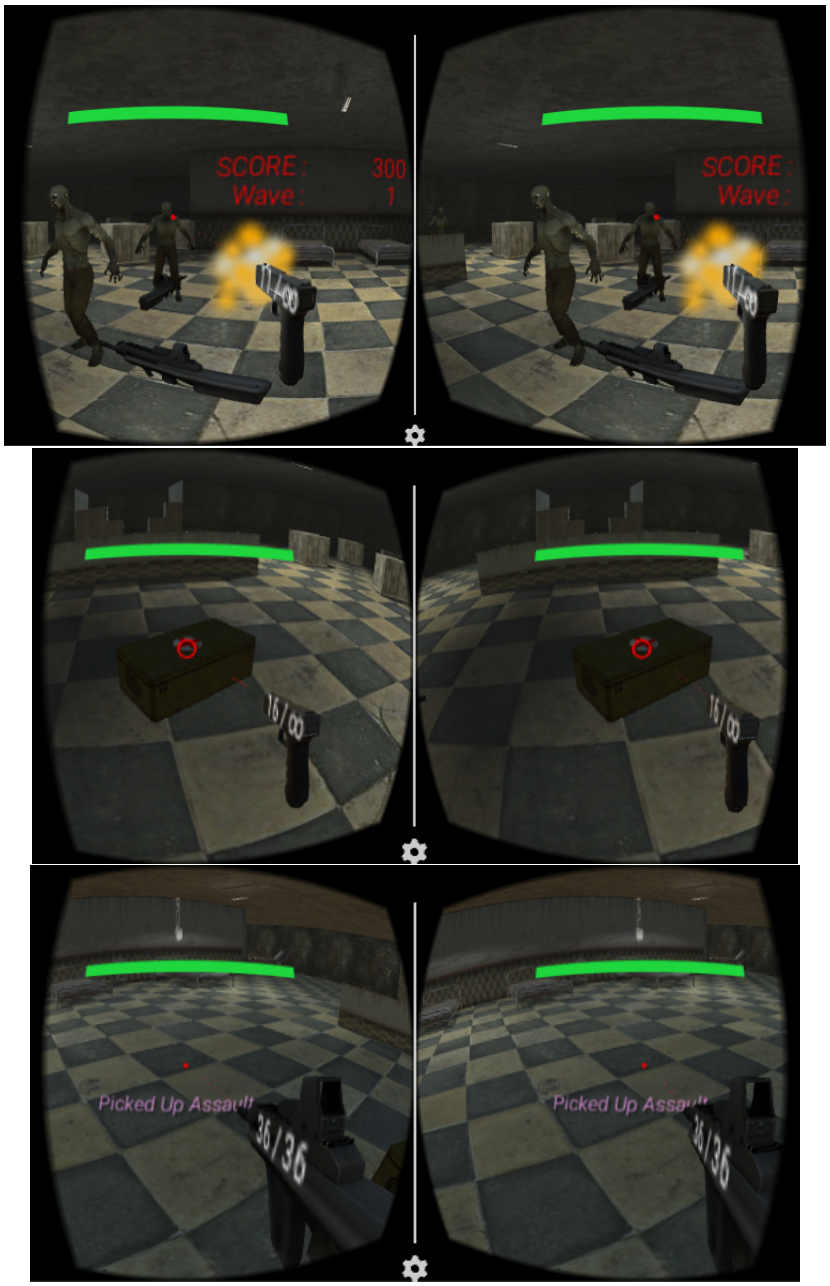

Figura 5.12: Scene di gioco

Durante l'azione di gioco, l'utente ha la possibilità di cambiare arma o ricaricarla utilizzando l'interfaccia grafica a comparsa posizionata leggermente sopra al proprio sguardo.

Figura 5.13: Interfaccia grafica per il cambio dell'arma o la ricarica

Alla morte del giocatore si manifesta l'interfaccia grafica con cui l'utente può interagire per ricominciare la partita o tornare al menu principale.

Figura 5.14: Interfaccia grafica di Game Over

5.8 Sviluppi futuri dell'applicazione

L'applicazione presentata sicuramente non può definirsi completa, i miglioramenti ad essa apportabili sarebbero molteplici, a livello di ottimizzazione del codice, ad aspetto grafico o di meccaniche di gioco. Un esempio di miglioria sarebbe l'implementazione di svariati nemici differenti che estendessero una classe generica Enemy ed implementassero delle funzioni aggiuntive in base al proprio comportamento. Un'altra miglioria potrebbe essere un sistema di interazione *drag and drop* per lo spostamento di alcuni oggetti dell scenario, in modo tale da fornire la possibilità all'utente di costruire barricate ad ogni

assalto. Sarebbero inoltre implementabili effetti visivi agiuntivi per le fasi di sparatoria, per fornire un feedback più immediato durante il danneggiamento dei nemici o per evidenziare gli oggetti da raccogliere una volta selezionati.

Attualmente l'applicativo è compatibile solo con dispositivi Google Cardboard ma potrebbe essere portato su tutte le altre piattaforme di realtà virtuale, supportanti Unity, con estrema facilità.

Infine un ulteriore sviluppo potrebbe essere l'integrazione con dispositivi di rilevazione della posizione per poter far muover l'utente all'interno del mondo virtuale in un'area ben definita o qualsiasi altro device tracciante la posizione.

Conclusioni

Lo scopo di questa tesi è quello di valutare Google Cardboard, congiuntamente a tutta la famiglia di dispositivi che rappresenta, nell'aspetto puramente tecnico, in quello di mercato ed infine sullo sviluppo e sulla progettazione ad esso collegati, prendendo Unity come riferimento per il caso di studio.

Partendo da quest'ultimo, è importante rilevare l'estrema semplicità che l'ambiente fornisce nel realizzare applicazioni di realtà virtuale e non sono state riscontrate particolari problematiche durante lo sviluppo del software. Gli unici due elementi negativi riscontrati in Unity risultano essere la complessità nel mantenere ordinata l'architettura di un applicativo, dovuta alla possibilità di inserire riferimenti a GameObject direttamente dall'Inspector dell'applicazione, e le limitazioni che questi offre alla programmazione multi-threading portando gli sviluppatori ad evitare la programmazione concorrente.

Gli aspetti peculiari di un'esperienza VR vengono gestiti dagli engine normalmente al giorno d'oggi, per cui lo sviluppo, l'ingegneria e il design per applicativi di realtà virtuale può basarsi sulle basi poste dalle applicazioni First Person per ambiente desktop. Le infrastrutture costituenti le fondamenta di queste esperienze risultano quindi le medesime, permettendo agli sviluppatori di porre maggior attenzione sulla progettazione delle interazioni utente-mondo virtuale e multi-utente all'interno di un mondo condiviso. La progettazione di tali mondi al giorno d'oggi risulta una delle sfide più avvincenti, che basandosi sulle strutture multi-utente delle attuali applicazioni desktop dovrà trovare nuove forme di interazione sempre più efficaci ed intuitive che possano far percepire all'utente la multi-presenza ed un senso di completa appartenenza ad una determinata esperienza. Si immagini la possibiltà di parlare con amici apartenenti ai network sociali all'interno di un mondo virtuale esattamente nel modo in cui lo si farebbe in quello reale, oppure avere la possibiltà di vivere in prima persona esperienze cinematografiche o rendere gli utenti partecipanti di un mondo virtuale completamente interattivo che potrebbe fornire compiti sia cooperativi che competitivi, in modo molto simile ai Massive Multiplayer Online Role-Playing Games (MMORPGs) odierni. Infine si pensi alla possibilità di rendere immortali riferimenti culturali della storia umana, grazie alla realtà virtuale congiunta a fotocamere con visione a 360 gradi, e visitabili da

chiunque in qualsiasi momento solo tramite l'avvio di un app per dispositivo mobile. Qualsiasi tipologia di contenuto che può essere vissuto attraverso una visuale in prima persona può quindi essere progettato, grazie a questi engine, condividendo una buona parte delle tecniche di sviluppo per applicazioni First Person desktop.

Per quanto concerne Google Cardboard invece, questo rappresenta una famiglia di dispositivi a basso costo che ha riscontrato, fino ad ora, un'ottima risposta da parte del pubblico. I tre principali fattori di tale successo risultano essere:

- il prezzo contenuto;
- l'idea di utilizzare gli smartphone come unità elaboratrici, posseduti ormai da chiunque;
- l'incremento di volume continuo di app sugli store più diffusi, grazie a cui è risultata possibile una penetrazione nel mercato mainstream molto forte.

Tuttavia tra questi fattori si annidano anche i lati negativi del dispositivo. Il primo è rappresentato dalla qualità cotruttiva di questa tipologia di dispositivi accompaganta da una carenza qualitativa a livello di lenti montate. Spesso questi risultano fragili, non considerando le versioni di cartone che si rivelano delicate per la natura del materiale, e non adatti per sessioni d'uso prolungate. Le lenti utilizzate, escludendo nei visori appartenti alla fascia di costo più alta, portano ad affaticare la vista e a non permettere la messa a fuoco degli elementi anche lievemente lontani dalla posizione dell'utente. Congiuntamente a ciò le prestazioni tecniche degli smartphone, per quanto siano già di buon livello, risultano molto sottotono se paragonate a quelle degli Head-Mounted Display per personal computer presenti sul mercato anche se, con l'avanzare della tecnologia, è auspicabile che tale gap si riduca sempre maggiormente. Un altro aspetto negativo di questa famiglia di dispositivi risulta essere la mancanza di tracking della posizione dell'utente all'interno del mondo virtuale. Cardboard non supporta il tracciamento del corpo poiché sprovvisto di qualsiasi tipologia di sensore preposta a questo scopo. Un'altra mancanza risulta essere l'assenza di un controller tracciante i movimenti della mano atto a rendere più naturali le interazioni con il mondo virtuale. Questa tipologia di device è stata introdotta nell'evoluzione di Cardboard, Daydream View, che rappresenta una piattaforma generata per sviluppare una maggior immediatezza nelle interazioni congiuntamente a prestazioni tecniche più elevate. Tale visore risulta più costoso (costo di circa 70$) anche a fronte della qualità dei materiali e del dispositivo aggiuntivo fornito per la mano. Questi purtroppo, attualmente, è

utilizzabile solo con il sistema operativo Android (dalla versione 7.0 Nougat) e compatibile con un gruppo di smartphone molto ristretto (e costoso). Tale prezzo e tale chiusura sicuramente non aiuteranno il dispositivo ad affermarsi attualmente sul mercato come il proprio predecessore (entrambi attualmente presenti all'interno di esso), tuttavia risulta abastanza prevedibile che un calo del prezzo d'acquisto in congiunzione ad un'apertura ad altri sistemi operativi, in primis tra tutti iOS, e ad una fascia di smartphone più ampia permetterebbe un attecchimento sul mercato mainstream realmente importante.

Se questa famiglia di dispositivi riuscirà a mantenere un prezzo di vendita relativamente basso migliorando la qualità delle lenti, ad integrare un sistema di tracciamento della posizione congiunto a dei supporti per effettuare le interazioni in maniera più naturale con il mondo virtuale e a potenziare le capacità di elaborazione così da permettere la creazione di mondi complessi, allora potrebbe divenire il punto di riferimento nel mercato mainstream per il medium, a discapito della differenza tecnica con gli Head-Mounted Display più evoluti.

Nel futuro prossimo gli applicativi di realtà virtuale diventeranno un campo di investimento per le aziende di ogni settore esattamente come lo è stato quello delle app per dispositivi mobile negl ultimi anni. Oggigiorno i tempi sono realmente maturi, grazie a dispositivi come il Carboard, affinché il medium possa entrare con rilevanza all'interno della vita quotidiana di ciascuno di noi.

Ringraziamenti

Desidero ringraziare tutti coloro che mi hanno aiutato nella realizzazione di questa tesi. Innanzitutto ringrazio il professore Alessandro Ricci, relatore di questa, per la disponibilità e per l'aiuto fornitomi durante il processo di stesura. Successivamente vorrei ringraziare un caro amico, Carlo Ghetti, che ha speso parte del suo tempo per leggere e darmi feedback riguardo le bozze dei testi. Proseguo con un ringraziamento particolare a tutte le persone che mi hanno sostenuto in questo lungo percorso di studi e che mi sono state vicine sia nei momenti bui che in quelli gioiosi della mia vita. Per concludere vorrei ringraziare tutte le persone a me più care e la mia famiglia, in modo particolare mia madre, Leoni Sfenia, a cui è dedicato questo lavoro.

Bibliografia

[1] Barnes & Noble Books, *Webster's New Universal Unabridged Dictionary*, 2003.

[2] Sutherland, *The Ultimate Display*, Procedings of the 1965 Congress 2: 506-508, 1965.

[3] Brooks F. P., *Whats's Real About Virtual Reality?*, IEEE Computer Graphics and Applications, 1999.

[4] Sherman William R., Alan B. Craig, *"Chapter 35: Scientific Visualization", The Computer Science and Engineering Handbook*, CRC Press, 1997.

[5] *Dexta Robotics*, http://www.dextarobotics.com.

[6] Morton L. Heilig, *Stereoscopic Television Apparatus for Individual Use*, `https://www.google.com/patents/US2955156`, 1960.

[7] Comeau C., J. Bryan, *Headsight Television System Provides Remote Surveillance*, Electronics 3445, 1961.

[8] Sutherland, *Sketchpad: A Man-Machine Graphical Communication System*, SJCC, 1963.

[9] Sutherland, *A Head-Mounted Three-Dimensional Display*, American Federation of Information Processing Societies' $AFIPS$ Fall Joint Computer Conference $FJCC$ $33Pt.1$:757-764, 1968.

[10] Myron Krueger, *Artificial Reality*, Mass.: Addison-Wesley, 1982.

[11] Defanti, Thomas A. and Daniel J. Sandin, *Final Project Report R60-34-163*, U.S. NEA, 1977.

[12] Furness, Thomas A., *Fantastic Voyage*, Popular Mechanics 16312: 63-65, 1986.

[13] Cruz-Neira, Thomas Defanti, *The CAVE Audio Visual Experience Automatic Virtual Environment*, Communications of the ACM 356: 65-72, 1992.

[14] *Oculus Rift*, https://www.oculus.com/rift/.

[15] *RealSpace 3D Audio*, http://realspace3daudio.com.

[16] *HTC Vive*, https://www.vive.com/eu/product/.

[17] *Valve*, http://www.valvesoftware.com.

[18] *HTC*, http://www.htc.com/it/.

[19] *HDMI Port*, http://www.google.it/patents/US20110039438?hl=it, 2011.

[20] *PlayStation VR*, https://www.playstation.com/it-it/explore/playstation-vr/.

[21] *Sony*, https://www.sony.it.

[22] *PlayStation 4*, https://www.playstation.com/it-it/explore/ps4/.

[23] *Google Cardboard*, https://vr.google.com/cardboard/.

[24] *Google Daydream View*, https://vr.google.com/daydream/smartphonevr/.

[25] *Google Daydream*, https://vr.google.com/daydream/.

[26] *Durovis Dive*, https://www.durovis.com/en/index.html.

[27] *Samsung Gear VR*, http://www.samsung.com/it/wearables/gear-vr-r323/.

[28] *Microsoft Hololens*, https://www.microsoft.com/en-us/hololens.

[29] Henry S. Kenyon, Signal AFCEA, *Virtual Reality In the Sky*, http://www.afcea.org/content/?q=virtual-reality-sky, 2005.

[30] Maryann Lawlor, Signal AFCEA, *Techonlogy Hits the Beach*, http://www.afcea.org/content/?q=Article-technology-hits-beach, 2017.

[31] Sandra Jontz, Signal AFCEA, *Augmented Reality to the Rescue*, http://www.afcea.org/content/?q=Article-augmented-reality-rescue, 2016.

[32] *CXC Simulations*, http://www.cxcsimulations.com.

[33] *EyecadVR*, http://www.eyecadvr.com.

[34] *Paris VR - Google Cardboard*, https://play.google.com/store/apps/details?id=com.cardboard360images.parisvr\&hl=it.

[35] *The VR Museum of Fine Art*, http://store.steampowered.com/app/515020/The_VR_Museum_of_Fine_Art/.

[36] *Blender*, https://www.blender.org

[37] *Maya*, https://www.autodesk.it/store/products/maya.

[38] *Modo*, https://www.foundry.com/products/modo.

[39] *3DStudioMax*, https://www.autodesk.it/products/3ds-max/overviewma.

[40] *Photoshop*, http://www.photoshop.com.

[41] *Logic Pro*, https://www.apple.com/it/logic-pro/.

[42] *Ableton Live*, https://www.ableton.com.

[43] *Steinberg Cubase*, https://www.steinberg.net/en/products/cubase/start.html.

[44] *Audacity*, http://www.audacityteam.org.

[45] Bass, Clements & Kazman, *Software Architecture in Practice 2nd ed.*, Addison-Wesley, 2003.

[46] Christopher Alexander, *The Timeless Way of Building*, Oxford University press, 1979.

[47] *Fakespace FS2*, http://www.fakespacelabs.com/tools.html.

[48] Foxlin Eric, *Inertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter*, Procedings of the IEEE 96 Virtual Reality Annual International Symmposium $VRAIS$ pp. 185-194, 1996.

[49] Klymento V., C. E. Rash, *Human Performance with New Helmet-Mounted Display Designs*, CSERIAC Gateway 4(4): 1-4, 1995.

[50] *Google Play Store*, https://play.google.com/store/apps?hl=it.

[51] *Apple App Store*, https://itunes.apple.com/it/app/apple-store/id375380948?mt=8.

[52] *Designing for Google Cardboard*, https://www.google.com/design/spec-vr/designing-for-google-cardboard/a-new-dimension.html.

[53] *Lumberyard*, https://aws.amazon.com/it/lumberyard/.

[54] *CooperCube*, http://www.ambiera.com/coppercube/.

[55] *CryEngine*, https://www.cryengine.com.

[56] *Fuzor*, http://www.kalloctech.com.

[57] *Autodesk Stingray*, https://www.autodesk.com/products/stingray/overview.

[58] *Unity3D*, https://unity3d.com.

[59] *Unreal Engine*, https://www.unrealengine.com/en-US/blog.

[60] *Epic Games*, https://www.epicgames.com/it.

[61] *Unity Technologies*, https://unity3d.com/company.

[62] *Mac OS X*, https://www.apple.com/it/macos/sierra/.

[63] *Linux*, https://www.linux.it/linux.

[64] *Microsoft Windows*, https://www.microsoft.com/it-it/windows/.

[65] *Android*, https://www.android.com/intl/it_it/.

[66] *iOS*, https://www.apple.com/it/ios/ios-10/.

[67] *Adobe Flash Player*, http://www.adobe.com/products/flashplayer.html.

[68] *Nintendo WiiU*, https://www.nintendo.it/Wii-U/Wii-U-344102.html.

[69] *Nintendo Switch*, https://www.nintendo.it/Nintendo-Switch/Nintendo-Switch-1148779.html.

[70] *Nintendo 3DS*, https://www.nintendo.it/Famiglia-Nintendo-3DS/Famiglia-Nintendo-3DS-94560.html.

[71] *Microsoft Xbox One*, http://www.xbox.com/it-IT/xbox-one/consoles.

[72] Robert Nystrom, *Game Programming Patterns*, Genever Benning, 2014.

[73] *GameObject Class in Unity*, http://docs.unity3d.com/ScriptReference/GameObject.html.

[74] *Prefabs in Unity*, https://docs.unity3d.com/Manual/Prefabs.html

[75] *Component in Unity*, http://docs.unity3d.com/ScriptReference/Component.html.

[76] *UI Canvas in Unity*, http://docs.unity3d.com/Manual/UICanvas.html.

[77] *UI Button in Unity*, http://docs.unity3d.com/ScriptReference/UI.Button.html.

[78] *UI Text in Unity*, http://docs.unity3d.com/ScriptReference/UI.Text.html.

[79] *UI Image in Unity*, https://docs.unity3d.com/ScriptReference/UI.Image.html.

[80] *Unity Documentation*, https://docs.unity3d.com/Manual/index.html.

[81] *MonoBehaviour Class in Unity*, http://docs.unity3d.com/ScriptReference/MonoBehaviour.html.

[82] *SendMessage("") function in Unity*, http://docs.unity3d.com/ScriptReference/GameObject.SendMessage.html.

[83] *Find("") function in Unity*, http://docs.unity3d.com/ScriptReference/GameObject.Find.html.

[84] *FindObjectsWithTag("") function in Unity*, http://docs.unity3d.com/ScriptReference/GameObject.FindGameObjectsWithTag.html.

[85] *MonoDevelop software*, http://www.monodevelop.com/.

[86] *Microsoft .NET Framework*, https://www.microsoft.com/net/download/framework.

[87] *AudioSource component in Unity*, https://docs.unity3d.com/Manual/class-AudioSource.html.

[88] *Animation Clip component in Unity*, https://docs.unity3d.com/Manual/AnimationClips.html.

[89] *Animator component in Unity*, http://docs.unity3d.com/Manual/Animator.html.

[90] *Unity's Asset Store*, https://www.assetstore.unity3d.com.

[91] *Google VR SDK for Unity*, https://developers.google.com/vr/unity/.

[92] *Physics Raycaster in Unity*, https://docs.unity3d.com/Manual/script-PhysicsRaycaster.html.