



UNIVERSITÀ DEGLI STUDI ROMA TRE

Dipartimento di Ingegneria
Corso di Laurea Triennale in Ingegneria Informatica

Tesi Di Laurea

Virtualized meeting room: realtà virtuale per smart working e formazione

Laureando

Lorenzo Goldoni

Matricola 456802

Relatore

Prof. Maurizio Patrignani

Correlatore

Raphaël Bussa, Fingerlinks s.r.l.

Anno Accademico 2016/2017

Alla mia famiglia,

Ringraziamenti

Ringrazio anzitutto il Professore Maurizio Patrignani per i preziosi consigli. Inoltre ringrazio la società Fingerlinks s.r.l che ha creduto in me e nella quale ho trovato persone speciali.

Un sincero ringraziamento agli amici con cui ho condiviso i miei studi universitari Federico, Marco, Simone, Dalila e Alessio, per essere fonte di grande ispirazione e per avermi supportato in questi anni, in particolare Claudia, che sin dai tempi del liceo, mi aiuta rendendo un gioco anche le cose più noiose.

Ringrazio i miei amici di sempre Elena, Gabriele, Matteo, Marco, Andrea, Riccardo, Giulia Sorrentino, Niccolò Di Giosaffatte e Niccolò Passeri, che mi hanno reso l'uomo che sono. In particolare ringrazio Antonio Martinelli, Luca Lallai e Giulia Sbragia, compagni di mille avventure che porterò per sempre con me.

Un immenso grazie ad Antonio Bannò, Camilla, Carolina, Edoardo, Francesco, Luca Mancini, Matteo Perozzi, Matteo Sicari, Tommaso e Valerio. La loro amicizia mi ha illuminato e dato il coraggio necessario per superare le sfide della vita.

Prosegua ringraziando il mio amico Jacopo, la cui amicizia è così speciale da trascendere spazio e tempo.

Ringrazio infinitamente Enrica, perché niente mi motiva nella ricerca della mia felicità quanto il suo amore e la sua fiducia e questo lavoro ne è il simbolo.

Ringrazio i miei genitori, a cui devo tutto e i cui sacrifici mi rendono così orgoglioso da spingermi verso il raggiungimento dei miei sogni.

Il ringraziamento più grande va a mia sorella Maria Carola, la cosa più preziosa che io abbia e che da sempre mi dà la forza di crescere.

Indice

Indice	iv
Introduzione	vi
1 Realtà virtuale	1
1.1 Cos'è la realtà virtuale?	1
1.1.1 Definizione della realtà virtuale	1
1.1.2 Lo stato dell'arte	3
2 Smart working e ambienti di lavoro virtualizzati	11
2.1 Cos'è lo smart working?	11
2.2 Ambienti di lavoro e di formazione virtualizzati	13
2.2.1 Telepresenza	13
2.2.2 I vantaggi	13
2.2.3 Limiti	16
3 Tecnologie adottate	18
3.1 Unity	18
3.1.1 L'Editor di Unity	21
3.1.2 Architettura Software di Unity	24
3.2 Samsung Gear VR	29
4 Requisiti del Progetto	30
4.1 Una Sala Riunioni in Realtà Virtuale	30
4.2 Principali Funzionalità e Casi d'Uso	31

4.3 Esperienza d'uso	32
5 Progettazione e Sviluppo	33
5.1 Ambientazione Grafica	33
5.2 Interfaccia grafica	38
5.3 Architettura Software	40
5.3.1 Librerie Esterne	40
5.3.2 Configurazione della Realtà Virtuale	41
5.3.3 Gestione delle funzionalità in Rete	42
Conclusioni e sviluppi futuri	47
Bibliografia	48

Introduzione

Negli ultimi anni la realtà virtuale ha aumentato la sua popolarità grazie agli avanzamenti tecnologici ottenuti nel campo dei telefoni cellulari. Stiamo ancora muovendo i primi passi nell'ambito di questa nuova tecnologia, ma le sue potenzialità sono già evidenti. Inizialmente infatti i sistemi immersivi erano esclusivamente utilizzabili dai ricercatori, ma oggi vi è una distribuzione su larga scala e la realtà virtuale viene impiegata nei contesti più disparati. Questo nuovo mercato ha catturato l'attenzione dei "giganti" del settore informatico che ora investono risorse significative su dispositivi o applicativi per la realtà virtuale. Lo sviluppo di un'applicazione immersiva è però un processo ancora complesso, poiché oltre alla scarsa documentazione, bisogna tenere in conto fattori che ad ora esulavano dalle competenze di uno sviluppatore, come ad esempio la salute dell'utente. Per questo sempre più programmati hanno iniziato a creare librerie o motori grafici che semplificassero lo sviluppo delle applicazioni in realtà virtuale, così da poter lasciare lo sviluppatore più libero di concentrarsi sui contenuti. Uno di questi strumenti è il motore grafico Unity che permette la creazione di progetti di questo tipo, compatibili con diverse piattaforme.

Virtualized Meeting Room è un'applicazione mobile per la gestione in rete di sale riunioni virtuali. In queste l'utente è rappresentato da un avatar e ha la possibilità di parlare con altri utenti connessi alla stessa stanza e di inviare loro dei documenti presenti sul telefono. L'applicazione è pensata per funzionare con i visori Samsung Gear VR. Lo scopo principale di questo progetto è quello di facilitare l'attuazione dei concetti di smart working e di rendere più efficaci le tecniche di formazione aziendali.

L'attività di tirocinio è stata svolta presso l'azienda Fingerlinks S.r.l. per la quale lavoro. Il mio compito durante il tirocinio è stato occuparmi dell'analisi e dello sviluppo dell'intera applicazione e in particolare di:

1. Incontrare il cliente
2. Stilare i casi d'uso
3. Ideare l'esperienza virtuale
4. Progettare e ottimizzare l'ambiente grafico della simulazione
5. Creare le interfacce grafiche
6. Progettare e sviluppare le classi software
7. Connettere e far interagire in rete due Gear VR
8. Gestire lo scambio di dati tra gli utenti

Nel primo capitolo si descrive la realtà virtuale partendo dalla sua definizione e arrivando alle attuali tecnologie che la rendono accessibile, spiegando in breve le componenti hardware e software.

Il secondo capitolo tratta il concetto di smart working e i benefici che può portare agli ambienti di lavoro se unito alla realtà virtuale. Vengono discussi anche i limiti dell'attuale stato dell'arte.

Nel terzo capitolo si descrivono le tecnologie utilizzate durante lo sviluppo dell'applicazione Virtualized Meeting Room, ovvero Unity e il visore GearVR. Viene anche discusso l'ambiente di sviluppo e l'architettura software del motore grafico.

Il quarto capitolo tratta dell'analisi dell'applicazione dal punto di vista della raccolta dei requisiti imposti dal cliente a cui va destinato l'applicativo.

Nel quinto capitolo si descrive la progettazione e lo sviluppo dell'applicazione, trattando gli elementi chiave e le classi software più significative.

Capitolo 1

Realtà virtuale

1.1 Cos'è la realtà virtuale?

La realtà virtuale (VR) è in costante evoluzione, rendendo difficile darne una definizione specifica che non decada nel giro di pochi anni. Il modo migliore per descriverla è quindi considerare gli aspetti intrinseci cruciali che prescindono la tecnologia utilizzata. La sua concezione deve essere abbastanza generale da abbracciare sia la realtà virtuale dei nostri giorni sia quella che verrà in futuro.

1.1.1 Definizione della realtà virtuale

"Indurre un comportamento premeditato attraverso l'uso di stimolazioni sensoriali artificiali in un organismo che ha una minima, se non nulla, consapevolezza dell'interferenza." [6]

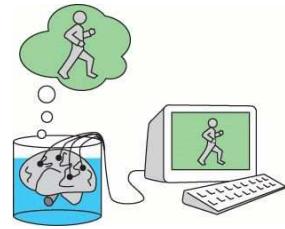
In questa possibile definizione di realtà virtuale compaiono quattro componenti chiave:

1. *Comportamento premeditato*: l'organismo sta facendo un'esperienza designata dal creatore che non coincide con la cognizione del mondo reale.
2. *Organismo*: si utilizza la parola organismo poiché qualsiasi essere vivente può immergersi in un ambiente virtuale. I neurobiologi dell'Università di Monaco effettuarono i primi test sperimentali sulla VR sottoponendo dei topi da laboratorio

a stimoli visivi mentre correva su un tapis roulant sferico. Altri esperimenti sono stati fatti su scimmie, pesci, scarafaggi e moscerini.

3. *Stimolazioni sensoriali artificiali*: attraverso degli strumenti tecnologici uno o più sensi dell'organismo vengono deviati e i loro input ordinari sono sostituiti da quelli artificiali
4. *Consapevolezza*: durante l'esperienza virtuale l'organismo viene "ingannato" nel sentirsi presente in un mondo fittizio e non essendone consci lo accetta come naturale.

Quest'ultimo fattore può sembrare frivolo, ma ci offre un'ulteriore possibilità di vedere in cosa consiste concretamente la realtà virtuale. Numerosi ricercatori di neurobiologia hanno dimostrato che il cervello di un animale sottoposto ad un'esperienza virtuale genera strutture cerebrali formate da *cellule di posizione e cellule grid*, contenenti le informazioni spaziali dell'ambiente circostante, proprio come accade nell'esplorazione di luoghi reali.



Un'altra componente cruciale della realtà virtuale è l'*interazione*: se la stimolazione sensoriale non dipende dalle azioni compiute dall'utente, il sistema VR viene definito *open-loop* ed è il più semplice grado di virtualizzazione della realtà. In caso contrario la configurazione viene chiamata *closed-loop*. In quest'ultima l'organismo ha un controllo parziale sulla manipolazione sensoriale, che potrebbe variare a seconda dei movimenti del corpo (testa, braccia, gambe, occhi, etc.), di comandi vocali, frequenza cardiaca, temperatura del corpo, resistenza elettrica cutanea (sudore).

E' infatti consuetudine far combaciare il più possibile la simulazione con il mondo fisico riproducendone anche le interazioni, poiché essendo il nostro cervello più familiare con queste, accetterà più facilmente l'ambiente come naturale.

1.1.2 Lo stato dell'arte

Siamo entrati nell'età moderna della realtà virtuale grazie agli avanzamenti nelle tecnologie di visualizzazione, percezione e computazionali portate dall'industria degli smartphones. I visori di realtà virtuale ora vengono prodotti in massa e distribuiti ad un'ampia varietà di persone. Questa tendenza è molto simile alle rivoluzioni da cui nacquero i personal computer e i web browser: maggiore è il numero di persone che hanno accesso alle tecnologie, maggiori sono le possibilità di queste ultime.

1.1.2.1 Applicazioni

Al giorno d'oggi la VR è attiva nei seguenti campi [1]:

1. *Videogiochi*: forse il campo in cui ha raggiunto la sua massima espressione ed il settore che traina la distribuzione su larga scala dei visori per la realtà virtuale.
2. *Militare*: ampiamente adottata dagli eserciti di molte nazioni per l'addestramento delle proprie truppe, poiché permette loro di imparare come reagire in situazioni ostili evitando i rischi e i costi delle vecchie simulazioni e aumentando il grado di realismo nello stress provato dai soldati (simulatori di combattimento, medicina da campo, guida di veicoli militari,etc.).
3. *Medicina*: spesso utilizzata per la simulazione di interventi chirurgici, superamento interattivo di fobie, chirurgia robotica, diagnostica, trattamenti per persone portatrici di handicap e per l'insegnamento.
4. *Moda*: attraverso cataloghi virtualizzati e progettazioni di negozi allestiti precedentemente in VR.
5. *Sport*: utile per migliorare le performance degli sportivi attraverso appositi ambienti virtuali, ma anche per avvicinare il pubblico mettendolo al centro dell'evento sportivo o facendogli vedere in anteprima come sarà la visuale dal suo posto dello stadio.
6. *Visualizzazione scientifica*: per mostrare più facilmente grosse moli di dati o informazioni complesse facilitando le collaborazioni interdisciplinari.

7. *Costruzione e Design*: le esplorazioni virtuali danno la possibilità di avere un feedback più realistico delle nuove strutture o soluzioni di design, come organizzazione di interni o progettazione di nuovi prodotti, prima ancora di realizzarli.
8. *Istruzione*: rende più diretto l'apprendimento in molte discipline che sarebbero difficili da visualizzare o ricordare, soprattutto nella scuola dell'infanzia.
9. *Media e informazione*: con l'utilizzo di film e documentari interattivi o di report giornalistici in cui l'utente è al centro della notizia e la può vivere in prima persona

1.1.2.2 Hardware

Il primo passo per capire come funziona la realtà virtuale è considerare cosa costituisce un intero sistema VR. Al contrario di quanto si possa pensare, non si tratta solamente delle componenti fisiche, come il visore, il computer o i controllers, ma l'organismo utilizzatore è altrettanto fondamentale. Per facilità in questo capitolo verrà considerato un essere umano .

il cervello decodifica lo spazio fisico, sia reale che virtuale, mediante la ricezione di input sensoriali. E' in grado di controllare la configurazione di questi ultimi mentre ne riceve le informazioni, permettendo la creazione di coordinate "coscienti" dello spazio circostante l'individuo.

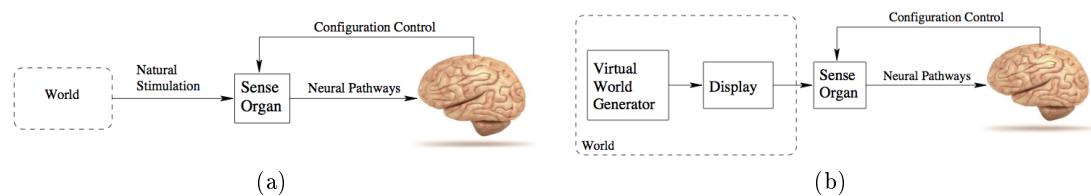


Figura 1.1: Il processo di raccolta, analisi e controllo di informazioni sensoriali nel mondo reale (a) e in quello virtuale(b).

L'hardware di una configurazione in realtà virtuale deve tenere traccia dei movimenti dell'utente per regolare gli stimoli artificiali in modo corretto, dando il desiderato senso di immersione. Un fattore di fondamentale importanza è lo studio degli spostamenti della testa nello spazio, in quanto codificati dal sistema nervoso centrale come un insieme di stimoli visivi, uditivi e vestibolari.

Per aumentare il grado di realismo della simulazione si tende a monitorare i movimenti di braccia o gambe. Infine è in egual modo significativo prendere in considerazione l'ambiente reale circostante come parte integrante del sistema VR perché, nonostante le stimolazioni controllate, l'utilizzatore continuerà a percepirllo.

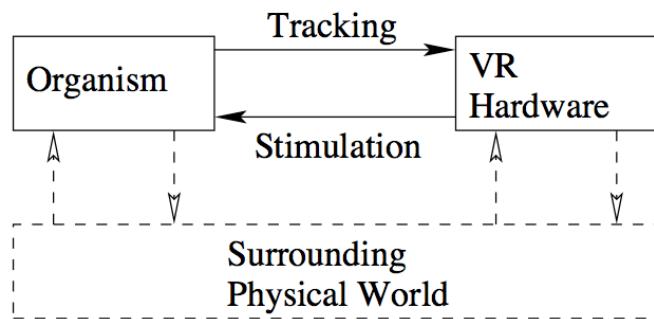


Figura 1.2: Lo schema di un sistema VR

Scendendo più nei dettagli i moderni sistemi di VR sono composti da tre componenti hardware principali: display, sensori, computer.

Displays:

Attraverso di essi avviene la stimolazione degli organi sensoriali, in particolare della vista e dell'udito.. Nei sistemi *CAVE*, un acronimo ricorsivo che sta per "cave automatic virtual enviroment", l'utente è avvolto da immagini 3D proiettate su sei piani di visualizzazione fissi ed immerso in un'audio surround. Nei sistemi *HMD*, head-mounted display, la configurazione consiste, come da definizione, in un visore montato sulla testa. Nei modelli più semplici viene utilizzato lo schermo di un cellulare e due lenti focali per gli occhi come nel caso dei Google Cardboard. Le apparecchiature più sofisticate utilizzano schermi progettati appositamente per la VR, che sfruttano la tecnologia LED o strumenti di *proiezione pico*, come *DLP(Digital Light Processing)*, *LCD(Liquid Crystal Display)* o *LQoS(Liquid Crystal on Silicon)*. Alcuni prodotti che implementano queste tecnologie sono *Google Glass*, *Microsoft Hololens* e *Avegant Glyph*. Prototipi in fase di sviluppo sono gli schermi a *campi luminosi* [18] e a *piani focali multipli* [12], purtroppo ancora poco competitivi perché non utilizzate nel mercato dei dispositivi mobili.



Figura 1.3: I Google Cardboard, un visore molto rudimentale (a) e un modello di Oculus DK2 con display integrati(b).

Sensori:

Per generare il corretto output artificiale vanno costantemente monitorati gli organi di senso dell'individuo, in particolare la loro posizione e l'orientamento. Per quest'ultimo si impiega usualmente un'*unità di misura interna* (IMU), formata da un giroscopio le cui misure sono integrate nel tempo dando una stima dell'inclinazione. Il rumore generato da questa misurazione, definito come errore di derivazione, può essere annullato o ridotto con l'impiego di altri strumenti come accelerometri e magnetometri.

A metà del secolo scorso le IMU erano pensabili solamente su velivoli e missili, dati i costi e le dimensioni molto elevate, ma al giorno d'oggi sono incorporate nel gran parte dei dispositivi mobili e sono la nuova tecnologia che ha portato ad una distribuzione su ampia scala della realtà virtuale.

Un'altra tecnologia che è diventata più accessibile sono le *telecamere digitali* grazie alle quali si può adottare un approccio di monitoraggio *visivo*. L'idea è di identificare nell'immagine dei *markers* o elementi fissi, che fungono da riferimento per il sistema. Questo processo impone però dei limiti sulla potenziale posizione o orientamento dei visori ed è per questo che sono in via di sviluppo intelligenze artificiali addette al riconoscimento di immagini che scalzino l'impiego di marker fisici.

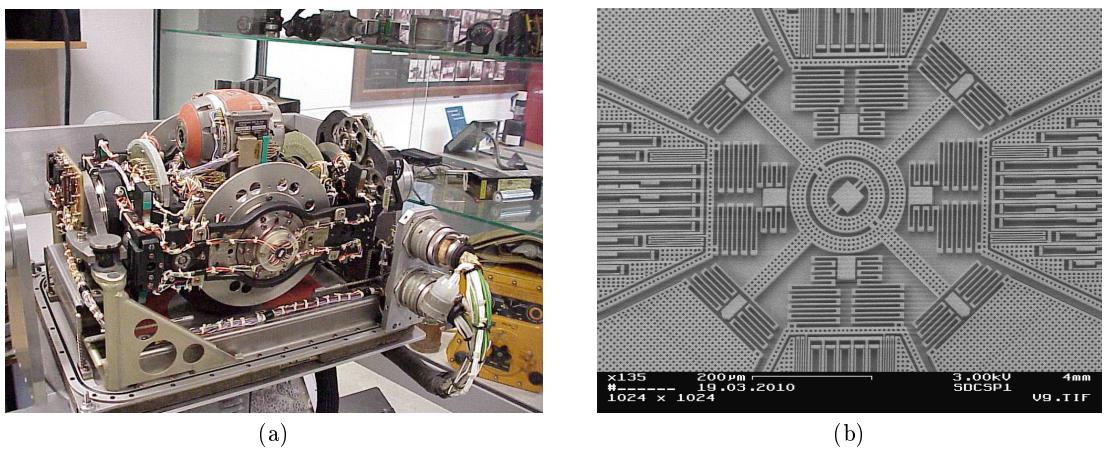


Figura 1.4: Un modello di IMU (LN-3 Inertial Navigation System) sviluppato negli anni '60 dalla Litton Industries (a) e un modello odierno situato in un giroscopio MEMS largo meno di un millimetro. (b)

Computers:

Sono considerati i *generatori di mondi virtuali* (VWG). Nei sistemi indossabili, il posizionamento del computer nello spazio fisico e il relativo scambio di informazioni con la periferica sono gli aspetti più importanti. Nelle configurazioni moderne si utilizzano dei cavi, limitandone e peggiorandone la mobilità. Quando i calcolatori non sono invece integrati al visore, come nel caso di un cellulare, questo problema non sussiste, ma la potenza di calcolo viene ridotta drasticamente, rendendo il mondo generato molto più semplice. Ci si aspetta in futuro la creazione di visori con computer integrati che adottino tecnologie wireless. Sono previsti ulteriori miglioramenti alle schede grafiche (GPU) che devono renderizzare, ovvero mostrare a schermo, elementi grafici sempre più complessi e numerosi che compongono gli ambienti virtuali.



Figura 1.5: Lo schema di un sistema VR

1.1.2.3 Software

La componente software che permette la creazione di esperienze virtuali è gestita dal VWG citato in precedenza. Esso prende in input i dati dei sistemi di basso livello, che indicano cosa stia facendo l'utente nel mondo reale e li gestisce in modo che le componenti di *rendering* possano creare una simulazione virtuale coerente e realistica.

La creazione dei mondi virtuali può essere gestita con diversi gradi di realismo, uti-

lizzando mondi puramente sintetici, ovvero che esistono solamente nel contesto della simulazione 3D, oppure estremamente realistici che ricreano luoghi reali mappati in precedenza con l'utilizzo di camere digitali, e di tecniche di rielaborazione dell'immagine come lo *SLAM* (*Simultaneous Localization and Mapping*) e il *texture mapping*.

La funzione base del software per un'applicazione VR rimane però il mantenimento della corrispondenza tra la posizione fisica e quella virtuale e del superamento di eventuali limiti qualora queste ultime non possano combaciare. Infatti nel caso in cui degli ostacoli fisici nel mondo reale non fossero replicabili nel mondo virtuale, o più probabilmente il contrario, lo sviluppatore deve gestire questa incompatibilità. Per fare ciò si può ampliare il mondo virtuale se l'utente non è limitato in uno spazio nel mondo fisico, considerando però che al crescere della libertà di movimento concessa ad una persona durante l'esperienza crescono anche i relativi problemi di sicurezza dato il parziale oscuramento dei sensi. Un'altra soluzione che permette la navigazione di spazi virtuali infinitamente grandi è l'inserimento di un sistema *locomotivo*, in cui l'indossatore si sposta nella simulazione stando in realtà fermo. Questo può essere implementato con l'ausilio di periferiche esterne. Purtroppo anche questa soluzione nasconde delle problematiche, poiché è facile indurre nell'utente effetti sgradevoli come il *motion sickness*, ovvero la sensazione di nausea data dall'incoerente differenza tra la vista e il senso dell'equilibrio.

Una parte più complessa affidata al lato software è la gestione della fisica virtuale. Quest'ultima è di fondamentale importanza per rendere più credibile la realtà virtuale.

Difatti se nella simulazione facciamo cadere un oggetto, il nostro cervello si aspetta che cada proprio come nella realtà. Vengono quindi utilizzati degli algoritmi di *rilevamento delle collisioni* che determinano se due o più oggetti si intersecano. Questo argomento raggiunge complessità così elevate quando si prendono in considerazione significative quantità di collisioni o gli effetti della propagazione di luce e suono virtuali, che sfocia in una disciplina a parte. Nel caso di un'applicativo connesso alla rete, il mondo virtuale condiviso è mantenuto su un server che gestisce e aggiorna i movimenti o le varie interazioni degli utenti.

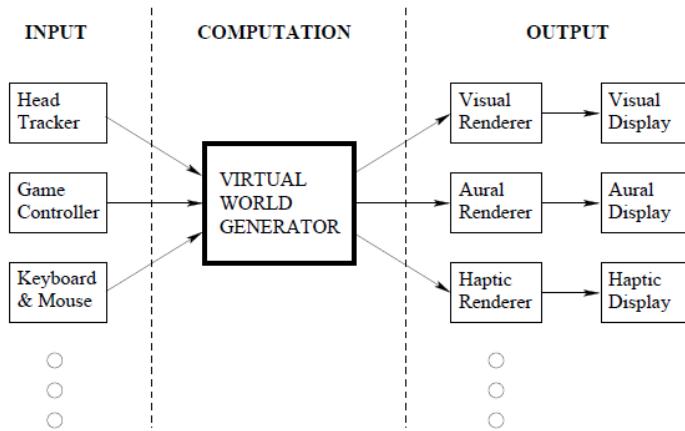


Figura 1.6: le funzioni di uno VWG schematizzate

In conclusione, per lo sviluppo VR, si può partire dal SDR *software development kit* di un visore che gestisca i driver e le componenti di basso livello come i sensori o le librerie grafiche, costruire da zero la fisica del mondo virtuale e implementare i requisiti dell'applicazione desiderata. Questo approccio consente di massimizzare le performance e amplia il controllo dello sviluppatore, ma è troppo complicato da gestire per applicazioni complesse. In questi casi si fa ricorso a VWG già pronti, lasciando in mano al creatore solo le componenti software ad alto livello. Alcuni esempi sono OpenSimulator, Vizard di WorldViz, Unreal Engine di Epic Games e Unity3D, che approfondiremo più avanti.

Capitolo 2

Smart working e ambienti di lavoro virtualizzati

2.1 Cos'è lo smart working?

Il termine *smart working* si riferisce alle nuove modalità lavorative all'interno di un'organizzazione, rese possibili dagli avanzamenti tecnologici e divenute essenziali dalle pressioni economiche, ambientali e sociali. Questi nuovi approcci non nascono da un'idea rivoluzionaria, ma sono il frutto della graduale evoluzione delle nostre pratiche lavorative che sono diventate sempre più collaborative e libere.

Alla base vi sono tre concetti fondamentali:

1. la revisione della leadership e del rapporto tra manager e dipendente, dal controllo alla fiducia;
2. il ricorso a tecnologie collaborative in sostituzione ai sistemi di comunicazione rigidi;
3. la riorganizzazione degli spazi di lavoro per slegarlo da un luogo fisico.

Il risultato dell'applicazione di questi concetti in contesti aziendali è l'aumento della produttività aziendale in un modus operandi che pone al centro dell'attenzione la persona, unificando gli obiettivi personali e quelli lavorativi. Questo processo responsabilizza il lavoratore che è consapevole e sente propri gli obiettivi aziendali e non vede più l'organigramma aziendale come una dura piramide da scalare, bensì si sposta l'attenzione dall'organizzazione, centrale nei vecchi modelli, al singolo, cercando di motivarlo e di aumentare la sua dedizione per le attività che svolge.

Diminuendo quindi la distanza tra la vita professionale e quella privata, il dipendente valorizza da sè le attività che svolge e risulta più efficiente. Spesso infatti il termine smart working si utilizza per intendere il telelavoro, la possibilità di lavorare da casa, la creazione di spazi per il coworking o a posti di lavoro più flessibili. Sebbene sia corretto, il concetto va molto oltre queste cose e risiede nella ridefinizione del lavoro per come lo intendiamo.

All'atto pratico la sua potenza non risiede nel fatto che gli smart workers non hanno orari d'ufficio rigidi o che non importi dove lavorino, ma nella richiesta del datore di lavoro, che vincola il lavoratore unicamente a ottenere i risultati previsti nei tempi stabiliti con il massimo della qualità,. Così facendo il subordinato si sente proprietario del proprio lavoro, autonomo in modalità e tempistiche e viene responsabilizzato, imparando a gestire il suo tempo in modo più intelligente. Il controllo in questo modello lascia spazio alla fiducia nei confronti di un sottoposto. Questa lenta rivoluzione è condotta dai mezzi tecnologici che forniscono più alte possibilità di condivisione fra persone, team e società.

Lo smart working favorisce quindi la nascita di nuove figure lavorative, come manager moderni che guidano dei gruppi di professionisti e ne gestiscono le risorse, passando così da controllori a più costruttivi mentori che lavorano sulle individualità della persona, che ne trae vantaggi sia professionalmente che personalmente.

2.2 Ambienti di lavoro e di formazione virtualizzati

2.2.1 Telepresenza

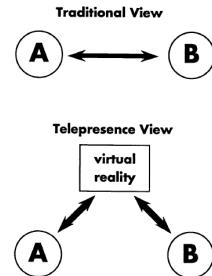
La *telepresenza* è un termine che si riferisce a sistemi che permettono agli utenti di sentirsi in altri luoghi con l'ausilio di un mezzo ingegneristico. Questo vocabolo fu coiato dallo scienziato cognitivo Marvin Minsky nel 1980 facendo riferimento ai sistemi di *teleoperazione* per la manipolazione remota di oggetti fisici.

Siamo già da decenni immersi in sistemi di telepresenza come quelli telefonici, le telecamere e persino registrazioni musicali, ma nessuno ha le potenzialità della realtà virtuale. Prendendo infatti in prestito il concetto di Minsky, possiamo definire nuovamente la VR come "*un'ambiente simulato nel quale un'individuo esperimenta la telepresenza*"[5]. Difatti il sistema di percezione umano si è evoluto con interazioni faccia a faccia, ed esse fungono da modello per tutti gli altri tipi di comunicazione, le quali vengono paragonate dal nostro cervello a esperienze in prima persona. Trovarsi dentro una simulazione realistica amplia quindi i livelli di intensità e interazione dei mezzi non immersivi.

Sebbene la realtà virtuale si riferisce all'esperienza di un individuo, condividere lo stesso spazio virtuale con altri utenti accresce il senso di realismo della simulazione, spingendone ancora oltre le potenzialità in ambiti lavorativo o di formazione.

2.2.2 I vantaggi

La realtà virtuale apre le porte a molteplici benefici in campi lavorativi e formativi. Fornendo una profondissima esperienza di telepresenza il luogo geografico da cui si svolge la propria professione non è più fondamentale, facilitando lo smart working e aumentando la centralità e il benessere della persona nella società.



Si può pensare di entrare in un edificio virtuale e lavorare o incontrare clienti senza il vincolo di essere nello stesso posto reale, utilizzando la comunicazione non verbale assente in altri strumenti di telepresenza. In un ambiente virtualizzato l'esperienza del

lavoratore può essere aumentata fornendogli strumenti altrimenti impossibili da utilizzare come traduttori real-time o più semplicemente integrazioni di servizi in modo più intuitivo e naturale.

Questo approccio può essere l'inizio di un radicale cambiamento nei luoghi in cui viviamo e lavoriamo, unificando tutti gli strumenti fisici che occupano spazio, in un unico visore. Un cambiamento simile è stato portato dagli smartphones che hanno incorporato diversi oggetti (agende, orologi, giornali, ecc.) in un unico dispositivo di ridotte dimensioni. La VR, in futuro, potrebbe eliminare gli edifici in cui oggi lavoriamo, lasciando spazio a nuovi luoghi di aggregazione, innescando un processo di *deurbanizzazione* più compatibile con le esigenze del nostro pianeta.

Un'altro proposto fondamentale della tecnologia è rendere le professioni più sicuri abbassando il rischio di infortuni e morte nel caso di lavori pericolosi. Con l'utilizzo di ambienti virtuali interattivi e di sistemi teleoperazionali, si possono svolgere compiti ad alto rischio, come lo sminamento di un campo, utilizzando un visore ed un robot specializzato che viene controllato a distanza in modo estremamente intuitivo e naturale. Questo processo di *ludicizzazione (gamification)*, ovvero l'utilizzo di elementi mutati dai giochi e delle tecniche di game design in contesti esterni ai giochi [10], può far beneficiare anche la classe operaia delle rivoluzioni tecnologiche di solito riservate a professioni più prestigiose e si propone di risolvere l'alto tasso di disoccupazione tra i giovani videogiocatori [11].

L'adozione di questo approccio porterebbe anche vantaggi socio economici come la riduzione dell'inquinamento dovuto agli spostamenti tra le abitazioni e i luoghi di lavoro. [3]

Risparmi Annuali	1.5 giorni/settimana	5 giorni/settimana
Distanza	1890 Km	6300 Km
CO2	365 Kg	1187 Kg
Tempo	61 ore	203 ore

Figura 2.1: I dati raccolti da uno studio dell'Università di Oxford sui risparmi ottenuti grazie al telelavoro nel Regno Unito.

Le potenzialità più grandi della realtà virtuale risiedono però in campo formativo. Bambini e ragazzi impossibilitati a raggiungere un luogo di studio possono comunque avere accesso ad un'istruzione. Nonostante i sistemi immersivi offrano strumenti unici ed estremamente utili, allo stesso tempo taglano i costi di formazione professionale evitando anche quelli di eventuali errori commessi da lavoratori inesperti. Questi due aspetti raggiungono l'apice in campi come la medicina, in cui sempre più applicazioni VR vengono utilizzate per insegnare a giovani dottori come portare a termine un'operazione chirurgica o come effettuare una diagnosi esplorando l'interno di un organo malato.

Altri campi che beneficierebbero della virtualizzazione sono le aree di ricerca scientifiche. Sempre più spesso infatti sono richiesti per l'insegnamento e la formazione simulatori molto costosi o relazioni geometriche in strutture di dati complesse difficili da visualizzare. La realtà aumentata è utile per l'insegnamento pratico, perché le abilità imparate nel mondo virtuale realistico possono essere applicate al mondo reale in modo intuitivo. Un'aspetto meno pratico, ma più psicologico su diversi ricercatori stanno studiando è il collegamento tra empatia e realtà virtuale. Se infatti consideriamo ambienti estremamente realistici, costruiti seguendo una fedele mappatura o con l'utilizzo di camere panoramiche, è possibile considerare un notevole aumento nell'empatia stimolata in un individuo che vive un'esperienza in prima persona. Vivere la guerra in Siria, un giorno nella cella d'isolamento in una prigione di massima sicurezza o nel corpo del sesso opposto sono alcune esperienze virtuali che più hanno scosso in modo empatico gli utenti, questo a dimostrazione di quanto può essere immersiva questa tecnologia. Sempre più organizzazioni stanno cambiando il loro modo di intendere il lavoro e la formazione, puntando su simulazioni in realtà virtuale incrementandone il mercato e innescando così un'ulteriore accelerazione dell'avanzamento tecnologico di questo nuovo mezzo.

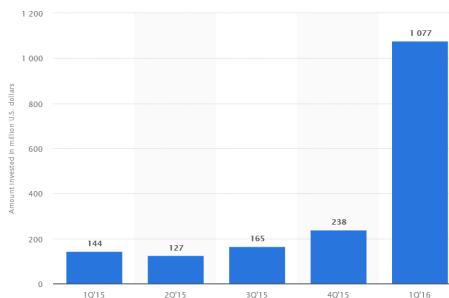


Figura 2.2: I dati raccolti da uno studio dell’Università di Oxford sui risparmi ottenuti grazie al telelavoro nel Regno Unito.

2.2.3 Limiti

Nonostante l’elevato potenziale della realtà virtuale in questi settori, l’adozione di queste nuove tecnologie sarà lento. I dispositivi che abbiamo a disposizione comportano alti prezzi e non vi sono ancora abbastanza applicazioni personalizzate che soddisfino le unicità dei vari settori in cui andrebbero ad operare. La principale funzione dei sistemi VR, ovvero quella di sovrapporsi agli organi di senso della persona, è spesso anche la causa principale della diffidenza nei confronti di questo nuovo mezzo.

Infatti, a differenza dei vecchi media (libri, radio, televisioni, ecc.) la realtà virtuale può causare stanchezza e nausea definita proprio *VR sickness*. Spesso il malessere può essere provocato da un hardware difettoso o da uno sviluppatore che ha sottovalutato gli effetti collaterali della VR. Chi sviluppa il software infatti deve avere come prima priorità la salute dell’utente, per questo motivo l’utilizzatore finale deve essere considerato come parte integrante del sistema. In molti casi la stanchezza sopraggiunge per il costante lavoro del cervello nel ricalibrare gli stimoli sensoriale incoerenti con la realtà che possono anche condurre a giramenti di testa. Essendo un’attività fisica in cui tutto il corpo è coinvolto, interazioni scomode all’interno dell’applicazione, come il continuo movimento delle braccia nell’esperienza virtuale per interagire con l’ambiente, possono portare ad effetti indesiderati come le *gorilla arms*, termine poco tecnico per definire la sensazione per la quale gli utenti non sopportano più il peso delle proprie braccia estese. Alcuni degli effetti collaterali devono ancora essere risolti, altri si formano con l’aggiunta di nuove funzionalità o componenti hardware.

Purtroppo anche con i migliori visori e con software di alta qualità una persona può reagire negativamente ad un'esperienza in realtà virtuale. ****Un fattore predisponente può essere l'etnia. Michael Sivak e Brandon Schoettle dell'University of Michigan hanno infatti dimostrato che è statisticamente più probabile per una persona asiatica soffrire della sopra citata VR sickness rispetto ad individui caucasici.

Aspect	U.S.	China	India	Japan	U.K.	Australia
Expected to be involved in activities that increase the frequency and severity of motion sickness	37.0%	40.3%	52.7%	25.9%	27.8%	29.7%
Would often, usually, or always experience some level of motion sickness	6-10%	6-10%	8-14%	4-7%	4-7%	4-8%
Would experience moderate or severe motion sickness at some time	6-12%	6-13%	8-17%	4-8%	4-9%	4-10%

Figura 2.3: I dati raccolti da uno studio dell'Università di Oxford sui risparmi ottenuti grazie al telelavoro nel Regno Unito.

In conclusione bisognerà aspettare sistemi meno ingombranti, più economici e un maggior numero di sviluppatori esperti per poter diffondere e sfruttare al massimo le potenzialità della realtà virtuale. Citando Dave Jones, il creatore del primo videogioco della serie Grand Theft Auto, "La realtà virtuale è ancora all'età della pietra" [?].

Capitolo 3

Tecnologie adottate

Questo capitolo descrive le tecnologie utilizzate per lo sviluppo di un'applicazione in realtà virtuale e connessa alla rete per i Samsung GearVR.

3.1 Unity

Unity è un motore grafico multipiattaforma sviluppato da Unity Technologies per la creazione di videogiochi (3D/2D) o di altri contenuti interattivi, quali visualizzazioni architettoniche o animazioni 3D in tempo reale.[17] La sua ampia lista di piattaforme supportate [15] e i bassi requisiti di sistema necessari per l'uso [16] lo hanno reso uno dei motori grafici più utilizzati dagli sviluppatori. L'ultima versione rilasciata è 'Unity 2017.2'.



A differenza di altri software, Unity permette l'utilizzo di diversi linguaggi di programmazione supportando:

- C#
- Javascript
- Boo

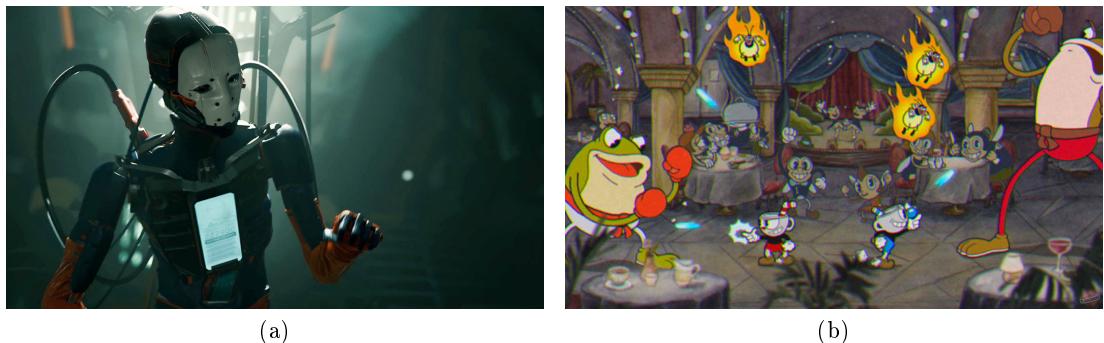


Figura 3.1: Esempi di due giochi creati con Unity: "Adam" (3D) [14] (a) e "Cuphead" in 2D [2] (b).

Nel caso di questo progetto è stato utilizzato il linguaggio C# ed ogni porzione di codice farà riferimento ad esso. Possono essere combinate all'interno dello stesso progetto, benché non sia buona norma.

Unity adotta due tipi di sotto iscrizioni per ottenere una licenza: *free* e *Pro*. La versione free, come si deduce dal nome, è gratuita perché pensata per un uso personale, mentre quella Pro è obbligatoria se l'organizzazione che la utilizza ha degli introiti superiori a 100.000 dollari. Sino alla versione "Unity 4", la licenza free imponeva grosse limitazioni sullo sviluppo delle applicazioni, ma dalla versione "Unity 5" non è più così, lasciando nella licenza Pro poche funzioni esclusive, come strumenti di profilazione o di condivisione del progetto all'interno del team di sviluppo.

Unity è stato utilizzato durante questo tirocinio poiché è risultato ottimale alla luce dei fattori chiave, considerati in termini di tempo e qualità, come mostrato nella figura sottostante.

	CryEngine	Unreal engine 4	Ogre	Unity3d	Project Anarchy
Entry level	Very high	Middle	High	Low	Middle
Language	C++/Lua	C++/ UnrealScript	C++	C#, JavaScrtipt, Boo	C++/Lua
Build-in AI system	Yes	Yes	No	No (in free version)	Yes
Community	Little	Big	Little	Very big	Medium
Price	\$9.90/month	\$19.90/month (free from March 2015) + 5% royalty	Free	Free; \$1500 or \$75/month (for Pro)	Free (\$500 for PC exporter)
Requirements for PC	High	High	Low	Medium	Medium
Graphic quality	Very high	Very high	Medium	Medium	Medium

Figura 3.2: Gli aspetti negativi sono colorati in rosso mentre quelli positivi sono colorati di verde. Il colore giallo viene usato per evidenziare funzioni accettabili ma che richiedono un maggiore investimento monetario o di tempo. [9]

3.1.1 L'Editor di Unity

Questo sottocapitolo fornisce le informazioni che concernono l'editor. Le principali sono sei [13]:

Il progetto: In questa finestra è possibile gestire e accedere alle risorse del progetto. Il pannello a sinistra mostra la struttura delle cartelle, mentre nel pannello di destra vengono mostrate le risorse interne alla cartella selezionata. Da notare come la radice della struttura sia una cartella particolare, auto generata con l'inizializzazione del progetto e denominata "Assets". Grazie alla toolbar situata in alto si possono cercare elementi all'interno del progetto.

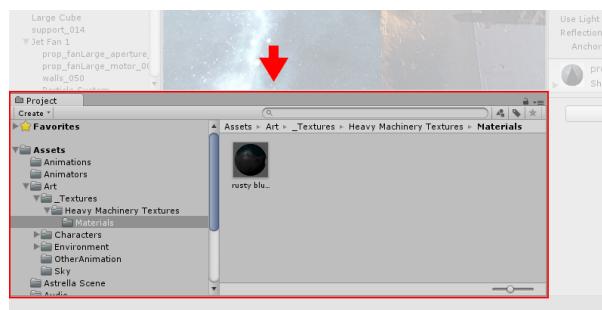


Figura 3.3: La finestra di progetto.

La scena: La finestra di scena ("Scene View") fornisce una vista interattiva sul mondo virtuale che si sta creando. Da qui si possono posizionare, ruotare, scalare elementi come modelli 3D, camere, luci, e altri tipi di *GameObjects* che analizzeremo più avanti.

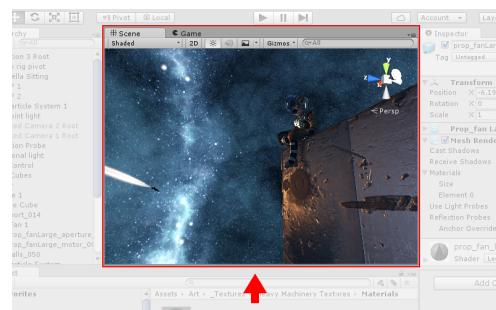


Figura 3.4: La finestra contenente la scena.

L'organigramma: Questo pannello contiene una lista di tutti gli elementi presenti nella scena corrente. Di default gli oggetti sono in ordine di inserimento, ma possono essere riordinati manualmente. Quando si annidano oggetti creando dei gruppi, quello a livello più alto viene definito *parent*, mentre quelli al suo interno sono chiamati *children*.

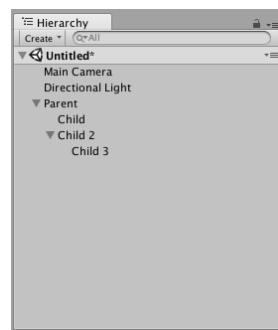


Figura 3.5: La finestra con la gerarchia.

L'inspector: I progetti in Unity sono formati da molteplici GameObjects, i quali contengono scripts, suoni, meshes (modelli 3D), e altri elementi grafici come luci o canvas. La finestra denominata *Inspector* permette la visualizzazione di sudette componenti sul GameObject selezionato. Le proprietà si possono vedere in dettaglio e modificare da questa finestra degli oggetti sulla scena o delle risorse non ancora istanziate nel mondo virtuale

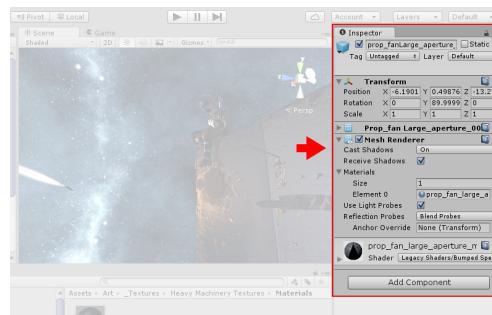


Figura 3.6: La finestra dell'Inspector.

La toolbar: La toolbar consiste in sette controlli base, ognuno dei quali connesso a parti dell'editor. I più importanti sono i bottoni di *Play/Pause/Step*, che permettono l'inizio e la gestione della simulazione visibile nella finestra di gioco.



Figura 3.7: La finestra contenente i controlli principali.

Il gioco: In questa sezione viene renderizzata la telecamera principale presente nella scena, fornendo una rappresentazione di come verrà visualizzato il prodotto finale.



Figura 3.8: La schermata che renderizza un'anteprima dell'applicazione finale.

3.1.2 Architettura Software di Unity

In questa sottosezione verranno presentati brevemente le componenti software che stanno alla base del framework.

3.1.2.1 Elementi Base

MonoBehaviour: MonoBehaviour è la classe base di Unity. Tutti i nuovi script sono sottoclassi di quest'ultima poiché MonoBehaviour fornisce i metodi che implementano il ciclo vitale di un'applicazione Unity. Se una classe deriva da MonoBehaviour allora potrà:

- essere collegata agli oggetti Unity come una componente;
- fornire la possibilità di accedere alle sue variabili attraverso l'Inspector;
- supportare i metodi e gli eventi standard forniti da Unity come le funzioni Start(), OnGui(), Update() ecc.;

GameObject: GameObject è la classe principale per tutti gli oggetti esistenti in Unity. Ogni oggetto funge da contenitore che può avere al suo interno diversi elementi (componenti), le cui combinazioni possono conferire proprietà speciali. Ogni GameObject contiene di default la componente *Transform* che ne definisce la posizione, la rotazione e la scala. Ogni GameObject presente nella scena può essere attivo o inattivo, in questo caso gli script collegati ad esso non saranno eseguiti ed eventuali mesh non saranno visualizzate.

Scripts: Sono tutte le nuove classi che estendono MonoBehaviour. Quando uno script viene creato e connesso ad un oggetto ne diventa una componente. Quest'ultima verrà mostrata nell' Inspector assieme a tutte le variabili pubbliche contenute nella classe. Attraverso lo script è possibile accedere al GameObject e alle altre componenti ad esso connesse attraverso il metodo:

```
public Component GetComponent<Type type>()
```

ma non essendo un metodo molto veloce è consigliabile salvare nella cache la componente ritornata.

Prefab: Un prefab è un tipo di *risorsa* che memorizza un GameObject e tutte le relative componenti e proprietà e funge da modello dal quale istanziare copie dell'oggetto nella scena. Ogni modifica effettuata sul prefab verrà riflessa sulle sue istanze sovrascrivendole. L'utilizzo di prefabs è vivamente consigliato poiché riduce i tempi di generazione e accesso degli oggetti nella memoria.

Resources: Unity non opera solamente con i prefabs ma anche con le *resources* che possono essere texture, audio, video. La differenza con i prefabs è che una resource è sempre caricata nella scena così com'è e i cambiamenti apposti nell'editor non inficiano la resource iniziale, ma solo l'istanza modificata. Di default Unity crea resource di ogni asset presente nelle cartelle denominate "Resources" all'interno del progetto. Il contenuto di queste cartelle è sempre copiato nella build dell'applicazione.

Scene: Un'applicazione sviluppata con Unity non è altro che una composizione di scene, ognuna contenente gli ambienti e i menu. Si può pensare ad esse come a dei livelli, o più semplicemente delle schermate, che formano l'applicativo.

Packages: Unity permette l'esportazione di scene o oggetti in file, chiamati *packages*. Il contenuto di questi ultimi può essere importato in qualsiasi progetto Unity. L'esportazione include tutte le dipendenze dei singoli oggetti, ovvero se si volesse esportare una scena, tutti i modelli, texture e altri asset al suo interno verrebbero automaticamente inclusi. Questa particolarità ha fatto sì che Unity sviluppasse una grande community molto collaborativa.

3.1.2.2 Ciclo di Vita

In particolari momenti Unity chiama determinate funzioni, che ne compongono il ciclo vitale. Ogni script che estende MonoBehaviour può implementare o riscrivere questi metodi vitali. I più importanti sono descritte qui sotto.

I seguenti metodi sono chiamati all'inizio della scena, una volta per ogni oggetto nel frame:

Awake()

Il metodo Awake() viene chiamato quando l'istanza dello script viene caricata. È principalmente utilizzata per inizializzare variabili o stati prima che l'applicazione inizi ed è chiamata solamente una volta nel ciclo di vita dell'istanza. La sua chiamata viene effettuata solamente dopo che tutti gli oggetti nella scena sono stati inizializzati, perciò si può già comunicare con altri oggetti attraverso la funzione

```
public GameObject GameObject.Find()
```

Awake() non viene chiamato negli oggetti inattivi presenti nella scena. In particolare in C# il metodo Awake() è utilizzato per l'inizializzazione invece di un costruttore, dato che lo stato serializzato dello script sarebbe ancora indefinito alla chiamata di quest'ultimo.

OnEnable()

Questa funzione viene chiamata quando l'oggetto viene attivato. Utilizzata principalmente per la registrazione ad un evento.

Le seguenti funzioni vengono invece chiamate sul primo frame dopo che lo script è stato attivato:

Start()

Viene eseguito Start() subito dopo la funzione Awake(), ma prima di ogni altra porzione di codice della classe. Come la funzione Awake() viene chiamata una sola volta nel ciclo vitale dello script.

Al centro del ciclo vitale di Unity vi sono queste funzioni sottoelencate che vengono lanciate per l'intera vita dell'applicazione:

FixedUpdate()

Questa funzione viene chiamata ad ogni *fixed frame*, ovvero intervalli di tempo prestabiliti indipendenti dal frame rate dell'applicazione. Per questo motivo all'interno di questa funzione vengono gestite le componenti che computano la fisica della simulazione. Essendo le più dispendiose e le più importanti a livello visivo, esse così non vengono condizionate da eventuali rallentamenti nel gioco.

Update()

Il metodo Update() svolge un ruolo fondamentale nel ciclo vitale dell'applicazione ed è sicuramente il più usato. Viene eseguito una volta per ogni frame.

LateUpdate()

Simile al metodo Update() ma viene eseguito subito dopo che tutti i processi di quest'ultimo sono terminati.

Infine i seguenti metodi sono utilizzati per gestire la disattivazione o morte di oggetti.

OnDisable()

Chiamato quando un componente viene disattivato o reso inattivo, utilizzato maggiormente per cancellarsi dalle registrazioni degli eventi.

OnDestroy()

Chiamato nell'ultimo frame dell'esistenza dell'oggetto. Utilizzato per pulire la memoria da eventuali riferimenti.

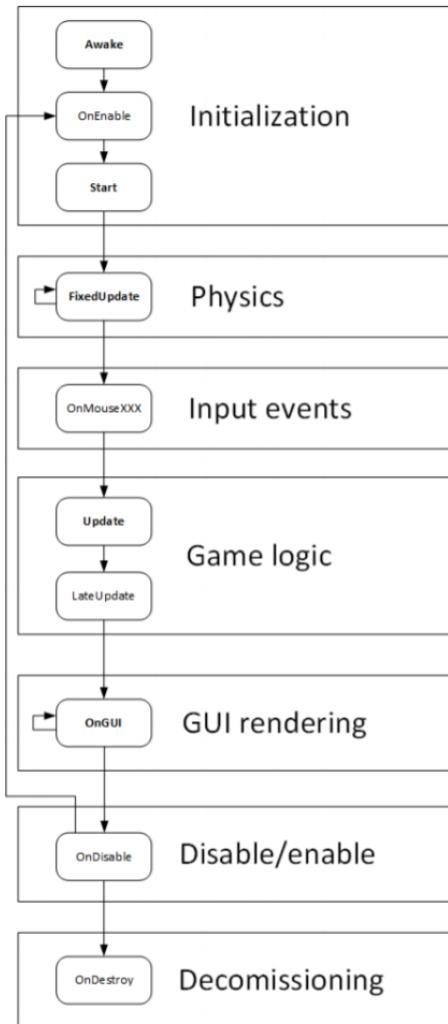


Figura 3.9: Questo schema mostra la sequenza delle chiamate alle principali funzioni di Unity.

3.2 Samsung Gear VR

Il *Samsung Gear VR* è un visore portatile per la realtà virtuale sviluppado da *Samsung Electronics* in collaborarzione con *Oculus*, e prodotto da *Samsung*. E' stato rilasciando nel Novembre 2015. Nell'utilizzo, un dispositivo compatibile (Galaxy Note 5, Galaxy S6/S6 Edge/S6 Edge+, Galaxy S7/S7 Edge, Galaxy S8/S8+ o Galaxy Note 8) funge da display e da processore, mentre l'unità Gear VR stessa funge da controller, contenente lenti focali e un'IMU. La connessione tra il visore ed il cellulare avviene tramite un connettore USB. Il Gear VR include anche un touchpad e bottoni fisici per la navigazione all'interno delle applicazione in caso non si possedesse un controller compatibile e un sensore di prossimità per determinare se l'utente lo sta indossando. La distanza focale in questo visore può essere calibrata attraverso una rotella posizionata nella parte superiore del prodotto, adattandosi all'utente. Le principali innovazioni apportate dal Gear Vr sono il raggiungimento di una latenza MTP(Motion to Photon) minore di 20ms e l'ottimizzazione dell'Hardware e del Kernel.

Unity forsnisce delle API di base per lo svilluppo su Gear VR senza l'utilizzo di plug-in esterni, che gestiscono la comunicazione tra una normale applicazione Android con il visore, trasformandola in un applicativo in realtà virtuale. Unity infatti gestisce automaticamente la renderizzazione delle camere presenti nella scena sul visore HMD (Head Mounted Display) generando le matrici di proiezione tenendo conto dei movimenti della testa e del campo visivo. E' invece compito dello sviluppatore preoccuparsi che il frame rate dell'applicazione non scenda sotto la frequenza di aggiornamento del display nel visore (60hz) nel caso del Gear VR o nascerebbero gli effetti collaterali precedentemente descritti. Per poter avviare applicazioni VR sui dispositivi Android compatibili va installato l'applicativo ufficiale Oculus Home che gestisce il lancio e l'esecuzione della nostra app.

Capitolo 4

Requisiti del Progetto

Questo capitolo tratta l’analisi dell’applicazione dal punto di vista della raccolta dei requisiti derivante dall’incontro con il cliente a cui va destinata.

4.1 Una Sala Riunioni in Realtà Virtuale

L’applicazione sviluppata e descritta in questa tesi è finalizzata alla creazione di sale riunioni virtuali a cui più utenti possono accedere contemporaneamente con l’utilizzo di un Samsung Gear VR. L’applicazione è pensata su un modello *enterprise*, ovvero il suo target principale non saranno i piccoli consumatori, ma grosse aziende che vogliono facilitare le riunioni tra dipendenti o gli incontri con i clienti. La creazione di una sala riunioni virtuale introdurrebbe i benefici dello smart working nei contesti aziendali. Lo scopo non è quello di sostituire gli attuali strumenti di teleconferenza (Skype, Google Hangouts, ecc.), ma di ampliarne il concetto, fornendo nuovi modelli e iniziando un processo di aggiornamento tecnologico.

Essendo il cliente di questo progetto lo stesso sviluppatore e produttore dei visori, vuole che l’applicazione funga da fattore chiave nella scelta dei loro dispositivi da parte di grandi aziende, andando ad ampliare le loro possibilità con nuovi strumenti e facendole sentire parte della rivoluzione tecnologica.

4.2 Principali Funzionalità e Casi d’Uso

L’applicazione finale dovrà contenere molte funzionalità, andando ad imitare e ad ampliare gli strumenti di una comune sala riunioni. Una volta avviata l’applicazione, l’utilizzatore si troverà in un ambiente virtuale dal quale, attraverso l’interazione del controller con menu posti su dei pannelli nel mondo virtuale, l’utente sarà in grado di accedere al suo account personale. Una volta effettuato l’accesso, potrà personalizzare il proprio profilo (avatar, nome, notifiche,ecc.), creare una riunione con la relativa sala e invitare altri utenti a partecipare. All’interno di questo spazio ogni utente sarà rappresentato da un avatar virtuale che seguirà i reali movimenti della testa dei partecipanti. Per tutta la durata della conferenza sarà attiva la chat vocale che permetterà una comunicazione rapida e naturale tra gli utenti. Nella sala riunioni essi avranno la possibilità di visualizzare documenti, sia personalmente che in condivisione su un proiettore; potranno inoltre modificarli sottolineandoli o evidenziandone le parti importanti e dividerli tra i partecipanti alla riunione. Di questa parte è importante la gestione sicura dei dati che transiteranno sull’applicazione e sui server che ne gestiscono la componente in rete, poiché le grandi corporazioni i principali destinatari d’uso, la sensibilità dei documenti è elevata. Sebbene molte di queste funzioni non siano ancora implementate, è importante elencarle perché sono state tenute in considerazione e hanno condizionato lo sviluppo della versione dimostrativa prodotta durante questo tirocinio.

Di seguito il caso d’uso dell’applicazione dimostrativa:

1. l’utente avvia l’applicazione e inserisce il dispositivo nel visore;
2. l’utente si ritrova in un ambiente virtualizzato;
3. l’utente sceglie il proprio avatar (maschile o femminile);
4. l’utente inserisce il nome che verrà visualizzato sopra al proprio avatar nell’effettiva sala riunioni con l’ausilio di una tastiera virtuale;
5. l’utente conferma il nome inserito;
6. l’utente avvia una riunione;

7. il sistema si collega ad un server creando una stanza pubblica;
8. il sistema si collega ad un server per gestire la chat vocale;
9. l'utente si trova in una sala riunioni virtualizzata assieme ad eventuali altri partecipanti, che hanno compiuto i passi 1-8, e ne visualizza l'avatar;
10. attraverso un menu posto di fronte all'utente, quest'ultimo può visualizzare i documenti presenti sul cellulare e proiettarli sullo schermo posto nella sala riunioni;
11. l'utente può disconnettersi dalla sala riunioni e ritornare alla stanza iniziale;
12. il sistema chiude la connessione con i server;

4.3 Esperienza d'uso

Essendo un'applicazione che verrà utilizzata in contesti lavorativi o di formazione, è fondamentale ottimizzare l'esperienza d'uso dell'utente. Grande attenzione infatti è stata posta sulla fluidità con la quale l'applicazione traccia i movimenti della testa. Quest'ultimo fattore infatti è ciò che più contraddistingue la Virtualized Meeting Room dagli altri strumenti di teleconferenza, nei quali un mancato approccio visivo tra i partecipanti può creare confusione, rendendo difficile seguire o intervenire in un discorso. Anche la qualità dei documenti gioca un ruolo fondamentale, poiché la visualizzazione e la loro condivisione deve essere il più naturale possibile ricreando la sensazione di una vera riunione. Lo stesso discorso si può applicare alla chat vocale che gioca il ruolo più importante nella comunicazione all'interno della stanza.

Capitolo 5

Progettazione e Sviluppo

Questo capitolo tratta il passo successivo, ovvero la progettazione e lo sviluppo dell'applicativo facendo riferimento al caso d'uso descritto. Per motivi di tempo legati alle richieste del cliente, nell'applicazione dimostrativa vengono tralasciate le questioni che concernono la sicurezza dei dati.

5.1 Ambientazione Grafica

Particolare attenzione è stata posta alla progettazione dell'ambientazione 3D e alla scelta dei modelli. Essendo infatti l'utente completamente immerso nella sala virtuale, la grafica di quest'ultima assume grande rilevanza, in quanto può migliorare o rovinare l'esperienza. Ho infatti progettato l'ambiente insieme a 3D Maurizio Poccia, collega addetto alla modellazione 3D, utilizzando un approccio molto simile a quello impiegato nello studio di design d'interni.

Siamo partiti da immagini di riferimento rappresentanti sale riunioni moderne, senza particolari troppo caratteristici, mantenendo uno stile neutro e impersonale, soprattutto nella scelta dei colori e dell'arredamento virtuale.

Durante lo sviluppo di applicazioni per il Gear VR è fondamentale tenere in considerazione la limitata potenza di calcolo dei cellulari compatibili, in buona parte già impegnata nel gestire la simulazione virtuale. Come dispositivo per i test infatti è stato utilizzato il cellulare meno potente in termini di processore, ovvero il Samsung S6. Per evitare effetti sgradevoli su questi dispositivi è necessario ottimizzare al meglio l'applica-

cazione e soprattutto i modelli 3D.

Per prima cosa bisogna eliminare dalle *mesh* ogni faccia geometrica dalle che non verrà mai visualizzata, poiché non vogliamo sprecare risorse renderizzando qualcosa che non verrà mai visto. Anche i modelli devono contenere il numero minimo di poligoni (*low poly*) e di dettagli semplificando al massimo le mesh. Nel caso del nostro progetto la sfida risiedeva nell'ottenere un'ambientazione che pur essendo low poly, avesse uno stile realistico e formale, in contrasto con il tono "cartoonesco" dei modelli 3D con pochi poligoni. Questo è stato possibile attraverso l'aggiunta di dettagli, anziché sui modelli, sulle texture, utilizzando *normal* e *bump maps* di alta qualità e tramite la creazione di *atlas* che unificassero queste ultime riducendo i loro tempi di carico e scarico sulla memoria.



Figura 5.1: Visualizzazione della stanza non illuminata con il *wireframe* in evidenza (21k triangoli, 30k vertici).

Un altro effetto che ho ottimizzato nella creazione della scena è l'*overdraw*, ovvero quando oggetti sono renderizzati gli uni di fronte agli altri, sprecando le risorse della GPU. Per evitare che ciò accada tutti i modelli che non subiranno trasformazioni devono essere settati come *static*, in particolare *occluder static* e *occludee static* in modo da poter generare dall'apposito pannello una mappa che contiene i dati relativi all'occlusione. Questi ultimi permettono anche di risparmiare non renderizzando gli oggetti fuori dal campo visivo (*Occlusion Culling*).

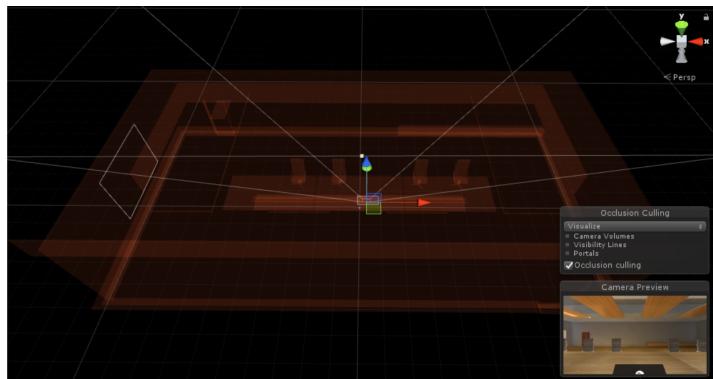


Figura 5.2: Visualizzazione dell'*overdraw*: nei punti in cui i modelli vengono sovrapposti, l’arancione si fa più intenso. Da notare l’assenza delle poltrone fuori dal capo visivo della telecamera (linee bianche).

Un altro strumento che ho adoperato nel processo di ottimizzazione è il *raggruppamento delle chiamate alle API grafiche* (*Draw Call Batching*), spesso molto esigenti dal punto di vista delle risorse. Unity aiuta gli sviluppatori offrendo due strumenti:

- Raggruppamento dinamico (*Dynamic Batching*): ottimale per piccole mesh, i cui vertici simili vengono raggruppati dalla CPU e renderizzati una volta sola. Se utilizzato eccessivamente può appesantire troppo la CPU.
- Raggruppamento statico (*Static Batching*): combina GameObject statici in mesh più grandi, rendendo più veloce il processo di rendering. Può aumentare i costi di accesso memoria se utilizzato inopportunamente.

Solamente oggetti che condividono gli stessi materiali possono essere raggruppati insieme, per questo ho cercato di utilizzarne pochi in tutta la scena creando delle *texture atlases* in cui più texture sono unite formandone una più grande. Se i GameObject rispettano questo vincolo e non hanno *mirroring* (un oggetto con scala +1 ed un altro con scala -1) nelle componenti *transform* vengono raggruppati dinamicamente in automatico. Invece il vincolo affinché un oggetto venga raggruppato staticamente nella fase di rendering è che sia statico.

Probabilmente la parte che più grava sulle performance di un'applicazione è quella relativa all'illuminazione e questo è ancora più importante se si parla di realtà virtuale. L'illuminazione infatti deve essere calcolata precedentemente, poiché un approccio real time sarebbe troppo dispendioso per un dispositivo mobile. Quando questi atlanti dell'illuminazione (*lightmaps*) sono calcolati a priori, gli effetti della luce sugli oggetti statici sono scritti sulle texture che verranno applicate alle geometrie, riproducendo l'illuminazione desiderata. Queste lightmaps possono includere sia la luce diretta che colpisce le superfici, sia quella indiretta che rimbalza tra le geometrie. L'unico limite è che le luci non possono cambiare a tempo di esecuzione.

Importanti ai fini di snellire il processo d'illuminazione della scena sono i *light probes*. La loro unica differenza con le lightmaps è che invece di immagazzinare e usare le informazioni riguardanti la luce che colpisce le superficie, gestisce la luce che passa per gli spazi vuoti nell'ambientazione, conferendo alla scena un aspetto più naturale.

Infine per dare l'ultimo tocco di realismo alla scena ho utilizzato un *reflection probe*, ovvero uno strumento che funge da camera che cattura un'immagine sferica di ciò che la circonda. Questa immagine è immagazzinata in una *cubemap* la quale fornisce informazioni ai materiali che riflettono o sono specchiati.



Figura 5.3: L'ambientazione finale con la corretta illuminazione.

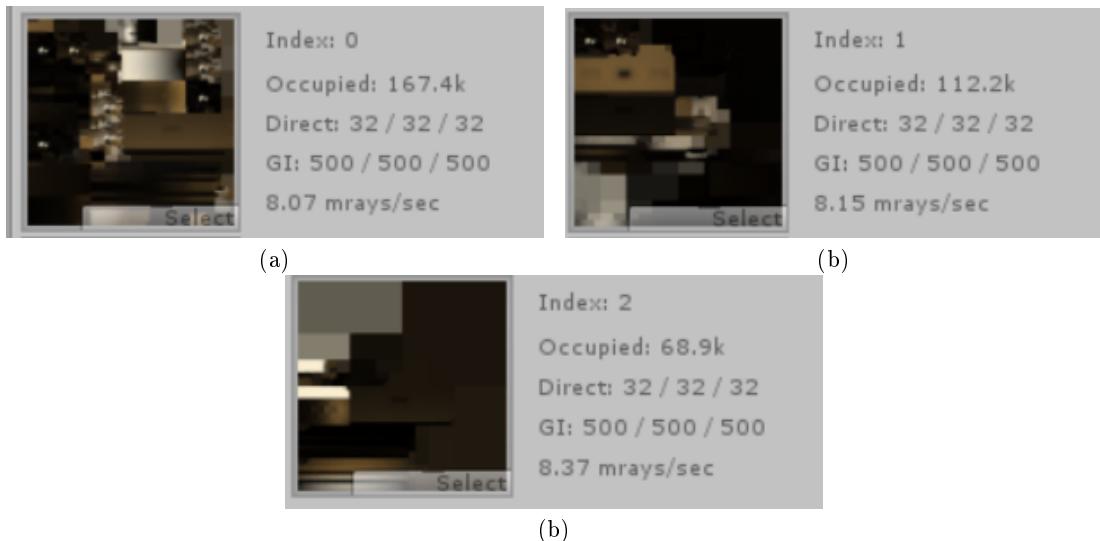


Figura 5.4: Le tre lightmaps computate da Unity contenenti le informazioni sulla luce salvate in forma di texture atlases.

L'utilizzo di un filtro *anti-aliasing* è altamente consigliato se si vogliono sviluppare applicazioni VR, poiché aiuta a "levigare" l'immagine evitando la presenza di bordi frastagliati. In particolare per le esperienze con i Gear VR è quasi obbligatorio l'adozione di un filtro MSAA, ai fini di ottenere una buona esperienza per l'utente.

Infine per rendere efficiente la computazione della GPU si deve tenere in considerazione che tipo di *shader* si utilizzano. Essi sono piccoli script che contengono i calcoli matematici e gli algoritmi che computano il colore di ogni pixel renderizzato basandosi sulle informazioni della luce e dei materiali. Possono fungere da collo di bottiglia nel processo di ottimizzazione. Nel caso di questo progetto mi sono servito degli *shader* già presenti all'interno di Unity specializzati nell'operare su dispositivi mobili, in particolare il tipo *unlit* che non gestisce la parte relativa alla luce, lasciando il compito alle lightmaps.

5.2 Interfaccia grafica

Quando si progetta un’interfaccia grafica per un’esperienza VR ci sono diversi fattori da tenere in considerazione. La risoluzione dei Gear VR è di 2560 x 1440 (1280 x 1440 per occhio) e può portare ad un effetto di *pixelation*, ovvero quando i singoli pixel diventano evidenti. Questo è particolarmente vero per elementi grafici di grandi dimensioni in cui è presente un testo.

Nelle applicazioni non immersive, la UI è spesso posizionata in cima e renderizzata per ultima davanti a tutti gli altri elementi, formando quello che si definisce un HUD (*Heads Up Display*). Questo tipo di UI è riferita come un’interfaccia grafica *non diegetica*, ovvero che non esiste nel mondo, ma ha senso per l’utente nel contesto della applicazione. Un approccio simile si ha con la musica nelle opere cinematografiche. In Unity questo processo è applicabile scegliendo un *canvas* di tipo *overlay* o *camera* posizionato sullo spazio del display. In un’applicazione VR queste interfacce grafiche sono prive di senso, poiché seguirebbero il movimento della testa e le informazioni visualizzate sarebbero difficilmente messe a fuoco dai nostri occhi, data la vicinanza degli schermi a quest’ultimi.

Dobbiamo puntare quindi ad una UI *spaziale*, ovvero posizionata nello spazio del mondo virtuale ed indipendente dai movimenti del visore. Questo permette una navigazione visiva semplice e naturale tra gli elementi dei canvas. L’interfaccia grafica quindi va considerata come parte integrante del mondo e deve essere coerente con esso. È utile posizionare i canvas ad una distanza confortevole per la lettura, in quanto se troppo vicini all’utente potrebbero creare i problemi descritti in precedenza. Un ostacolo in cui si imbatte con questi tipi di UI è la perdita di controllo su cosa mostrare all’utente. È possibile collegare i movimenti della UI a quelli della testa dell’utente, in modo da tenere sempre

		Is the representation visualized in the 3D game space?	
		no	yes
Is the representation existing in the fictional game world?	no	non-diegetic representations	spatial representations
	yes	meta representations	diegetic representations

davanti ai suoi occhi le informazioni che deve visualizzare. Tuttavia questo metodo è consigliabile solo per elementi come puntatori o mirini che fanno concettualmente parte dell’utente e che gli permettono di interagire con il mondo, poiché potrebbe portare a una sensazione claustrofobica, per cui chi indossa il visore non può scrollarsi di dosso le informazioni ed è quindi obbligato a guardarle. E’ pur vero che essendoci totale libertà di rotazione della testa, non si è costretti a guardare nella giusta direzione e si potrebbero perdere informazioni o momenti importanti. Per risolvere questo inconveniente vanno progettati ambienti in cui fin dall’inizio è presa in considerazione la posizione dell’interfaccia grafica. Nel caso della Virtualized Meeting Room ho gestito il problema, più evidente nella fase iniziale in cui l’utente deve configurare il proprio profilo, oscurando tutta la stanza ed illuminando con dei riflettori i due avatar e il canvas con il quale l’utente deve interagire. Portiamo così l’utilizzatore a trovare in modo naturale le informazioni cruciali ai fini dell’esperienza.

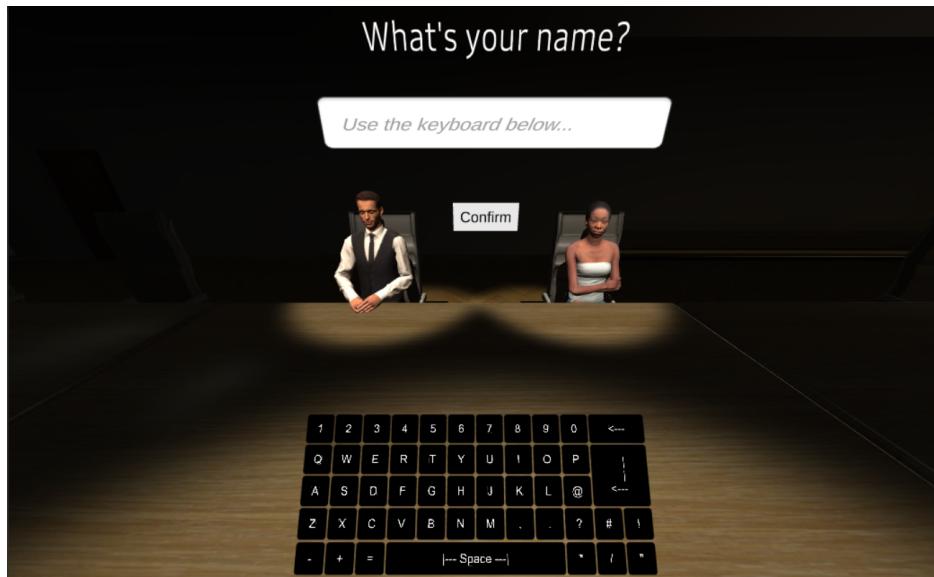
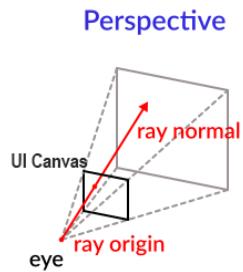


Figura 5.5: Passo quattro del caso d’uso preso in considerazione. L’assenza di elementi distraenti porta a interagire con l’UI.

Per gestire l’inserimento dei dati e delle preferenze dell’utente ho utilizzato un diegetico, in cui i componenti grafici sono immersi nella scena. La tastiera ad esempio è composta da un insieme di pannelli posizionati dinanzi all’utente. La scelta degli ava-

tar è gestita tramite l'inserimento di un canvas tra i modelli e l'utilizzatore. I documenti nella sala riunioni sono condivisi su un proiettore rendendo più naturale l'esperienza. Il nome dell'utente è visualizzato sopra il suo avatar assieme ad altre icone che indicano se sta parlando o se ha disattivato il microfono.

L'utente può interagire con questi menu in due modi: con l'utilizzo del controller ufficiale Gear VR (scelta consigliata); con l'utilizzo del *touchpad* presente sul visore e l'ausilio di un "mirino" che aiuta il puntamento. In entrambi i casi il processo di selezione è gestito attraverso dei *collider*, componenti che gestiscono le interazioni fisiche, posizionati sui canvas che rilevano se il puntatore *raycast* è orientato su di esse. Se ciò avviene e si preme il pulsante di selezione (il grilletto del controller oppure un tocco sul touchpad) si chiamerà il corrispettivo metodo che gestisce gli eventi associati al bottone. Ho generato una classe *UIController* singleton che permane tra le varie scene, addetta alla gestione di tutte le interazioni con l'interfaccia grafica.



5.3 Architettura Software

Ora che l'applicazione ha un'ambientazione virtuale ottimizzata e delle interfacce grafiche configurate è necessario implementare le funzionalità.

5.3.1 Librerie Esterne

Per facilitare lo sviluppo dell'applicazione, il progetto implementa quattro librerie esterne principali. Due di queste gestiscono il visore e forniscono degli strumenti aggiuntivi per la creazione di applicazioni in realtà virtuale (*Oculus SDK*, *Virtual Reality Toolkit*); le altre due si occupano della connessione ad un server e della comunicazione real-time tra i due Gear VR (*Photon Unity Networking*, *Photon Unity Networking Voice*).

Oculus SDK: integrata già all'interno di Unity, questa libreria si interfaccia con le componenti hardware del visore fornendo delle API di base che gestiscono: la

renderizzazione in modalità stereo dei display, i bottoni fisici o gli eventuali controller e il tracciamento dei movimenti del visore. [8]

"VRTK": è una libreria estremamente utile poiché fornisce molteplici script e prefab che semplificano e arricchiscono gli strumenti di base forniti dalla SDK ufficiale di Oculus e funge da *wrapper* di quest'ultima. [7]

Photon Unity Networking: gestisce la parte back-end dell'applicazione. Offre in un'opzione gratuita alcuni server per la gestione real-time degli eventi, con un limiti imposti sul numero di utenti connessi contemporaneamente. Utile perchè fornisce una soluzione *plug and play* adatta alle tempistiche del progetto dimostrativo. Per usufruirne bisogna registrare l'applicativo sul sito ufficiale. [7]

Photon Unity Networking Voice: libreria gemella di quella appena descritta che offre un servizio di *VOIP* da utilizzare all'interno dell'applicazione. [4]

5.3.2 Configurazione della Realtà Virtuale

Una volta importate le librerie descritte in precedenza e configurato il progetto per la realtà virtuale bisogna popolarlo con le componenti opportune. Come prima cosa deve essere creato un oggetto vuoto a cui va connesso lo script *VRTK_SDK Manager* che si occupa della configurazione delle SDK supportate. Questo oggetto è *parent* di un *GameObject* contenente lo script *VRTK_SDK Setup* nel quale va configurata la libreria da utilizzare, nel caso di questa applicazione quella relativa al Gear VR. A sua volta quest'ultimo *GameObject*, sarà *parent* del prefab fornito dalla SDK Oculus che gestisce il visore.

A questo punto è necessario connettere allo script *VRTK_SDK Manager* un'oggetto che fungerà da controller. Questo *GameObject* dovrà contenere la classe *ControllerEvents*, fornita dalla libreria VRTK, che gestisce gli eventi scaturiti dai diversi tipi di input. Oltre a questa ho implementato i seguenti script:

- *Pointer*: permette la creazione e la renderizzazione del puntatore.
- *UI Pointer*: permette le interazioni con gli elementi dell'UI.

- *Radial Menu*: permette di aumentare il numero di disponibili creandone alcuni virtuali e mostrandoli come un menu radiale renderizzato sopra il controller. Queste voci saranno rese visibili e accessibili scorrendo il dito sul touchpad del telecomando.



Per poter visualizzare il controller dobbiamo inserire una mesh come figlio di questo .

Al termine di questi passaggi si ha un'applicazione in realtà virtuale funzionante all'interno dei visori Gear VR, in cui l'utente può guardarsi attorno e selezionare le voci dai menu.

5.3.3 Gestione delle funzionalità in Rete

Questa risulta essere la parte più delicata del progetto poiché lo scopo principale dell'applicazione è quello di permettere a più utenti di ritrovarsi nella sala riunioni attraverso la rete.

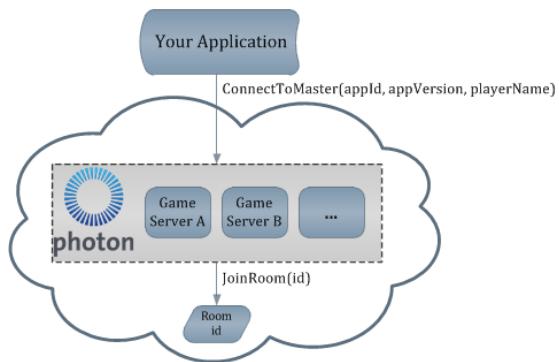


Figura 5.6: La gestione della connessione alla rete schematizzata.

Per prima cosa bisogna stabilire una connessione con il server e creare una stanza utilizzando la libreria Photon Unity Networking. Per fare ciò è necessario creare una classe *NetworkController* che gestisca questa parte.

Algorithm 1 Classe che gestisce la connessione con il server

```
public class NetworkController : MonoBehaviour {  
  
    private string m_roomName = "VMR_Test"  
  
    void Start() {  
        PhotonNetwork.ConnectUsingSettings("0.1");  
    }  
  
    void OnJoinedLobby() {  
        RoomOptions roomOpt = new RoomOptions() {maxPlayers = 4};  
        PhotonNetwork.JoinOrCreateRoom(m_roomName,roomOpt,TypedLobby.Default);  
    }  
  
    void OnJoinedRoom() {  
        PhotonNetwork.Instantiate("NetworkedPlayer",Vector3.zero,Quaternion.identity,0);  
    }

---


```

In questa classe definiamo una variabile di tipo stringa *m_roomName* che sarà il nome della stanza che creeremo sul server. Questo parametro funge da "chiave"; infatti con questa stringa altri utenti potranno accedere alla stessa stanza. Essendo un' dimostrativa non ci preoccupiamo di generare un nome univoco e sicuro, vogliamo anzi che tutti gli utenti finiscano nella stessa stanza.

Il metodo *ConnectUsingSettings* si connette ai server offerti da Photon e si unisce ad una lobby pubblica prendendo come parametro la versione dell'applicazione. Appena ci si connette alla lobby viene chiamata la callback *OnJoinedLobby* che sovrascriviamo nella classe *NetworkController*. Al suo interno infatti prima gestiamo le opzioni della stanza(numero di utenti massimi, visibilità, ecc.) poi la creiamo se non ne esiste un'altra con lo stesso nome, altrimenti proviamo ad unirci a quella esistente. Qualora riuscissimo ad entrare, verrà attivata un'altra callback: *OnJoinedRoom*. Nel nostro caso questo metodo istanzia a lato del server un prefab dell'utente (*NetworkedPlayer*) specificandone posizione e rotazione.

Dobbiamo ora produrre lo script NetworkPlayer che collegheremo ad un avatar 3D. Questa classe estende *PhotonMonoBehaviour* che ci fornisce gli strumenti necessari per la sincronizzazione real-time. Quando viene istanziata, disattiva localmente il modello 3D del proprio avatar, il quale altrimenti ostruirebbe la vista all'utente. Procede poi scegliendo il posto da occupare attorno al tavolo della sala riunioni a seconda del valore assunto dalla variabile *countOfPlayersInRooms* che indica quante persone sono già connesse. Se la connessione va a buon fine, attiva le due componenti relative alla chat vocale: *Photon Voice Speaker* e *Photon Voice Recorder*. Queste si collegano ad un server separato, e si occupano del funzionamento del servizio di VOIP, registrando la voce e inviandola a tutti i membri della stanza. In questo caso ho sovrascritto alcuni metodi della classe "Photon Voice Recorder" per inserire la funzionalità *push-to-talk*, con la quale solo con la pressione di uno specifico pulsante si può attivare la chat vocale, evitando fastidiosi rumori ambientali.

Un'altra componente fondamentale dell'oggetto è definita *PhotonView*. Grazie a questa infatti il server sa che deve aggiornare per tutti i client i movimenti e le rotazioni compiuti dal GameObject, che nel nostro caso sono proprio i movimenti del capo dell'utente.

A questo punto due utilizzatori possono entrare nella sala riunioni virtuale, vedere i rispettivi movimenti della testa e comunicare attraverso una chat vocale.

L'ultima parte che rimane da gestire è l'invio dei documenti ad altri dispositivi. Questa ultima funzione non rientra totalmente nelle funzionalità di un motore grafico ed è stata la parte più complessa del progetto poiché Unity non fornisce nessuna libreria né per navigare tra i dati del cellulare nè per l'invio di file sulla rete. Il mio approccio è stato quello di creare una classe "FileExplorer" che implementasse il codice nativo Android. Unity permette questo processo offrendo dei tipi di oggetto speciali denominati "*AndroidJavaObject*", i quali fungono da *wrapper* per una classe nativa. Con l'aiuto della classe *java.io.File* ho creato un semplice "esplora risorse" per navigare nelle cartelle del cellulare filtrando i file a seconda dell'estensione (pdf, doc, ecc.).

Algorithm 2 L'algoritmo che gestisce la ricerca dei file PDF trovati nella cartella passata come parametro.

```
AndroidJavaObject dir=GetExternalStoragePublicDirectory(DirectoryName);
AndroidJavaObject[] files=directory.Call<AndroidJavaObject[]>("listFiles");
List<AndroidJavaObject> filteredFiles = new List<AndroidJavaObject>();
for AndroidJavaObject file in files do
    if getFileExtension(file) == "pdf" then
        filteredFiles.Add(file)
    end if
end for
return filteredFiles;
```

Cliccando il documento che vogliamo trasferire si invia il percorso del file a alla classe "FileConverter". Questa utilizza le classi *nativeandroid.os.ParcelFileDescriptor*, *java.io.FileOutputStream*, *java.io.File*, *android.graphics.pdf.PdfRenderer*, *android.content.Context*, *android.graphics.Bitmap* per leggere il file e convertirlo nella rispettiva bitmap.

Quest'ultima si può finalmente inviare agli altri dispositivi all'interno di Unity utilizzando una *RPC (Remote Procedure Calls)* ovvero degli eventi forniti da *Photon* che chiamano metodi su client remoti (ma nella stessa stanza). A questo punto ogni client riceve in forma parametrica la bitmap, la quale viene riscritta sotto forma di texture dalle classi questa volta interne a Unity.

Qualora si volesse condividere sul proiettore un documento, premendo l'apposita voce nel menu spaziale, verrà attivata una RPC che mostrerà a tutti i presenti la stessa immagine sullo schermo.

Algorithm 3 L'algoritmo che gestisce l'invio dei dati con una chiamata RPC

```
[PunRPC]
void ReciveByteArray(byte[] data) {
    Texture2D tex = new Texture2D(2000, 658, TextureFormat.PVRTC_RGB4, false);
    tex.LoadRawTextureData(data);
    Sprite sprite = Sprite.Create(tex, new Rect(0.0f, 0.0f, tex.width, tex.height), new Vector2(0.5f, 0.5f), 100.0f);
    return sprite
}
```

Algorithm 4 L'algoritmo che gestisce la creazione di una Texture partendo dalla relativa Bitmap passata come parametro.

```
MemoryStream msImage = new MemoryStream();
    _bitmap.Save(msImage, _bitmap.RawFormat);
newTexture.LoadImage(msImage.ToArray());
SpriteRenderer newSprite;
newSprite.material.maiTexture = newTexture;
return newSprite;
```

A questo punto dello sviluppo l'applicazione implementa tutte le funzionalità descritte nel caso d'uso citato in precedenza. L'utente, avviata l'applicazione e indossato il visore, si ritrova immerso in un ambiente virtuale nel quale ha la possibilità di personalizzare le caratteristiche base del proprio profilo. Può inoltre creare una sala riunioni o connettersi ad una già esistente. In quest'ultima egli è rappresentato da un avatar che segue i suoi movimenti e può parlare o condividere documenti con altri utenti.

Conclusioni e sviluppi futuri

L'obiettivo di questo tirocinio è stato quello di creare un'applicazione compatibile con i visori Gear VR che permetesse a più utenti di interagire, parlare e condividere documenti in un unico ambiente virtuale.

Durante la fase di sviluppo e la progettazione ho acquisito delle competenze tra le quali la conoscenza dei processi d'ottimizzazione di un'applicazione VR all'interno del motore grafico Unity, le implementazioni di librerie che gestiscono la parte relativa al *networking* di un'applicazione in realtà virtuale e il funzionamento di diverse classi native Android.

Il tirocino è avvenuto presso l'azienda Fingerlinks S.r.l. per la quale lavoro. Mi sono occupato dell'intera progettazione e sviluppo dell'applicazione, imparando ad interfacciarmi con un cliente di grande prestigio.

Per quanto riguarda gli sviluppi futuri, oltre all'implementazione delle funzioni già richieste dall'utente e descritte nel capitolo quattro, si potrebbe pensare di integrare la libreria Samsung *Knox* per aumentare la sicurezza nella gestione dati all'interno dell'applicazione. Inoltre il prodotto finale dovrebbe essere accompagnato da un'altra applicazione nativa che gestisca gli inviti alle riunioni e la selezione dei documenti da condividere, così da snellire la struttura dell'applicazione in realtà virtuale lasciando l'utente più libero di immergersi nella simulazione.

Concludo affermando che, in base alle aspettative circa lo sviluppo di questo software, gli obiettivi prefissati sono stati ampiamente soddisfatti.

Bibliografia

- [1] Applications of virtual reality.
- [2] M. Corp. Cuphead.
- [3] C. N. David Banister. The cost of transport on the environment - the role of teleworking in reducing carbon emission. 2007.
- [4] E. Games. *Photon Unity Networking*.
- [5] S. Jonathan. *Defining Virtual Reality: Dimensions Determining Telepresence*. 1992.
- [6] S. M. LaValle. Virtual reality. In *Virtual Reality*, 2017.
- [7] S. S. Ltd. *Virtual Reality Toolkit*.
- [8] L. Oculus VR. *Oculus Documentation*.
- [9] A. Penzentcev. Architecture and implementation of the system for serious games in unity 3d.
- [10] R. K. L. N. Sebastian Deterding, Dan Dixon. From game design elements to gamefulness: Defining gamification. 2011.
- [11] R. K. L. N. Sebastian Deterding, Dan Dixon. Leisure luxuries and the labor supply of young men. 2017.
- [12] A. R. G. Simon J. Watt, Kurt Akeley. Achieving near-correct focus cues in a 3-d display using multiple image planes.

- [13] U. Technologies. *The Main Windows.*
- [14] U. Technologies. Unity - adam.
- [15] U. Technologies. Unity supported platforms.
- [16] U. Technologies. Unity system requirements.
- [17] Wikipedia. Unity (motore grafico).
- [18] M. L. y and S. U. Pat Hanrahan, Computer Science Department. Achieving near-correct focus cues in a 3-d display using multiple image planes.