ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

UN FRAMEWORK PER LO SVILUPPO DI INTERFACCE UTENTE INNOVATIVE IN APPLICAZIONI HANDS-FREE BASATE SU SMARTGLASS: IDEAZIONE E SPERIMENTAZIONE IN UN CASO DI STUDIO

Elaborato in
PROGRAMMAZIONE DI SISTEMI EMBEDDED

Relatore
Prof. ALESSANDRO RICCI

Co-relatore
Ing. ANGELO CROATTI

Presentata da
FEDERICO MARINELLI

Prima Sessione di Laurea
Anno Accademico 2015 – 2016

PAROLE CHIAVE

sistemi embedded

smart-glass

human-compuer interaction hands-free

interfaccia utente smart-glass

mixed-reality blackboard

Alla mia famiglia e ai miei amici

Indice

Introduzione

La legge di Moore afferma che " la complessità di un microcircuito, misurata ad esempio tramite il numero di transistor per chip, raddoppia ogni 18 mesi". Partendo da questo presupposto è possibile evincere che nel corso degli ultimi decenni le tecnologie informatiche hanno subito numerose ed incredibili evoluzioni. In particolare, negli ultimi anni, stiamo assistendo alla comparsa di dispositivi embedded sempre più piccoli e portatili, tanto da poter essere indossabili dall'utente che ne vuole far uso. Questi tipi di dispositivi hanno raggiunto capacità di calcolo paragonabili a quella di un computer desktop di media fascia ed hanno cambiato radicalmente il modo con cui un utente medio si interfaccia con essi. L'obiettivo di questa tesi è stato inizialmente lo studio dei dispositivi wearable, in particolare degli Smart-glass, che rappresentano la versione embedded di un semplice paio di occhiali. In particolare si è voluto produrre un interfaccia utente innovativa che ha come scopo quello di semplificare l'interazione dell'utente con un dispositivo di tipo smartglass, avvicinando l'esperienza di utilizzo a quella che è la normale esperienza visiva dell'indossatore, così da rendere il più possibile semplice, utile e funzionale l'utilizzo di questo tipo di tecnologie. Seguendo questa linea guida si è voluto inoltre studiare un modo per interagire con questi sistemi tramite un processo di human-computer interacion hands-free, quindi che non prevedesse l'utilizzo delle mani. L'obiettivo di questo progetto di tesi è quindi quello di ideare e sviluppare un nuovo modo per interfacciarsi con queste nuove tecnologie emergenti, analogo a quanto successo negli anni passati con l'avvento degli smartphone.

Capitolo 1

Sistemi Embedded Wearable

Attraverso il termine Embedded System è possibile definire tutti quei dispositivi elettronici di elaborazione specializzati, integrati in un dispositivo fisico, in modo da controllare le funzioni tramite un apposito programma software dedicato.

Questi tipi di sistemi sono costituiti per funzionare in maniera continuativa, ricevendo informazioni dal mondo fisico e calcolando risposte in output. Tutti gli input vengono ricevuti dalla realtà che circonda il sistema stesso attraverso oppurtuni sensori, quali il girorcopio, l'accelleromentro, il magnetomentro, la fotocamera oppure il gps. Il dispositivo elabora queste informazione attraverso una componente software e fornisce delle risposte in ouput che possono essere ad esempio delle informazioni impresse su uno schermo LCD oppure informazioni espresse attraverso attuatori fisici.

1.1 Sistemi Wearable

All'interno di questo progetto di tesi non descriverò tutte le varie tecnologie embedded presenti al giorno d'oggi ma incentrerò la mia trattazione su un caso particolare di sistemi, ossia tecnologie wearable che permettono l'utilizzo della realtà aumentata in un contesto dove, tali dispositivi, possono ricevere input di controllo da parte dell'utente senza l'uso diretto e costante delle mani, più comunemente definiti hands-free.

I dispositivi wearable, la cui definizione potrebbe essere "anything that can be put on and adds to the user's awareness of his or her environment" [1], sono dispositivi elettronici che vengono indossati direttamente dall'utilizzatore e possono svolgere svariati compiti. Alcuni esempi possono essere gli smartwatch, gli smart-glass, bracciali per il fitness che presentano, ad esempio, un indicatore del battito cardiaco e del numero di passi percorso nell'arco della giornata.

Un'importante caratteristica che un device wearable deve possedere è la persistenza e facilità di accesso dei dati. Il dispositivo deve essere ideato e progettato per l'uso quotidiano, e deve essere sempre possibile per l'utente interagire con esso senza distogliere l'attenzione dalle attività che sta svolgendo [2]. Questi tipi di sistemi presentano attualmente anche alcune problematiche legate al funzionamento e al suo utilizzo nel lungo periodo. La scelta della fonte di energie, i problemi sulla dissipazione del calore sono degli esempi. Tra i vari dispositivi indossabili il mio lavoro ha posto il focus su uno in particolare: gli smart-glasses utilizzati in un contesto di realtà aumentata.

1.2 Smart-Glasses

Negli ultimi anni stiamo assistendo ad un progresso tecnologico non indifferente per tutto ciò che riguarda le tecnologie wearable. Questi tipi di dispositivi trovano sempre più applicazioni nei più svariati ambiti, ad esempio nello sport, nell'educazione oppure nel mondo del lavoro.
Oggigiorno gli ambienti enterprise stanno divenendo terreno fertile per lo studio, la progettazione e lo sviluppo di sistemi riguardanti, in particolare, il concetto di mobile augmented reality (MAR). Un esempio di dispositivo che sicuramente sta suscitando molte attenzioni in tale contesto sono gli Smart-glass.
Questi tipi di sistemi informativi possono essere utilizzati sia per realizzazione di altri sistemi e il testing degli stessi, ma anche per migliorare l'efficienza, la produttività e l'efficacia del processo di produzione.
Purtroppo questi tipi di sistemi non sono ancora diffusi in grande scala, ma rappresentano comunque un caso di studio molto interessante per molte aziende e ricercatori.
Sebbene esistano diversi dispositivi disponibili nel mercato, questi presentano ancora molte limitazioni sia dal punto di vista hardware che software. Dal punto di vista software, soprattutto, ancora non si è riuscito a realizzare un'interfaccia utente che permetta una buona user experience e quindi la possibilità di poter utilizzare ed usufruire appieno di tutte le caratteristiche e potenzialità che tale sistema dispone.

1.2.1 Caratteristiche degli Smart-glasses

Gli smart-glass sono degli occhiali tecnologici indossabili che aggiungono delle informazioni a ciò che l'indossatore vede e percepisce. Le informazioni vengono proiettate su uno schermo semiriflettente e semitrasparente, in modo che l'utilizzatore sia in grado di vedere il mondo reale con sovraimpressi gli

oggetti virtuali. Questo tipo di tecnologia viene chiamata optical see-through diaplay. Tipicamente questo tipo di risultato viene raggiunto attraverso l'utilizzo di schermi ottici head-mounted (OHMD, Optical Head-Mounted Display) oppure grazie all'utilizzo di head-up display, conosciuti anche come HUD. Quest'ultimi sono dei display trasparenti che rappresentano le informazioni all'utilizzatore senza che quest'ultimo guardi al di fuori dei propri "viewpoints".

Dal punto di vista dell'evoluzione di tali tecnologie, mentre i primi modelli erano in grado di eseguire solo delle operazioni di base, come ad esempio quello di mettere a disposizione dell'utilizzatore un semplice display front-end per interfacciarsi con un sistema remoto, i dispositivi attuali sono molto più efficaci, dispongono di un proprio sistema operativo e le capacità di tipo hardware possono essere paragonate a quelle degli smart-phone di ultima generazione. Difatti essi sono in grado di raccogliere informazioni dai sensori interni o esterni; supportano le ultime tecnologie senza fili come Bluetooth, Wi-Fi e GPS; inoltre, molti degli ultimi modelli presentano il sistema operativo Android.

1.2.2 Classificazione di Smart-Glasses

Oltre alle caratteristiche presentate precedentemente, è possibile suddividere il display di tipo optical see-through degli smart-glass come segue:

Tipologia display

- Display stereoscopico: questo tipo di display è dotato di due schermi, ognuno corrispondente ad un singolo occhio dell'utilizzatore. Questo tipo di display è in grado di creare l'effetto di tridimensionalità dei contenuti che l'utente vede all'interno del proprio campo visivo.

- Display singolo: questo tipo di display può essere visto solo da uno dei due occhi dell'utilizzatore oppure può essere condiviso da entrambi. Gli oggetti virtuali vengono presentati all'utente, quindi, ad entrambi gli occhi come unica immagine, e ciò rende tali oggetti meno realistici.

Immersività display

- Display immersivo: possibilità di sovrapporre oggetti virtuali alla realtà all'interno del campo visivo dell'utente. Solitamente ha un campo visivo abbastanza ampio ed è posto centralmente rispetto al field of view dell'utente.

- Dispaly non immersivo: non vi è la possibilità di sovrapporre oggetti virtuali alla realtà all'interno del campo visivo dell'utente. Il display è solitamente disposto nella periferia del campo visivo dell'utente ed ha un campo di vista di dimensioni limitate.

1.2.3 Alcuni esempi di Smart-Glasses

Attualmente, alcune importanti aziende tra cui Google, Epson, Vuzix, Sony e Microsoft stanno proponendo le prime soluzione prototipali basate su smart-glass con target enterprise e consumer. Di seguito elencherò gli smart-glass con maggior notorietà riscontrata nell'ultimo anno.

Google Glass I Google Glass sono stati il primo prototipo di smart-glass ad ottenere un importante impatto mediatico. La prima versione per sviluppatori è stata rilasciata nel mese di Aprile del 2013. Successivamente, a partire da Maggio del 2014 il dispositivo è divenuto disponibile nel mercato statunitense. Nel gennaio 2015 Google ha deciso di ritirare il prodotto dal mercato ma ha subito avviato collaborazioni con aziende partner per progettare una nuova versione del dispositivo.

Figura 1.1: Google Glass Immagine

Le caratteristiche sono le seguenti:

- Display: 14 gradi FOV (Field of View), singolo. Il display dei Google Glass non è in grado di sovrapporre contenuti virtuali ad oggetti reali,

infatti esso è disposto nella periferia del campo visivo dell'utente ed è dotato di un campo di vista di dimensione limitata(non immersivo).

- Prezzo: 1500 dollari.
- Target di mercato: consumer ed enterprise.
- Acquistabile: no.

Vuzix Wrap 1200DXAR Vuzix è un'azienda leader nel settore di smart-glass. I vuzix Wrap 1200DXAR sono pensati per applicazioni di realtà aumentata immersiva. Essi dispongono di una fotocamera stereoscopica binoculare che presenta un ampio campo di vista.

Figura 1.2: Immagine Vuzix Warp 1200DXAR

Le caratteristiche sono le seguenti:

- Display: 35 gradi FOV (Field of View) stereoscopico ed immersivo.
- Acquistabile: sì.
- Prezzo: 1500 euro circa.
- Target di mercato: enterprise.

Sony SmartEyeglass Questo dispositivo, rilasciato dall'azienda Sony si tratta di un occhiale di realtà aumentata orientato sia al mondo consumer che enterprise.

Figura 1.3: Immagine Sony SmartEyeglass

Le caratteristiche sono le seguenti:

- Display: 20 gradi FOV (Field of View) stereoscopico ed immersivo.
- Acquistabile: sì.
- Prezzo: 800 euro circa.
- Target di mercato: cosnumer ed enterprise.

Microsoft Hololens Questo dispositivo, prodotto dall'azienda Microsoft a partire dal 2016, è un head-mounted display di realtà aumentata orientato si per il mondo enterainment, che per il mondo del lavoro e dell'industria. Grazie a questo dispositivo è possibile visualizzare sovraimpressi al mondo reale degli oggetti di tipo virtuale in tre dimensioni, definiti dall'azienda stessa come ologrammi [3].

Figura 1.4: Immagine Microsoft Hololens

Le caratteristiche sono le seguenti:

- Display: stereoscopico ed immersivo.
- Acquistabile: sì.
- Prezzo: 3000 euro circa.
- Target di mercato: cosnumer ed enterprise.

DAQRI Smart Helmet Questo dispositivo, rilasciato dall'azienda DAQRI, è stato progettato ed ideato specificatamente per il mondo enterprise. Esso è equipaggiato di una videocamera 3D ad alta risoluzione ed un set di videocamere che permettono la navigazione e visione di realtà aumentata a 360 gradi.

Figura 1.5: Immagine DAQRI Smart Helmet

Le caratteristiche sono le seguenti:

- Display: 80 gradi FOV (Field of View) stereoscopico ed immersivo.
- Acquistabile: sì.
- Prezzo: non disponibile.
- Target di mercato: enterprise.

ODG R-7 ODG è una società statunitense specializzata in sistemi wearable e smart-glass. Gli ODG-R7 sono smart-glass con un target principalmente enterprise.

Figura 1.6: Immagine ODG R-7

Le caratteristiche sono le seguenti:

- Display: 30 gradi FOV (Field of View) stereoscopico ed immersivo.
- Acquistabile: su prenotazione.
- Prezzo: non disponibile.
- Target di mercato: enterprise.

Epson Moverio BT-200 Descriverò questo occhiale più in dettaglio in quanto tutta la fase di studio, prototipazione e testing di questa tesi è avvenuta su questo dispositivo. Il dispositivo Moverio BT-200 è un occhiale intelligente prodotto dall'azienda giapponese Epson [4]. Esso è dotato di numerose funzionalità tecnologiche per sfruttare appieno le app di Realtà Aumentata (Augmented Reality, AR), oltre a migliorare l'approccio con alcune attività che rientrano sia nell'ambito professionale sia nel tempo libero. Questi smart-glass sono costituiti da lenti trasparenti con un visore integrato su ogni lente, che agiscono in maniera indipendente dallo smartphone e si affidano ad un

controller esterno collegato con un cavo. All'interno di questa unità di controllo tascabile con trackpad touch e una serie di pulstanti essenziali (Power, Volume, Home) vi è un hardware con processore dual core da 1,2 GHz, 1 GB di RAM e 8 GB di capacità espandibile con una scheda microSD sino a 32 GB. Le immagini vengono cosi visualizzate in alta definizione e di grande formato fino a 320 pollici con risoluzione qHD nell'intero campo visivo.
La fotocamera, il giroscopio, il GPS e gli altri sensori integrati consentono al software di percepire tutti i movimenti e l'ambiente circostante. In particolare fa uso di un set di sensori a 9 assi (accelerometro, giroscopio, compass), presenti sia sul controller che sugli occhiali (sugli occhiali sono posizionati in alto a destra). Moverio BT 200 è basato su il sistema operativo Android 4.0.4 e dunque gli sviluppatori possono utilizzare il kit SDK standard di Android per creare app innovative o adattare quelle esistenti.

Figura 1.7: Immagine Moverio BT-200

Le caratteristiche sono le seguenti:

- Display: 23 gradi FOV stereoscopico ed immersivo.
- Acquistabile: sì.
- Prezzo: 800 euro circa.
- Target di mercato: enterprise e consumer.

1.3 Human Computer Interaction Hands-Free

Negli ultimi anni molte attività di ricerca hanno focalizzato il loro interesse su progetti che mirano a produrre sistemi universalmente accessibili che possono essere utilizzati da tutti, indipendentemente dalle loro abilità fisiche o cognitive. Le interazione tra uomo-computer convenzionali richiedono l'uso delle mani da parte dell'utente per muovere il mouse e per premere dei tasti sulla tastiera.
Difatti, quando si usa, ad esempio, un qualsiasi tipo di mouse, l'utente deve essere in grado di impugnarlo, muoverlo in una specifica direzione e riuscire a premere un pulsante per poter svolgere determinate azioni. [6] Molte persone con handicap fisici si trovano impossibilitati, proprio per questi motivi, a non poter utilizzare un dispositivo elettronico, mentre altre persone che usano un qualsiasi sistema con un convenzionale HCI(Human Computer Interaction), in un contesto dove devono utilizzare le loro mani per altre mansioni, sono impossibilitate nel poter lavorare ed utilizzare un qualsiasi sistema computazionale nello stesso momento.
Proprio per questi motivi sono stati ideati nuovi processi che permettono l'interazione tra uomo-computer senza la necessità di utilizzare obbligatoriamente e in modo costante le mani. Quest'ultimi si configurano perfettamente per essere utilizzati insieme ad un dispositivo di Mobile Augmented Reality (MAR) [7]. Si pensi ad un dottore che sta effettuando un'operazione, quest'ultimo potrà indossare uno smart-glass sul quale potrà leggere informazioni di necessità, e sarà in grado di controllare il dispositivo wearable senza l'utilizzo delle mani, cosi che l'utilizzo del sistema elettronico non influirà con il normale conseguimento dell'operazione. Esempi come questo possono essere trovati in moltissimi altri casi lavorativi.
Di seguito descriverò i sistemi HCI hands-free più utilizzati e che stanno riscuotendo una certa notorietà negli ultimi anni.

1.3.1 Speech Recognition

Lo Speech Recognition, in italiano riconoscimento vocale, è il processo attraverso il quale un linguaggio orale umano viene riconosciuto e successivamente elaborato attraverso un apposito sistema di riconoscimento vocale.
Il primo sistema di riconoscimento vocale è stato ideato nel 1964 dall'IBM, azienda statunitense con un importante notorietà nel mondo dell'informatica. Questo dispositivo era in grado di riconoscere e interpretare singole cifre parlate. Attualmente il riconoscimento vocale viene utilizzato da quasi tutti i sistemi informativi, specialmente smart-phone, per l'elaborazione del linguaggio naturale.

Questo tipo di sistema HCI si configura perfettamente per l'invio di input ad uno smart-glass senza l'utilizzo delle mani. Esso infatti comporta miglioramenti significativi nell'efficienza e nella velocità di inviare comandi di controllo al proprio device, basti pensare che il primo modello dei Google Glass era basato quasi esclusivamente su input vocali.

1.3.2 Gesture Recognition

Il processo di Gesture Recognition, in italiano riconoscimento di gesti, si basa sul fatto di riconoscere ed inviare input al proprio device attraverso un dispositivo di supporto, wearable tipicamente, in grado di riconoscere determinati gesti eseguiti dall'utente stesso.
Ad esempio, sfruttando in dispositivo che identifica i cambiamenti di tensione e rilassamento della pelle, grazie ad un sensore piezoelettrico posizionato al di sotto del mento, è stato possibile inviare precisi input a seconda della compressione e distensione dei muscoli presenti al di sotto della pelle dell'utilizzatore. Un dispositivo che ho avuto modo di studiare e utilizzare all'interno di questo elaborato è il dispositivo Myo gesture control armband.

Myo gesture control armband Il dispositivo Myo armband, prodotto dall'azienda Thalmic Lab, si tratta di un bracciale da indossare sull'avambraccio [5]. Esso è composto da 8 blocchi rettangolari, legati tra loro con una gomma elastica da indossare direttamente a contatto con la pelle. Questo perché i sensori EMG (acronimo di elettromiografia) integrati sono in grado di leggere l'attività elettrica dei muscoli e rilevare i gesti della mano combinati con altri sensori che invece registrano i movimenti del braccio, tramite questo dispositivo è infatti possibile collegarsi attraverso Bluetooth ad altri dispositivo come computer e smartphone ed inviare segnali di input con un ritardo appena percettibile. Il dispositivo per captare i movimenti del braccio utilizza un IMU (accelerometro, magnetometro e giroscopio) a 9 assi.

Figura 1.8: Immagine Myo gesture control armband

Il Myo di base è in grado di riconoscere, di default, cinque gesture della mano:

- Wave In: Chiusura della mano verso l'interno.
- Wave Out: Apertura della mano verso l'esterno.
- Double Tap: Doppio tap del pollice e dell'anulare.
- Fist : Chiusura della mano a pugno.
- Finger spread: Distensione delle dita della mano.

1.3.3 Eye-Tracking

L'eye tracking, in lingua italiana oculometria, è il processo di misurazione del punto di fissazione oculare o del moto di un occhio rispetto alla testa. Si tratta di un processo molto innovativo e ancora non trova molte applicazioni, oggigiorno, nel mondo consumer. Grazie all'oculometria, oltre a far scorrere una pagina semplicemente con gli occhi, è anche possibile studiare e capire come una persona interagisce con un dispositivo digitale, come si comporta alla visualizzazione di un messaggio oppure cosa cattura il suo interesse visivo all'apertura, per esempio, di una nuova applicazione. Con questi studi mirati sarà possibile migliorare l'user experience del device stesso.

1.3.4 Brain–computer interface

Il Brain-computer interface, letteralmente interfaccia cervello-computer, più comunemente nota come interfaccia neurale, è un mezzo di comunicazione diretto tra un cervello e un computer. Con questo tipo di processo è possibile captare segnali derivanti dall'attività cerebrale, come ad esempio il segnale elettroencefalografico, ed inviare determinati input al device a cui si è collegati. L'acquisizione e l'interpretazione di segnali elettroencefalografici è stata utilizzata con successo per comandare il movimento di una sedia a rotelle su percorsi predefiniti, o la sintesi vocale di un set definito di parole [9] .

Capitolo 2

Mixed Reality

Questo progetto da me svolto ha come scopo quello di studiare un dispositivo wearable, in particolare gli smart-glass, interagire con esso attraverso un processo HCI hands-free, ma soprattutto ha come obiettivo applicativo quello di analizzare, progettare e sviluppare un framework che permetta di comporre una sorta di lavagna virtuale sulla quale è possibile apporre oggetti sovrapposti alla realtà che l'utente percepisce; tutto ciò deve avvenire utilizzando le tecnologie precedentemente esposte. Sino ad ora, infatti, mi sono limitato a descrivere quali sono le tecnologie presenti attualmente sia in fase di ricerca che nel mercato stesso. In questo capitolo esporrò tutto ciò che ho avuto modo di apprendere, e successivamente di applicare, per quanto concerne la realtà aumentata, la realtà virtuale e la mixed reality, all'interno del mio caso di studio.

2.1 Virtuality Continuum

Per comprendere appieno cosa si intende per Virtuality Continuum, concetto proposto da Milgram e Kishino nel 1994 [8], è opportuno esporre i concetti di realtà aumentata (Augmented Reality, AR) e di realtà virtuale (Augmented Virtuality, VR).

- Virtual Environment: in lingua italiana ambiente virtuale. Con questo termine si vuole esprimere tutte le applicazioni software in cui tutto ciò che si vede e si percepisce non è reale. Qualsiasi gioco di computer, da Super Mario a Modern Warfare, può essere inteso come un ambiente virtuale. Questi tipi di applicazioni ti fanno percepire un mondo che non ha nessun collegamento con le forme fisiche reali.
- Augmented Virtuality: in lingua italiana realtà virtuale. Con questo termine è possibile esprimere tutti quei concetti e forme del mondo reale

"aumentate" ed utilizzate all'interno del mondo virtuale. Un semplice esempio può essere l'utilizzo di specifici sensori, come il giroscopio o l'accellerometro, per guidare un'automobile all'interno di un gioco di corse automobilistiche installato sul proprio smartphone. Negli ultimi anni sono stati prodotti diversi headset, come ad esempio l'Oculus Rift, che applicano la realtà virtuale in un contesto entrateinment. Questo particolare headset indossabile permette all'utilizzatore di vedere un mondo totalmente virtuale, che è possibile esplorare ruotando la proprio testa. Ciò è reso possibile attraverso l'utilizzo di particolari sensori, quali accellerometro, giroscopio, e magnetometro. Attraverso questi sensori infatti è possibile registrare la posizione e i movimenti, come la rotazione e l'inclinazione della testa, e applicare tali componenti fisiche al software stesso presente all'interno del device.

- Augmented Reality: in lingua italiana realtà aumentata. Con questo termine è possibile definire il processo attraverso la quale oggetti di tipo virtuale, quindi immagini, testo video, vengono rappresentati all'interno del mondo reale. Di seguito quest'ultimo concetto verrà trattato in maniera più approfondita.

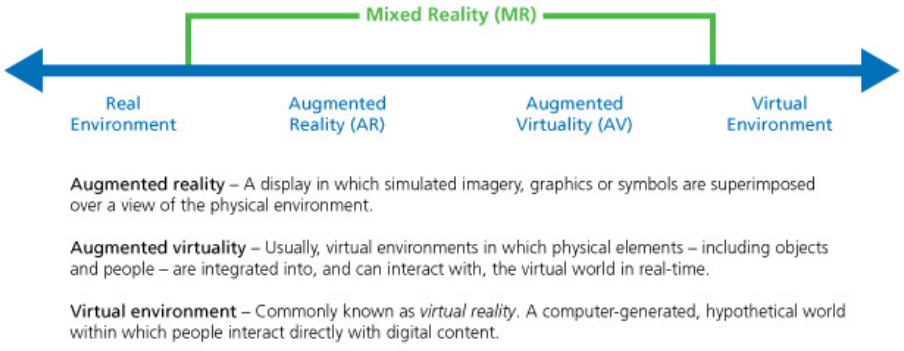

Figura 2.1: Immagine Virtuality continuum (Milgram and Kishino, 1994)

L'immagine soprastante mostra il concetto di Virtuality Continuum. Quest'ultimo può essere schematizzato visivamente con una linea (Figura 2.1), i cui estremi rappresentano rispettivamente un ambiente completamente reale e un ambiente completamente virtuale, quindi artificiale. I livelli intermedi di fusione tra gli ambienti reali e virtuali si trovano tra i due estremi del continuum, e caratterizzano, nel loro insieme, il concetto di mixed reality. Quest'ultimo è definibile, infatti, come un mix tra Augmented Reality e Virtual Reality. Drascic e Milgram nel 1996 danno la seguente definizione: "between the extremes

of real life and Virtual Reality lies the spectrum of Mixed Reality, in which views of the real world are combined in some proportion with views of a virtual environment" [10]. All'interno di queste considerazioni trovano definizione i concetti di augmented reality e augmented virtuality: come precedentemente espresso il primo rappresenta un ambiente reale a cui sono aggiunte informazione del tipo virtuale, mentre il secondo rappresenta un ambiente virtuale in cui sono presenti informazioni reali. Mentre il Virtual Environment e l'Augmented Virtuality sono caratterizzati dal fatto che l'utente visualizza un ambiente del tutto artificiale che può esplorare interattivamente, l'Augmente Reality invece ha lo scopo di supportare l'utente durante lo svolgimento di un'attività, arricchendo l'ambiente reale con informazioni utili e pertinenti al contesto in cui si trova. In definitiva è possibile affermare che l'Augmented Reality combina oggetti reali e virtuali in un ambiente reale, questo tipo di ambiente è interattivo e sia la visualizzazione che l'interazione con l'ambiente, da parte dell'utente, avvengono in tempo reale. Il processo attraverso la quale le informazioni reali e virtuali sono correlate tra loro viene denominato tracciamento e registrazione.

2.2 Tracciamento e registrazione

2.2.1 Registrazione

Una delle maggiori difficoltà che si riscontrano quando si lavora con sistemi di realtà aumentata consistono nell'allineamento e nella sincronizzazione tra il mondo fisico e il mondo virtuale.

Il processo di registrazione del sistema consiste infatti, una volta valutato l'orientamento e la posizione dell'utente, nell'associare e allineare gli oggetti virtuali all'interno del mondo reale percepito dal fruitore del sistema. Questo processo può essere sia di tipo spaziale che temporale. Per spaziale si intende la registrazione coerente dell'oggetto virtuale rispetto al mondo fisico. Se, ad esempio, viene posizionato un oggetto alla destra dell'utilizzatore e quest'ultimo fa un giro su se stesso ritornando alla posizione iniziale, l'oggetto deve rimanere, per la registrazione spaziale, alla sua destra. La registrazione temporale si riferisce, invece, ai cambiamenti che devono essere applicati al mondo virtuale in seguito al cambiamento del punto di vista dell'utente. Nel momento in cui l'utilizzatore cambia posizione, l'elemento virtuale dovrà essere visualizzato dal nuovo punto di vista. All'interno di questo progetto sono stati utilizzati processi e algoritmi che garantiscono solo una registrazione spaziale e non temporale.

2.2.2 Tracciamento basato sui sensori

Tutti i moderni smartphone, tablet e dispositivi wearable presentano al loro interno una vasta gamma di sensori che possono essere utilizzati per tracciare la posizione e l'orientamento dell'utente cosi da effettuare una corretta registrazione, sopra descritta. I sensori utili al fine dell'applicazione della realtà aumentata sono i seguenti [11] :

Sensori hardware

- Fotocamera: una fotocamera installata in un device può essere utilizzata per catturare e tracciare la realtà in maniera drift-free oppure più semplicemente per creare applicazioni in un contesto di video see-through display.
- Magnetometro: il magnetometro è lo strumento atto a misurare il campo magnetico. La misura delle componenti del campo lungo tre direzioni indipendenti permette di definire unicamente il vettore del campo magnetico nel punto in cui si effettua la misurazione. Grazie a questo sensore è possibile percepire sia il campo magnetico rispetto alla terra, ma anche quello rispetto ai device circostanti.
- Giroscopio: il giroscopio è lo strumento che serve per misurare la velocità angolare del dispositivo. Esso può essere utilizzato per osservare il cambiamento della rotazione su tre componenti del device (yaw, pitch e roll).
- Accellerometro: l'accellerometro è lo strumento che serve a misurare la risultate delle forze applicate al device, tra cui l'accellerazione e la gravità.
- GPS: il GPS è il sensore utilizzato per ricevere la posizione assoluta del device rispetto alle coordinate della terra (longitudine, latitudine, altitudine e il brearing).

Sensori software Oltre i sensori hardware sopra descritti, esistono dei sensori, definiti virtuali, che vengono derivati grazie alla fusione dei sensori hardware.

- Gravity: questo sensore viene ricavato dalla fusione del giroscopio e dell'accellerometro. Viene utilizzato per ottenere la direzione dalla quale le forze sono applicate sul device. Viene spesso utilizzato prendendo i dati direttamente dal giroscopio, e i problemi di drift causati da quest'ultimo, vengono corretti grazie ai dati catturati dall'accellerometro.

- Nord: il magnetometro è in grado di misurare il campo magnetico applicato al device rispetto della terra, ma purtroppo il nord magnetico determinato da questo sensore non è in tutte le parti del mondo il vero nord, questo fenomeno viene chiamato magnetic declination. In Nuova Zelanda per esempio il nord si trova tra i 18 e i 25 gradi di declinazione.

- Orientamento: la vecolità angolare catturata attraverso il giroscopio può essere convertita in un orientamento. Generalmente questo tipo di dato genera drift e non è del tutto preciso.

- Orientamento assoluto: è uno dei dati più rilevanti all'interno della sensor fusion. Per ottenerlo è necessario fondere insieme i dati provenienti dal giroscopio, accellerometro e magnetometro dove dal giroscopio si ottiene l'orientamento che viene poi corretto dai dati catturati dall'accellerometro e il magnetometro. All'interno di Android questo tipo di sensore viene chiamato Rotation Vector. Questo tipo di dato viene calcolato attraverso algoritmi di low-pass e high-pass filter oppure attraverso il Kalman filter. Generalmente l'accellerometro, il giroscopio e il magnetometro non sono mai utilizzati singolarmente in quanto causano problemi di drift, deviazione e noise.

Altri sensori Sono presenti all'interno dei più recenti dispositivi anche altri sensori, che non sono sempre utilizzati all'interno di un applicazione di realtà aumentata, ma che possono risultare importanti in contesti particolari.

- Wi-Fi: viene utilizzato nei grandi centri abitati dove il device è circondato da molte reti wi-fi. Google Location API utilizzano quest'approccio per determinare la posizione del device in maniera più precisa prendendo i dati sia dal posizionamento Wi-Fi che dal posizionamento GPS.

- Sensore di luminosità: utilizzato per percepire la quantità di luce che colpisce il device.

- Sensore di prossimità: utilizza il sensore di luminosità per capire se l'utente sta tenendo il cellulare in prossimità del suo orecchio oppure no.

- Microfono: sensore che viene utilizzato per captare i suoni provenienti da ciò che circonda il device. Fondamentale per applicare un HCI di tipo Speech-Recognition.

- Termometro

- Barometro

- Sensore di umidità

Rappresentazione Due diversi sensori, che forniscono in output una rotazione a tra dimensioni, possono essere fusi insieme nel modo corretto se viene scelto il giusto tipo di rappresentazione. E' infatti possibile rappresentare una rotazione di un corpo rigido in tre dimensioni in diversi modi.

- Angoli di Eulero: questo tipo di rappresentazione si compone di tre angoli di rotazione: yaw, pitch e roll. Gli Euler-Angles hanno il beneficio di essere semplici da comprendere e utilizzare, ma soffrono di diversi problemi, come ad esempio il gimbal lock.
- Rotation Vector: tramite il rotation vector è possibile rappresentare una rotazione arbitraria lungo uno specifico asse con tre componenti x,y,z e una rotazione di angolo alfa lungo l'asse.
- Matrice di rotazione: si tratta di una matrice 3x3 con cui è possibile rappresentare direttamente una rotazione. E' molto usata in computer graphics perché può essere combinata con altre trasformazioni in modo efficiente.
- Quaternioni: i quaternioni sono strettamente correlati con il rotation vector in quanto, se ci viene fornita una rotazione intorno ad un determinato asse, di angolo alfa, e le componenti x,y e z della rotazione, i quaternioni rappresentano quest'ultimo come q = cos(alfa/2) + i (x * sin(alfa/2)) + j (y * sin(alfa/2)) + k (z * sin(alfa/2)). Questo tipo di rappresentazione è sicuramente più complessa rispetto alle altre, ma offre un modo matematico molto elegante per interpolare due quaternioni, chiamato Spherical Linar Interpolation (SLERP).

Sensor Fusion La fusione dei dati provenienti dai sensori avviene attraverso l'utilizzo di filtri. Essi possono essere definiti come algoritmi che realizzano delle funzioni di trasformazione ed elaborazione di segnali ricevuti come input. Due importanti filtri, utilizzati nella realtà aumentata, sono il filtro complementare e il filtro di Kalman. Il primo viene utilizzato per processare i dati provenienti dal giroscopio, dall'accelleromentro e dal compass. Il giroscopio è in grado di fornire solamente una misura della velocità di rotazione, attraverso il filtro complementare è possibile aggiungere a tale componente, i dati provenienti dall'accellerometro e dal compass ed ottenere un orientamento assoluto drift-free. Il Kalman filter, rispetto al filtro complementare, è una soluzione più sofisticata, ma nettamente più precisa. Tramite questo filtro è infatti possibile fondere insieme sia i dati provenienti dall'accelleromentro, giroscopio e compass, con i dati provenienti dal GPS.

2.2.3 Tracciamento visuale

Oltre ad un tracciamento basato sui sensori, è possibile possibile effettuare tale processo attraverso i fotogrammi provenienti dalla fotocamera installata in un device. Questo tipo di processo si tratta di una elaborazione ad anelli chiuso: il risultato del tracciamento è utilizzato per correggere dinamicamente l'errore introdotto. Un esempio è il metodo model-based dove le corrispondenze tra scena reale e modello di un oggetto virtuale vengono elaborate usando linee e bordi, sia dell'oggetto virtuale che quello della scena reale, come feature di allineamento. Un altro metodo utilizzato è quello che si basa su appositi marker, immagini stampate su carta con geometrie e proprietà note a priori (tipo immagini QR o BCH), sulla quale vengono elaborate le corrispondenze per posizionare e allineare, sopra di esso, l'oggetto virtuale.

2.2.4 Tracciamento ibrido

Sia le tipologie sensor-base che visual-based presentano dei vantaggi e svantaggi. Combinando opportunamente questi due processi è possibile ottenere un tracciamento, chiamato ibrido, sicuramente più preciso ed accurato.

2.3 Sistemi di realtà aumentata

I sistemi che utilizzano questo tipo di tecnologie consentono agli utilizzatori di vivere esperienze che altrimenti sarebbero troppo costose o impossibili da sperimentare in prima persona. In particolare, un sistema di mobile augmented reality deve permettere l'immersione dell'utente nell'ambiente applicativo, la navigazione e l'interazione del fruitore con il sistema stesso. L'utilizzatore infatti deve avere la sensazione di essere immerso, di far parte, della realtà virtuale che viene rappresentata. Per ottenere questo tipo di effetto è importante che l'utente si senta partecipe dell'ambiente che viene visualizzato, deve esserci quindi la possibilità di esplorare elementi e strutture virtuali da diversi punti di vista e la facoltà di poter interagire con quest'ultimi. Al giorno d'oggi, per ottenere questi tipi di effetti, vengono utilizzati due diversi tipi di approcci che dipendono dalla tipologia del display che compone il sistema informativo. La prima tipologia di device che può essere utilizzata è composta da un optican see-through display. Come già descritto nel primo capitolo, il dispositivo consisterà in uno smart-glass con un display immersivo. Con questo tipo di approccio è possibile arricchire la normale visione dell'utente del mondo reale con scritte testuali e/o immagini virtuali.

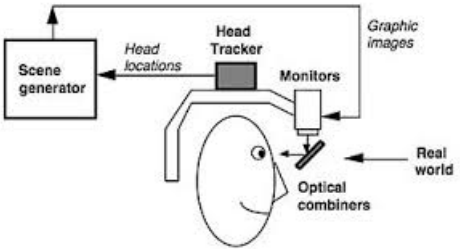

Figura 2.2: Immagine Optical See-Through Display

La seconda tipologia di display utilizzata per applicazioni di realtà aumentata è definita video see-through display. A differenza dell'optical see-through display, che utilizza un particolare vetro semitrasparente e semiriflettente per permettere all'utilizzatore di visualizzare la realtà, questa tipologia non permette la visualizzazione diretta della realtà da parte dell'utilizzatore, ma l'immagine dell'ambiente reale viene catturata attraverso una o più fotocamere, che ricostruisco la scena reale e sopra di essa vengono applicati gli oggetti virtuali, allineati attraverso processi di tracking e registration con la realtà stessa. L'immagine del mondo reale è di conseguenza mescolata elettronicamente con l'immagine generata dal computer ed esposta, tipicamente, su uno schermo a cristalli liquidi. Questo tipologia di realtà aumentata può essere implementata sia utilizzando dei particolari head-set, sia attraverso l'utilizzo di smart-phone o tablet.

Un video see-through display ha alcuni vantaggi, ma anche gli svantaggi sono diversi. La luminosità relativa delle due scene può essere gestita opportunamente, la digitalizzazione della scena reale consente un miglior processo di tracciamento e registrazione da parte del sistema. Inoltre, rispetto all'optical see-through display la miscelazione tra i contenuti virtuali e reali avviene in maniera più semplice. D'altro canto presenta anche diversi svantaggi, la scena dell'ambiente reale ha sicuramente una risoluzione limitata, così come il campo visivo dell'utente.

Con l'utilizzo degli optical see-through display la risoluzione del mondo reale rimane intatta, inoltre se l'alimentazione dovesse interrompersi improvvisamente l'utente continuerà comunque a vedere la scena reale. D'altra parte questo tipo di diplay presenta anche alcuni svantaggi. Quelli che personalmente ho avuto modo di constatare sono che la gestione delle occlusione tra oggetti reali e virtuali, a volte, può risultare problematica. Inoltre la scena virtuale può presentare un livello ridotto di luminosità, cosi che l'utilizzo dell'occhiale in ambiente outdoor può diventare problematico. Infine, tipicamente, il campo di vista della scena virtuale, quindi il suo field of view, è minore rispetto a quello della scena reale.

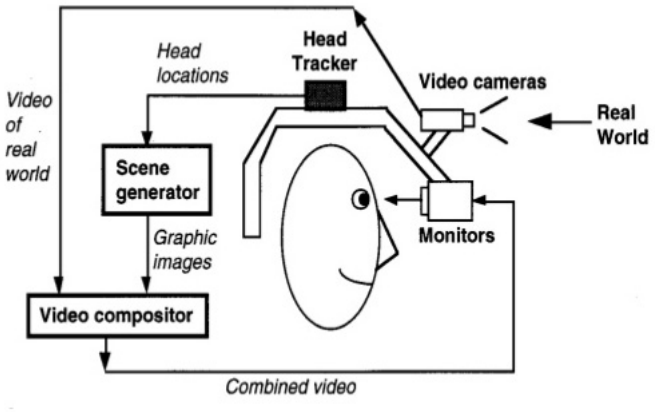

Figura 2.3: Immagine Video See-Through Display

Capitolo 3

Ambiti applicativi

L'innovazione tecnologica portata dalla realizzazione di sistemi hands-free in un contesto di realtà aumentata, attraverso l'uso di smart-glass, è applicabile in molteplici ambiti applicativi che tratterò in questo capitolo. E' importante sottolineare che tali sistemi, per essere applicati in diversi contesti, dovranno avere delle ottime caratteristiche sia dal punto di vista degli algoritmi utilizzati per generare la realtà aumentata, sia un processo di interazione uomo-macchina consono e comodo per il fruitore del sistema. Il sistema deve essere in grado di mostrare informazioni pertinenti a seconda dell'ambito applicativo in cui ci si trova e queste informazioni non devono essere troppo invadenti.

3.1 La realtà aumentata oggigiorno

Di seguito elencherò i principali contesti applicativi in cui dove è possibile utilizzare gli smart-glass, basato su input hands-free, in un contesto di realtà aumentata.

3.1.1 Intrattenimento

Al girono d'oggi esistono diverse applicazioni entrateinment di realtà aumentata presenti per device come tablet e smartphone. In questo ambito, sin ad ora, la maggior parte delle applicazione si basano su sistemi di tipo video-through display. Un'applicazione che ha riscosso un discreto successo prende il nome di Augmented. Grazie a questa applicazione è possibile catturare il mondo reale attraverso l'uso della fotocamera e aggiungere sopra di esso modelli 3D di oggetti virtuali. Questi oggetti possono essere ingranditi, rimpiccioliti e spostati nella posizione dove l'utente preferisce. Muovendo il device in qualsiasi direzione anche le immagini catturate si sposteranno di conseguenza e l'oggetto posto al di sopra di questa realtà rimane fermo, garantendo un

registrazione di tipo spaziale. Con l'utilizzo di smart-glass che garantiscono un optical see-through display, questo tipo di esperienza risulterà sicuramente più immersiva e interattiva, soprattutto grazie all'utilizzo del riconoscimento di gesti e comandi vocali.

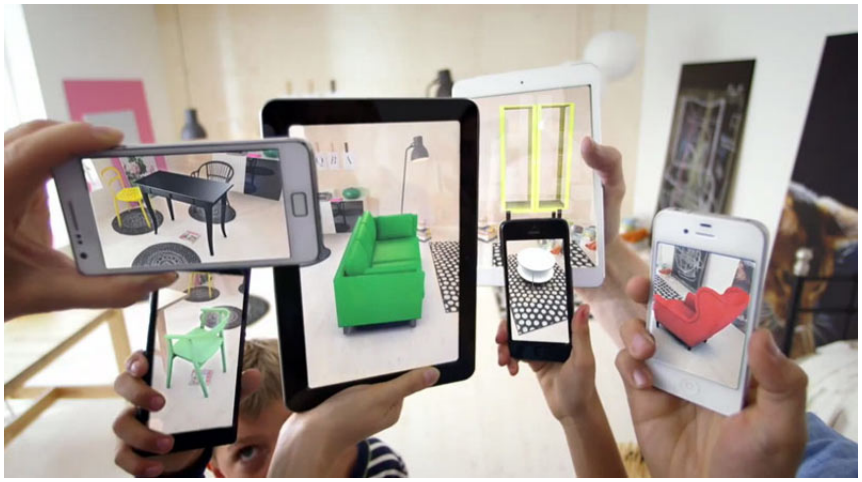

Figura 3.1: Immagine applicazione Augmented

3.1.2 Turismo

Molti musei e siti archeologici hanno proposto in passato soluzioni di realtà aumentata per mostrare ai turisti informazioni aggiuntive su ciò che vedevano all'interno di queste strutture. Purtroppo quando ciò è avvenuto ancora non era presente un hardware in grado di sfruttare appieno tali concetti. Anche in questo caso, con l'utilizzo di smart-glass, si potrebbe offrire una soluzione utile e gradevole per un visitatore di uno di questi centri turistici. Difatti i benefici che possono essere apportati sono molteplici. L'utente non dovrà più perdere tempo nella ricerca di pannelli informativi o di guide cartacee, inoltre il visitatore sarà libero di visualizzare informazioni aggiuntive solo per i contenuti di cui ha interesse, tralasciando quelli, da lui considerati, meno interessanti.

3.1.3 Automotive

In ambito automotive è di fondamentale importanza che l'autista di un qualsiasi mezzo di trasporto non si distragga mai dalla guida dello stesso. In un contesto di questo tipo un sistema basato su smart-glass con input hands-free può costituire una valida soluzione per guidare e nello stesso tempo utilizzare un dispositivo che mostri informazioni rilevanti. Tali informazioni potrebbero

essere, ad esempio, indicazioni stradali, informazioni sul traffico, limiti di velocità. In questo contesto il miglior processo di human-computer interaction si ha con l'utilizzo del riconoscimento vocale. La casa automobilistica Mini, attraverso lo smart-glass ODG presentato nel Capitolo 1, ha recentemente proposto una soluzione di questo tipo.

3.1.4 Soccorso

Sistemi di realtà aumentata possono essere utilizzati anche in situazione di soccorso e di pubblica sicurezza. Un pilota di un veicolo aereo alla ricerca di un escursionista disperso in luoghi montuosi, ad esempio, può utilizzare questo tipo di tecnologie per avere una maggiore consapevolezza geografica, visualizzando a video le informazioni riguardanti i nomi delle strade forestali e possono cosi fornire, in modo tempestivo ed efficiente, tale informazioni ai soccorritori.

3.1.5 Medicina

Con l'utilizzo di queste tecnologie un medico che sta effettuando un'operazione chirurgica, può utilizzare visualizzare a video informazioni rilevanti sul paziente, come il battito cardiaco, immagini di radiografie o ecografie precedentemente effettuate, senza che esso distolga lo sguardo dal suo operato. Anche gli input inviati al device non devono distrarre il medico, dovranno quindi essere utilizzati opportuni processi di human-computer interacion hands-free. Anche in questo caso un processo di riconoscimento vocale risulta consono e utile. Recentemente è stata stipulata una partnership tra la Microsoft e la Western Reserve University, in Ohio. Grazie all'utilizzo degli occhiali Hololens sarà infatti possibile, per i laureandi in medicina di questa università, visualizzare e studiare in modo interattivo e pratico un corpo umano a grandezza naturale, attraverso l'uso di ologrammi, e successivamente andare a selezionare ed analizzare uno specifico strato di quest'ultimo. Sarà quindi possibile programmare simulazioni virtuali per approfondire le conoscenze apprese sui libri.

3.1.6 Enterprise

La realtà aumentata può facilitare la collaborazione di persone che operano all'interno dello stesso team di sviluppo. Una sistema simile ad una lavagna virtuale interattiva può essere utilizzato per avere uno spazio di progettazione comune. In tale contesto, questo tipo di tecnologie, aumenterebbero sicuramente l'efficienza della collaborazione tra il personale.

3.2 Framework e tools

Sono stati sviluppati diversi framework e tools che permettono, agli sviluppatori di sistemi di realtà aumentata, di focalizzare la loro attività sulle funzionalità di alto livello, mettendo a loro disposizione strumenti come API, SDK e librerie. Tra i framework che stanno riscontrando una certa notorietà vi sono Wikitude, Meataio, Layar, Vuforia e tanti altri. Tutti questi strumenti offrono esperienza significative nell'ambito della realtà aumentata, ma sono spesso limitati dalla necessità di utilizzare tecniche avanzate di riconoscimento delle immagine o markers per risultare efficaci. In particolare:

- Wikitude: è un tool di sviluppo atto a creare applicazioni di realtà aumentata. Nato in Austria nel 2008, inizialmente si prefiggefa l'obiettivo di essere un valido supporto ad applicazioni AR location-based. Dal 2012 ha iniziato ad offrire anche funzionalità per il riconoscimento di immagini e markes, con il lancio di Wikitude SDK. Esso sfrutta anche tecnologie per la geolocalizzazione GPS e sensori quali giroscopio, accellerometro e compass per permettere il calcolo della posizione degli oggetti da visualizzare sullo schermo di smart-phone e smart-glasses.

- Metaio: si tratta di una compagnia fondata nel 2003 a Monaco di Baviera. Attraverso il suo SDK è possibile sviluppare applicazioni AR in vari ambiti come ad esempio iOs, Android e Windows. Nel 2009 ha lanciato un browser di realtà aumentata, chiamato Junaio, ampiamente diffuso e utilizzato.

- Layer: compagnia olandese fondata nel 2009. Anch'esso si tratta di un browser di realtà aumentata molto diffuso ed utilizzato.

- Vuforia: si tratta di una piattaforma di sviluppo di applicazioni di mobile augmented reality, prodotta dall'azienda Qualcomm. L'SDK da loro offerto permette di utilizzare tecniche di computer vision per riconoscere, tracciare e registrare immagini e oggetti 3D in un contesto di realtà aumentata in real-time. Essa può essere utilizzata sia con l'utilizzo di marker, ma anche senza, utilizzando un loro algoritmo chiamato Occlusion Detection.

Capitolo 4

Caso di Studio

4.1 Caso di studio

L'idea di base che ha guidato questo lavoro è stata l'esplorazione sia in linea teorica che pratica di un possibile scenario di utilizzo di smart-glass, di tipo optical see-through display, come strumento per ottenere un'interfaccia in cui l'utente vede oggetti virtuali contenenti informazioni di vario tipo, generati da un applicazione che gira sull'occhiale stesso, in trasparenza, ossia all'interno della realtà che lo circonda.
In particolare il lavoro ha posto il focus sulla progettazione e lo sviluppo di un framework che permettesse all'utente inserire oggetti virtuali, come immagini o testo, all'interno di una lavagna virtuale di una dimensione prefissata. Una volta inizializzata la finestra e posti gli elementi al suo interno, l'indossatore potrà muovere la testa liberamente e di conseguenza modificare la finestra di visione rispetto all'ambiente circostante senza che gli elementi virtuali della realtà aumentata seguino il suo movimento. In questo modo si ottengono elementi virtuali che non appartengono al sistema di riferimento degli occhiali, che rimangono fermi nel punto dell'ambiente dove sono stati posizionati, quasi come se fossero veri oggetti fisici. Tale interfaccia permette quindi al fruitore del sistema una maggiore libertà di movimento, e la possibilità di avere il campo visivo più libero, mantenendo allo stesso tempo una rapida accessibilità alle informazioni portate da questi oggetti, raggiungibili semplicemente ruotando la testa fino a portare il campo visivo nella loro direzione.
Il framework permette anche di interagire sia con la finestra virtuale, che con gli oggetti stessi, attraverso una interazione human-computer hands-free. In particolare sarà possibile sbloccare e bloccare il movimento della lavagna virtuale generato dal movimento della testa oppure selezionare gli oggetti virtuali cliccando su di essi, tutto questo attraverso un dispositivo di riconoscimento di gesture, così da permettere un'interazione hands-free.

4.2 Considerazioni

Utilizzando un'applicazione che si basa sul nostro framework, quando ruotiamo la testa quello che ci aspettiamo è che alcuni oggetti escano dal nostro campo visivo, mentre altri vi entrino.
Il motivo per cui si è voluta esplorare un'interfaccia utente di questo genere è che essa si avvicina maggiormente all'esperienza naturale che l'utilizzatore ha con gli oggetti del mondo fisico nelle sue normali interazioni di tutti i giorni. Questo invece non avviene nelle normali applicazioni di default che possono essere eseguite su smart-glass, in cui il campo visivo è sempre e comunque riempito dalla schermata di tali applicazioni e dagli elementi in esso disegnati. Per ottenere questo è necessario progettare un sistema in cui il sistema di riferimento per gli oggetti virtuali non è più legato al dispositivo che l'utente indossa, ma all'utente stesso. L'interfaccia che si vuole realizzare deve avere le seguenti caratteristiche:

- Deve garantire un certo grado di immersività dell'utente all'interno del contesto applicativo che si viene a creare.
- Naturale, semplice ed immediata da utilizzare.
- Adattabile alle varie necessità.
- Hands-Free: Deve lasciare all'utente le mani libere per svolgere altre attività durante l'utilizzo del sistema, prevedendo altri metodi di interazione con esso che non comportino l'uso delle mani.

Un'interfaccia di questo tipo apre l'utilizzo degli smart-glass ad un numero molto grande di possibili applicazioni reali, sia in ambito lavorativo che nella vita di tutti i giorni. Essa permette infatti di aggiungere una componente aumentata alla realtà circostante senza che essa entri in modo prepotente nell'esperienza utente, che potrà a suo piacimento visualizzare o meno le informazioni aggiuntive semplicemente con una piccola rotazione della testa. L'utente potrà inoltre interagire con le componenti virtuali in maniera naturale, senza dover distrarre la propria attenzione dall'attività che sta svolgendo.E' facile constatare che un'applicazione di questo tipo si configura perfettamente per venire incontro alle necessità che si hanno in vari contesti di tipo enterprise, medico, soccorso, militare, turistico e istruttivo.

4.3 Caso applicativo

All'interno di questo progetto di tesi è stato sviluppato e studiato anche un particolare caso di applicazione del caso di studio precedentemente descrit-

to. In particolare si è pensato di utilizzare questo tipo di interfaccia utente, disponibile ed utilizzabile attraverso il framework realizzato, in un contesto dove più utenti indossano nello stesso momento uno smart-glass dove è presente un'applicazione basata sul nostro framework. Un utente esterno avrà a disposizione un'applicazione computer dalla quale sarà in grado di inviare determinati comandi a tutti gli occhiali connessi a tale applicazione; un esempio può essere l'invio di un testo da stampare in una determinata posizione della lavagna virtuale presente all'interno degli occhiali degli utilizzatori. Tutti gli utenti che indossano gli smart-glass saranno in grado di visualizzare, attraverso la rotazione della propria testa, gli oggetti che l'amministratore del sistema intende mostrare, e questi oggetti saranno visualizzati rispetto al sistema di riferimento dell'indossatore. Nel capitolo 6 sarà trattato nel dettaglio questo caso applicativo.

Capitolo 5

Il Framework

Il framework progettato e sviluppato per questo progetto di tesi è reperibile presso il seguente repository bitbucket pubblico:
https://bitbucket.org/djanno/wearableui

5.1 Analisi dei requisiti

In questa sezione tratterò la parte di analisi dei requisiti del framework, senza entrare nello specifico dei dettagli implementativi, esponendo il funzionamento dell'applicativo software realizzato ad alto livello.

5.1.1 Caratteristiche

Innanzitutto è opportuno definire le caratteristiche che questo progetto deve possedere. Essendo un framework, esso deve consistere in una architettura logica di supporto per la progettazione e la realizzazioni di applicazioni ineerenti al contesto descritto all'interno del capitolo 4. Un framework di questo tipo si prefigge come obiettivo quello di facilitare e velocizzare lo sviluppo di un'applicazione composta da una lavagna virtuale interattiva, offrendo allo sviluppatore che ne fa uso una serie di API utili ai fini del suo progetto. Lo scopo di un framework è infatti quelli di risparmiare allo sviluppatore la scrittura di codice già scritto in precedenza per compiti simili. Entrando più nello specifico, una delle parti più difficili nello sviluppo di un'applicazione di questo tipo si ha nel calcolare il punto di vista dell'utente in real-time, così da mostrare gli oggetti virtuali disposti in modo coerente rispetto alla realtà che l'utente percepisce. Deve esserci quindi la possibilità di inizializzare la lavagna virtuale, con una dimensione decisa dallo sviluppatore, sulla quale poter inserire gli oggetti non reali. L'utente finale, infine, deve avere la possibilità di

interagire con questi oggetti e con la lavagna stessa, attraverso un processo di human-computer interaction hands-free.

5.1.2 Concetti chiave

E' importante definire inizialmente alcuni concetti utilizzati per denominare alcune funzionalità e riferimenti presenti all'interno di questo progetto. Voglio sottolineare che tali concetti saranno trattati in seguito più nel dettaglio, ma vengono definiti in modo essenziale sin da subito, in quanto utili per esprimere in modo consono la parte di analisi del framework.

- Viewport: questo termine viene utilizzato per riferirsi alla finestra virtuale sulla quale vengono posti gli elementi come immagini o testo. Questa lavagna può essere navigata attraverso il movimento della testa. Essa ha due dimensioni e il suo centro è definito nel punto (0,0). Di default ha una dimensione tale che è possibile scorrerla orizzontalmente, a partire dal suo centro, di 60 gradi verso destra e 60 gradi verso sinistra. Verticalmente è possibile scorrerla di 45 gradi verso l'alto e il basso. Sono state scelte queste dimensioni in quanto sono le più comode e pratiche per il suo utilizzo, difatti un essere umano può ruotare la proprio testa, rimanendo fermo, e scorrere tutta la viewport, di una quantità di angolazione pari a quella settata di default.

- Cursor: corrisponde al puntatore utilizzato dal fruitore del sistema per selezionare o indicare gli elementi presenti all'interno della viewport.

- Field of View (FOV): con questo termine si vuole indicare ciò che un utente sta vedendo in un determinato momento, in particolare ci si riferisce alla porzione di realtà sulla quale sono applicati gli oggetti virtuali.

5.1.3 Funzionalità offerte

Uno sviluppatore che utilizza questo framework avrà a disposizione una serie di API che potrà utilizzare per sviluppare un'applicazione di realtà aumentata. Esso non si dovrà preoccupare di implementare un proprio algoritmo di realtà aumentata, attraverso l'uso dei sensori inerziali, ma avrà la possibilità di disegnare oggetti virtuali e di permettere all'utente finale di interagire con essi. Più in dettagli le funzionalità offerte sono le seguenti:

- Creazione della viewport con dimensione scelte dallo sviluppatore stesse.

- Possibilità di bloccare, quindi disabilitare, l'aggiornamento della viewport a seconda del movimento della testa dell'utente.

- Possibilità di sbloccare, quindi attivare, l'aggiornamento della viewport a seconda del movimento della testa dell'utente.

- Possibilità di disegnare un rettangolo con determinate dimensioni, posizionare il punto di disegno dove meglio crede, settare il colore, settare la trasparenza e infine stabilire se la forma geometrica deve essere di tipo fill (tutto colorato) oppure deve mostrare solo i bordi e l'interno trasparente.

- Possibilità di disegnare un cerchio con determinate dimensioni, posizionare il punto di disegno dove meglio crede, settare il colore, settare la trasparenza e infine stabilire se la forma geometrica deve essere di tipo fill (tutto colorato) oppure deve mostrare solo i bordi e l'interno trasparente.

- Possibilità di disegnare una linea con una determinate lunghezza, posizionare il punto di disegno dove meglio crede, settare il colore, settare la trasparenza

- Possibilità di disegnare un punto nel punto della viewport che preferisce, impostare il colore e il grado di trasparenza.

- Possibilità di disegnare un'immagine nel punto della viewport che preferisce, con delle dimensioni da lui scelte.

- Possibilità di fare uno snapshot della porzione della viewport attualmente visibile all'utente, quindi il FOV.

- Possibilità di rimuovere gli oggetti inseriti.

- Possibilità di modificare gli oggetti inseriti.

- Possibilità di interagire attraverso un click del cursor con gli oggetti inseriti.

- Possibilità di forzare lo scroll della viewport nel punto di FOV che preferisce

Diagramma casi d'uso Di seguito viene riportato il diagramma dei casi d'uso del sistema.

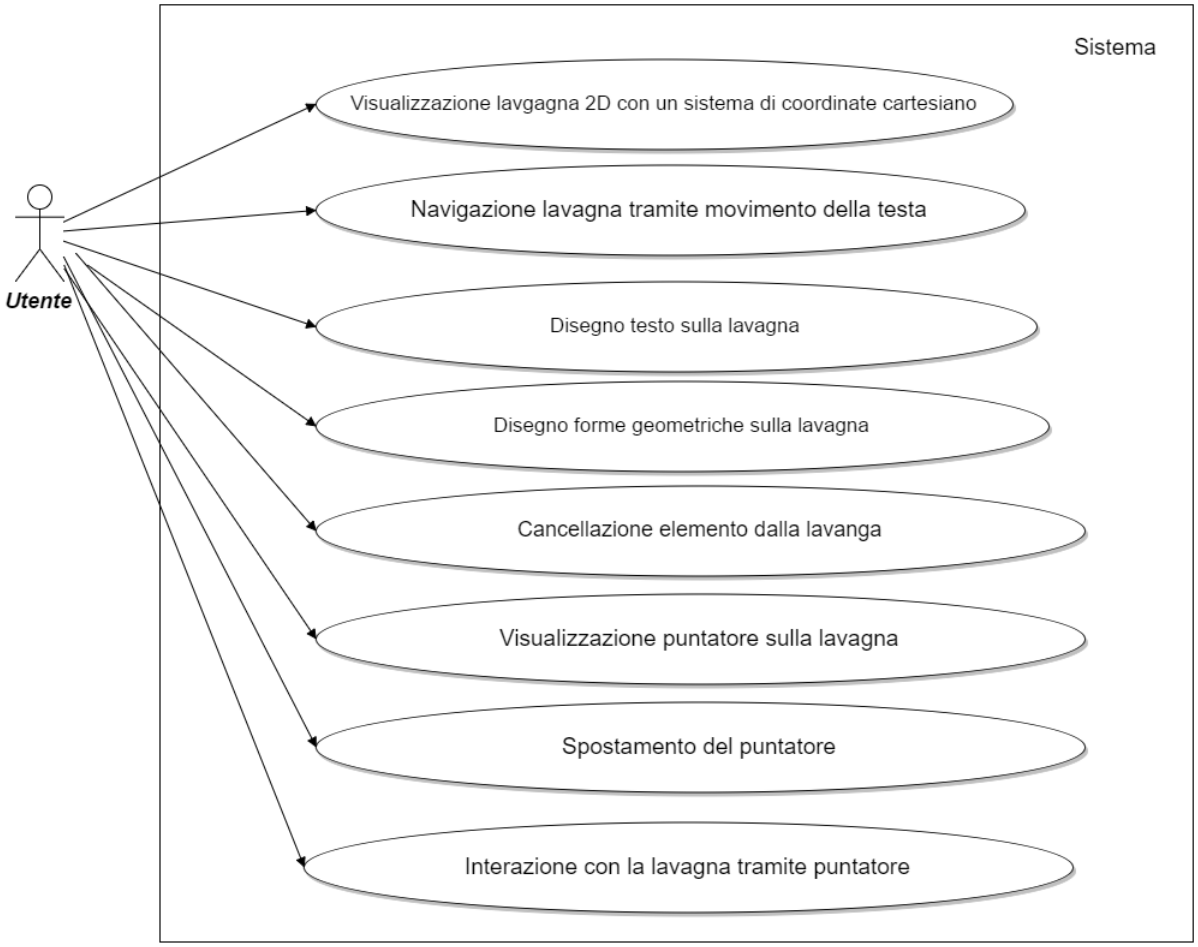

Figura 5.1: Diagramma dei casi d'uso del sistema

5.2 Architettura logica di supporto

In questa sezione tratterò l'architettura logica realizzata all'interno del framework, che dovrà essere utile allo sviluppatore che intende progettare e realizzare applicazioni basate su questo lavoro. In dettagli analizzerò i seguenti aspetti del progetto:

- Interfaccia del framework
- Rendering degli oggetti e interazione con essi
- Processo di comunicazione con un device di input esterno

5.2.1 Interfaccia del Framework

In questa sezione esporrò come uno sviluppatore che vuole usufruire del nostro framework può inizilizzarlo ed utilizzare le funzionalità offerte.

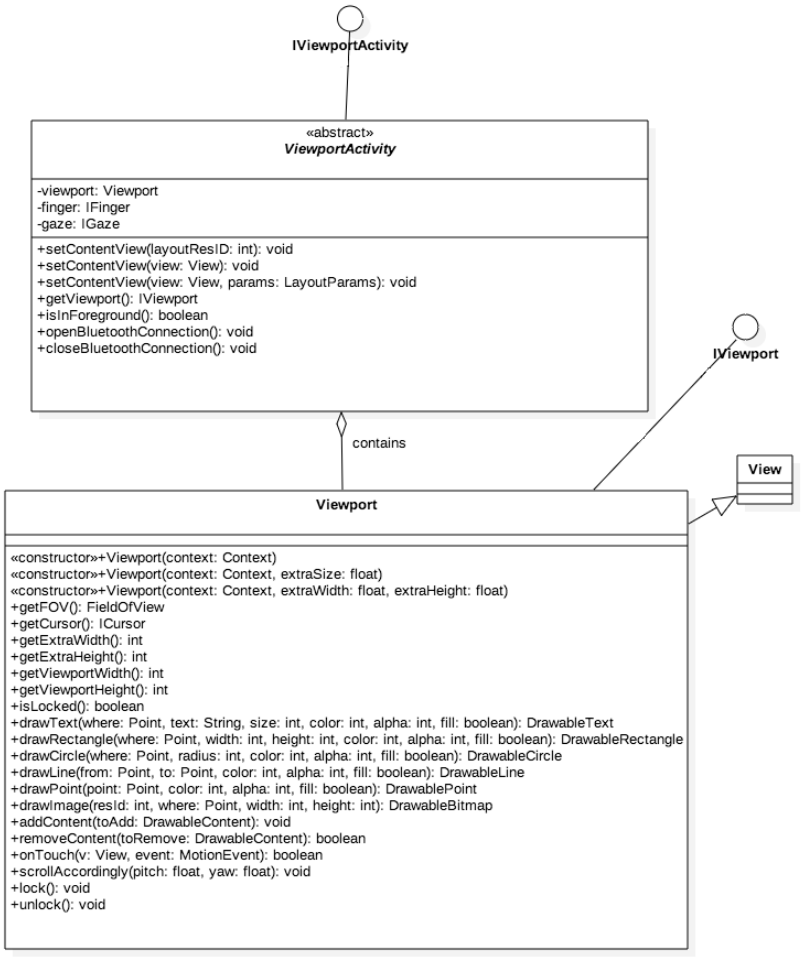

Figura 5.2: Diagramma UML ViewportActivity e Viewport

L'immagine soprastante mostra lo schema UML, in formato lollipop notation, della classe ViewportActivity che contiene al suo interno il riferimento alla classe che modella la Viewport. [1]

[1]Nel seguente UML, e in tutti gli UML che riporterò in questa sezione, verranno raffigurati solamente i metodi utili al fine di una descrizione esaustiva alla parte di design, e quindi non tutti i metodi presenti all'interno delle classi stesse, che, ad esempio, vengono utilizzati per scelte implementative e non di design e modellazione.

Viewport Innanzitutto è opportuno definire correttamente la classe che modella la viewport. Essa rappresenta una custom view (estende la classe View e implementa IViewport) che ha una larghezza e un altezza maggiore o uguale alla grandezza dello schermo del dispositivo. Essa corrisponde all'unica view che la ViewportActivity può contenere. Su di essa vengono posizionati gli oggetti virtuali che compongono la realtà aumentata che andremo a creare attraverso questo framework. Essa rappresenta una specie di lavagna 2D virtuale che può essere navigata attraverso il movimento della testa. La classe View rappresenta il blocco di componenti atte a creare la user interface. I metodi pubblici atti ad interagire con essa, offerti dalla sua interfaccia sono i seguenti:

- Costruttore:
  - Parametri in ingresso: nessuno.

  Questo costruttore non prende in ingresso nessun parametro e crea una viewport di dimensioni pari a quella delle dimensioni dello schermo del device utilizzato.

- Costruttore:
  - Parametri in ingresso: float extraSize

  Questo costruttore prende in ingresso un numero reale che esprime, in percentuale, quanto la viewport deve essere più grande rispetto alle dimensioni dello schermo del dispositivo utilizzato.

- Costruttore:
  - Parametri in ingresso: float extraWith, float extraHeight

  Questo costruttore prende in ingresso due numeri reali che servono per stabilire quanto la viewport deve essere più grande, in larghezza e altezza, rispetto alla dimensione dello schermo del device utilizzato.

- lock() : questo metodo deve essere utilizzato per bloccare l'aggiornamento e la navigazione della viewport di conseguenza al movimento della testa dell'utente.

- unlock() : questo metodo viene utilizzato per attivare l'aggiornamento e la navigazione della viewport di conseguenza al movimento della testa dell'utente.

- isLocked() : questo metodo restiusce un valore booleano a seconda se l'aggiornamento della viewport di conseguenza al movimento della testa dell'utente è bloccato oppure no.

- getCursor(): restituisce l'oggetto ICursor.
- getExtraWidth() : restituisce un valore intero che indica di quanto la viewport è più larga rispetto alle dimensioni dello screen del device utilizzato.
- getExtraHeight() : restituisce un valore intero che indica di quanto la viewport è più alta rispetto alle dimensioni dello screen del device utilizzato.
- getViewportWidth() : restituisce un valore intero che indica la larghezza della viewport.
- getViewportHeight() : restituisce un valore intero che indica l'altezza della viewport.
- getFov() : ritorna un oggetto di tipo FieldOfView che indica la porzione della viewport attualmente visibile all'utente.
- drawText:
  - Paramentri in ingresso:
    - Point where : punto della viewport sulla quale si vuole disegnare il testo. Corrisponde al punto in alto a destra della hit-box.
    - String text : testo da disegnare
    - int size : dimensioni del testo
    - int color : valore corrispondente al colore che si vuole applicare al testo
    - int alpha : valore di trasparenza
    - boolean fill : fill oppure stoke style
  - Paramentri in uscita: Oggetto di tipo DrawableText.
- drawRectangle:
  - Paramentri in ingresso:
    - Point where: punto della viewport sulla quale si vuole disegnare l'oggetto. Corrisponde al punto in alto a destra della hit-box.
    - int width: larghezza del rettangono
    - int height : altezza del rettangolo
    - int color: valore corrispondente al colore
    - int alfa: valore corrispondente alla trasparenza

        * boolean fill: fill or stroke style
    - Paramentri in uscita: Oggetto di tipo DrawableRectangle

- drawCircle:
    - Paramentri in ingresso:
        * Point where: punto della viewport sulla quale si vuole disegnare l'oggetto.
        * int radius: raggio del cerchio
        * int color: valore del colore
        * int alpha: valore della trasparenza da applicare
        * boolean fill : fill or stroke style
    - Paramentri in uscita: Oggetto di tipo DrawableCircle

- drawPoint: metodo per disegnare un punto simile ai metodi precedenti di disegno. Ritorna un oggetto di tipo DrawablePoint.

- drawLine: metodo per disegnare una linea simile ai metodi precedenti di disegno. Va impostato il punto di partenza e il punto di fine della linea. Ritorna un oggetto di tipo DrawableLine.

- drawImage: prende in ingresso un id corrispondente ad una immagine presente all'interno dell'applicativo. Ritorna un oggetto di tipo DrawableBitmap

- addContent: metodo per disegnare un qualsiasi degli oggetti di tipo Drawable

- removeContent: metodo per eliminare un qualsiasi oggetto di tipo Drawable

- scrollAccordingly:
    - Paramentri in ingresso: float pitch, float yaw

  Questo metodo deve essere utilizzato per portare il FOV percepito dall'utente ai valori degli angoli, in numeri reali, presi in ingresso. Questo metodo effettua il refresh della viewport e forza il suo ridisegnamento.

E' inoltre importante sottolineare che quando si disegna un qualsiasi oggetto

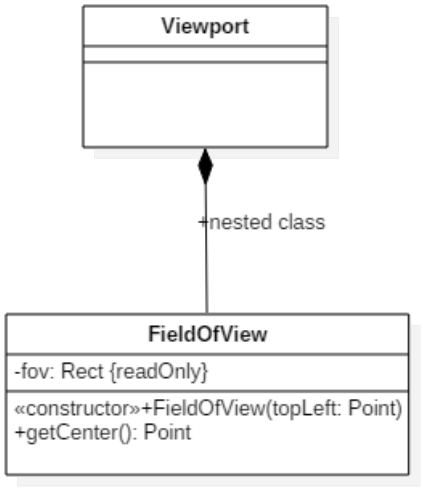

Figura 5.3: Diagramma UML FieldOfView

FieldOfView La classe Viewport presenta al suo interno una nested class di nome FieldOfView. Quest'ultima rappresenta la dimensione e la posizione della viewport stessa che è visibile, in un dato momento, all'utilizzatore del sistema. Il FieldOfView utilizza lo stesso sistema di coordinate della Viewport.

ViewportActivity La classe ViewportActivity è una classe astratta che deve essere estesa dallo uno sviluppatore che vuole utilizzare il nostro framework. Essa funge come il main controller per l'intero sistema realizzato. Lo sviluppatore che vuole utilizzare questo framework dovrà quindi estendere la classe ViewportActivity e creare un oggetto Viewport. Una volta fatto ciò può applicare l'oggetto di tipo Viewport alla sua applicazione richiamando il metodo setContetView(View Viewport). E' importante sottolineare che la classe ViewportActivity può prendere in ingresso, tramite il metodo setContentView, solo View di tipo Viewport. La classe ViewportActivity funge da controller del sistema, è infatti essa che si occupa di: attivare/disattivare le connessioni Bluetooth con un device di input; effettuare aggiornamento del rendering deli oggetti virtuali a seconda del movimento della testa dell'utente; effettuare l'aggiornamento del rendering del cursor in base alla sorgente di input; bloccare o sbloccare l'aggiornamento degli oggetti virtuali; ridisegnare la viewport; effettuare la calibrazione iniziale del dispositivo; Tutti questi aspetti verranno trattati in dettaglio più avanti.

5.2.2 Oggetti Virtuali

In questa sezione viene trattata la parte di design e modellazione del rendering degli oggetti virtuali e le funzionalità di interazione che essi offrono. Come già precedentemente descritto gli oggetti che si possono inserire all'interno della viewport sono i seguenti:

- Testo
- Immagini
- Rettangoli
- Cerchi
- Punti
- Linee

Oltre a questi oggetti sarà descritto:

- Cursor: mirino utilizzato dall'utente per selezionare e cliccare gli oggetti virtuali.

Rendering degli oggetti virtuali Di seguito viene mostrato il diagramma UML che raffigura le classi e le interfaccie utilizzate per il processo di rendering.

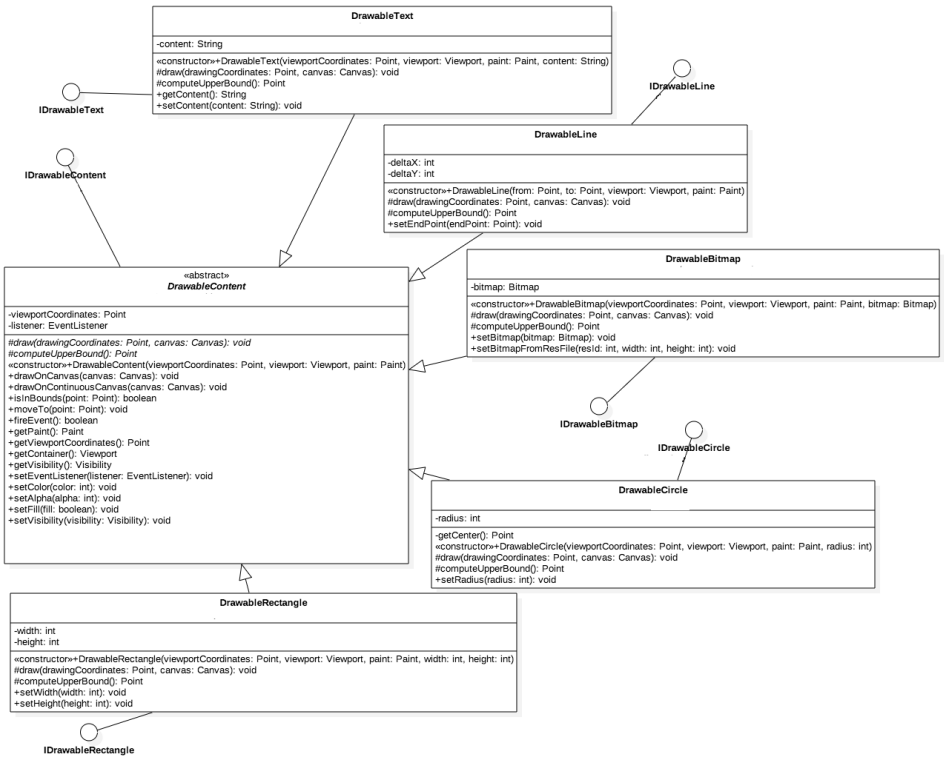

Figura 5.4: Diagramma UML DrawableContent

Come è possibile vedere dal diagramma soprastante ogni oggetto che deve essere disegnato all'interno della viewport estende la classe astratta DrawableContent e implementano la propria interfaccia. La classe astratta DrawableContent implementa la propria intefaccia IDrawableContent. DrawableContent rappresenta un elemento che può essere aggiunto all'interno della Viewport. Tutti gli elementi che estendono questa classe sapranno come disegnarsi all'interno della viewport e condividono lo stesso sistema di coordinate della lavagna virtuale stessa. All'interno della classe DrawableContent sono presenti due metodi astratti draw() e computerUpperBound() che dovranno essere utilizzati all'interno della classe che estende quest'utlima. Il primo, che rappresenta il metodo più importante, viene utilizzato per disegnare l'oggetto in delle specifiche coordinate. Il secondo viene utilizzato per calcolare il punto più in alto a destra dell'oggetto, che dovrà essere utilizzato per determinare la hit-box dell'oggetto stesso, cosi da permettere all'utente di interagirci, esso verrà trattato in dettagli nel paragrafo successivo. Ogni oggetto avrà al suo interno delle caratteristiche specifiche, a seconda se sia un'immagine oppure una figura geometria, e non si dovrà preoccupare di sapersi disegnare.

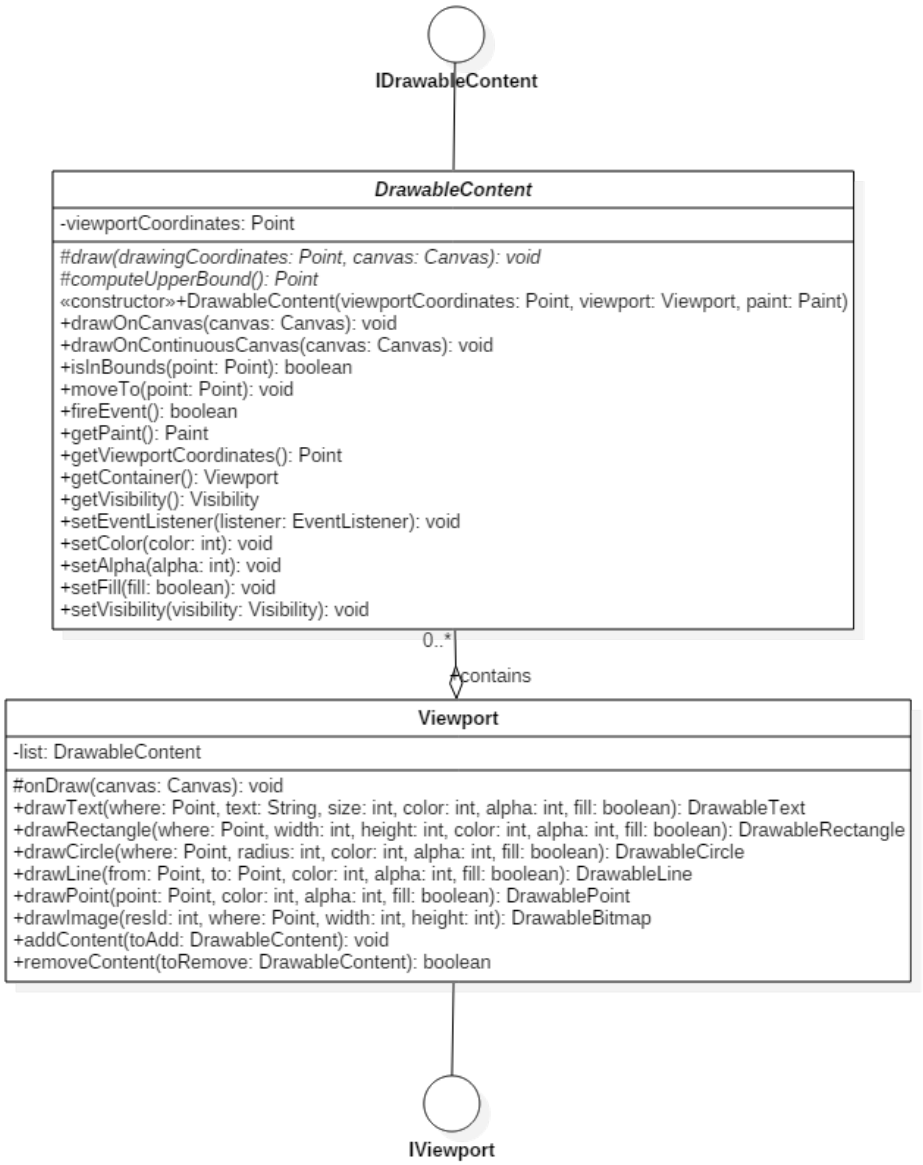

Figura 5.5: Diagramma UML Viewport - DrawableContent

Dalla figura soprastante è possibile vedere lo schema UML che mostra come sono correlate tra loro le due classi Viewport e DrawableContent. In particolare la classe Viewport contiene al suo interno una lista di oggetti di tipo DrawableContent. Attraverso il metodo draw() presente all'interno della viewport verrà richiamato il metodo draw presente all'interno della classe DrawableContent, esteso da tutti gli oggetti che possono essere disegnati all'interno della view-

port. Grazie a questo metodo, a modo di cascata, è possibile cosi aggiornare e ridisegnare gli oggetti, per esempio con delle nuove coordinate, all'interno della viewport stessa. E' importante far notare che l'oggetto DrawableContent contiene al suo interno il sistema di coordinate della viewport stessa, utilizzato per disegnare gli oggetti nella giusta posizione.

Interazione con gli oggetti virtuali I metodi forniti dall'interfaccia della classe Viewport utilizzati per il disegnamento degli oggetti virtuali, oltre che permettere di disegnare uno specifico oggetto di tipo DrawableContent all'interno della Vviewport stessa, ritornano in uscita l'oggetto a cui essi si riferiscono. Ciò avviene per permettere allo sviluppatore di interagire con questi oggetti attraverso l'utilizzo di un listener.

Cursor Di seguito viene mostrato il concetto di Cursor e come esso viene rappresentato.

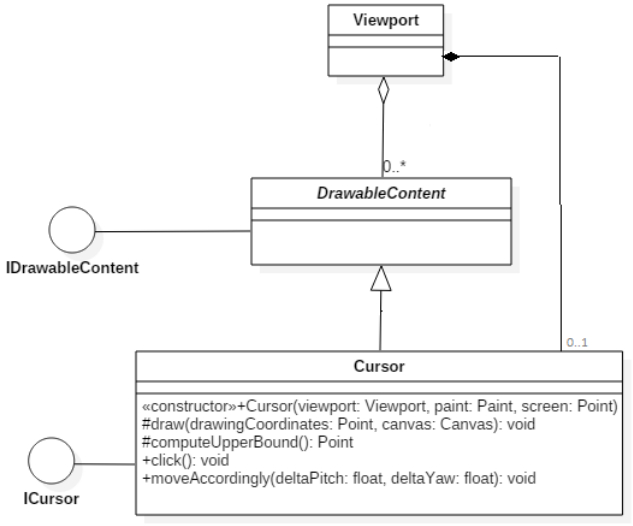

Figura 5.6: Diagramma UML Cursor

Il diagramma UML mostra come è stato progettato l'oggetto DrawableContent di tipo cursore. Questo oggetto rappresenta il mirino utilizzato dall'utente per indicare e selezionare gli oggetti virtuali presenti all'interno della viewport. Esso segue gli stessi meccanismi descritti precedentemente per tutti gli oggetti di tipo DrawableContent. All'interno della viewport esso non viene contenuto all'interno della lista di DrawableContent ma viene utilizzato come oggetto a se stante in quanto deve essere gestito singolarmente per offrire la possibilità di cliccare, attraverso di esso, sopra gli oggetti virtuali e, inoltre, ci deve

essere la possibilità di muoverlo all'iterno del FieldOfView percepito dall'utente, in un dato instante, attraverso l'interfaccia che permette il processo di human-computer interaction.

5.2.3 Comunicazione con un device di input

In questa sezione verrà trattato il modo in cui il framework permette la comunicazione con un device di input esterno. E' importante constatare che la progettazione di tale parte di progetto è stata ideata per device che permettono, in particolare, il riconoscimento di gesture e il processo di connessione e comunicazione avviene attraverso lo standard di trasmissione dati senza fili Bluetooth. Quasi tutti i moderni device e sistemi wearable utilizzano questo protocollo, infatti esso permette di ottenere una comunicazione con bassi consumi con un raggio d'azione di circa 100 metri.

Lato sviluppatore, all'interno della classe ViewportActivity, che funge da controller del sistema, sono presenti due metodi che gestiscono l'apertura e la chiusura della connessione Bluetooth verso il dispositivo che fornisce i comandi di input.

- openBluetoothConnection()
- closeBluetoothConnection()

Il framework, attraverso un Service, gestisce la connessione e la comunicazione con il device, ricevendo i messaggi ed effettuando opportuni parser, cosi da permettere l'interazione con il sistema applicativo.

Per uno sviluppatore che vuole usufruire del nostro sistema ed utilizzare un proprio device di riconoscimento di gesture come input, è stata progettata anche una classe template presente all'interno del framework. Quest'ultima si tratta di un service che può essere esteso dallo sviluppatore che ne vuole far uso, al suo interno vi sono una serie di metodi atti ad interagire col il sistema applicativo attraverso. Di seguito vengono elencati i vari comandi che il framework riconosce:

- Invio di un messaggio che permette di bloccare o sbloccare l'aggiornamento della viewport di conseguenza al movimento della testa dell'utente.
- Invio di un messaggio di click del cursore
- Invio di un messaggio di richiesta di calibrazione del sistema di aggiornamento della viewport di conseguenza al movimento della testa dell'utente.

- Invio di un messaggio di reset della posizione del cursore
- Invio di un messaggio contenente la posizione in cui deve muoversi il cursore, quindi il suo orientamento.

5.3 Sviluppo

5.3.1 Tecnologie Utilizzate

Per lo sviluppo di questo framework sono state utilizzate le seguenti tecnologie:

- Smart-glass: Moverio BT-200
- Dispositivo per input hands-free: Myo armband
- Tecnologie mobile: Smart-phone Android

Moverio BT-200 I Moverio BT-200 smart-glass (vedi anche capitolo 1.2.3) sono una tecnologia molto recente ed offrono un supporto molto interessante al caso di studio affrontato in questo progetto di tesi. Essi sono visori biculari con lenti semitrasparenti e semiriflettenti, sono considerati quindi occhiali di tipo optical see-through display. Questa loro caratteristica permette, difatti, di sovrapporre gli oggetti virtuali, visualizzati da entrambi gli occhi, sopra la realtà circostante che percepisce l'utente. Essi sono composti, oltre che dal visore, anche da una unità di controllo: un telecomando touch collegato al copro dei glasses. Al suo interno sono presenti diversi sensori tra qui: accellerometro, giroscopio, compass, videocamera, microfono. Una volta indossati essi permettono di vedere uno schermo virtuale, proiettato sull'orizzonte percepito dall'utente, di una grandezza di circa 50 pollici ad una distanza di 5 mentri. Questo dispositivo presenta un sistema operativo Android API 15, per cui sono programmabili come un qualsiasi dispositivo con questo operative-system. Sullo schermo proiettato è infatti possibile vedere una view simile a quella di un qualsiasi smart-phone o tablet versione 4.0.4 in modalità landscape. Attraverso l'unità di controllo touch è possibile sia muovere un cursore sullo schermo per interagire con il sistema, sia utilizzare dei pulsanti per effettuare determinate azioni. Tutta la fase di studio, prototipazione e sviluppo del framework è avvenuto tramite l'utilizzo di questi occhiali.

Myo armband Il dispositivo Myo armband (descrizione approfondita Capitolo 1.3.2) è stato utilizzato all'interno di questo progetto per permettere

all'utente di interagire con gli oggetti virtuali presenti all'interno della nostra applicazione. Inoltre, attraverso questo dispositivo l'utente può muovere un puntatore all'interno dell'applicazione con il movimento del suo braccio (alto-basso, destra sinistra). Grazie alle cinque gesture messe a disposizione da questo bracciale tecnologico, l'utente può cliccare un oggetto virtuale, può bloccare l'aggiornamento della viewport, scegliere o meno di attivare il puntatore e ricalibrare l'orientamento dello stesso.

Smart-phone All'interno di questo progettoè stato utilizzato anche uno smart-phone con un sistema operativo Android. Questo dispositivo ha avuto un ruolo di supporto per permettere la comunicazione tra gli smart-glass e il dispostivo di input Myo Armband. Non è infatti stato possibile collegare questi due dispositivi direttamente in quanto, visto che si è voluto utilizzare una comunicazione di tipo Bluetooth, la versione del moverio non è compatibile con quella del myo. D'altro canto, attraverso l'uso di uno smart-phone collegato agli smart-glass vi è la possibilità di utilizzare le funzionalità che esso offre, se mai ce ne sarà bisogno.

5.3.2 Linguaggi utilizzati

Lo sviluppo di questo progetto è avvenuto utilizzando come linguaggio di programmazione Java 7. L'intero sviluppo del framework è avvenuto attraverso l'utilizzo di Android Studio 2.0.

5.4 Considerazioni finali

Il framework è stato sviluppato in base alle fasi di analisi, progettazione e prototipazioni avvenute all'inizio e durante il corso del progetto di tesi. Le funzionalità principali del sistema sono state implementate con successo ed è stato possibile verificare l'effettivo funzionamento sul campo. Esso ha risposto bene ai requisiti e agli scenari descritti attraverso i casi d'uso. In particolare è possibile affermare che l'obiettivo di creare un nuovo tipo di interfaccia utente sovraimpressa alla realtà, navigabile attraverso il movimento della testa, e interagibile attraverso input basati su sistemi hands-free, è stato raggiunto con successo. E' comunque importante sottolineare che esso presenta alcuni problemi e si potrebbero applicare ulteriori miglioramenti. Innanzitutto è opportuno affermare che non si tratta di un vero e proprio framework di realtà aumentata: gli oggetti virtuali, seppur sovraimpressi alla realtà sono registrati rispetto al sistema di riferimento dell'indossatore degli occhiali. Sono stati utilizzati algoritmi di sensor-fusion per il tracking degli ogetti virtuali, applicati anche in contesti di realtà aumentata, in particolare il filtro complementare, ma gli oggetti non vengono registrati successivamente, a differenza dei normali framework e applicazioni presenti attualmente di realtà aumenta, in base alla realtà percepita dall'indossatore degli smart-glass. Un futuro miglioramento potrebbe essere proprio quello che riguarda gli algoritmi utilizzati per il tracking e la registration degli ogetti virtuali, così da costituire un framework di realtà aumenta, simili ai framework già esistenti e utilizzati oggigiorno, trattati nella parte di rassegna.
Ora come ora, inoltre, il rendering degli oggetti virtuali avviene solo per testo e immagini 2D, un futuro miglioramento potrebbe essere quello di migliorare il modulo del framework che si occupa del disegnamento e del rendering degli oggetti così da permettere anche l'inserimento di immagini e oggetti 3D. Con questo miglioramento il framework risulterebbe ancora più immersivo per l'utilizzatore dello stesso, e permetterebbe il suo utilizzo in molti altri contesti di applicazione.
Un altro futuro miglioramento potrebbe essere quelli di permettere un processo di interazione uomo-computer non solo attraverso dispositivi di input hands-free, basati sul riconoscimento di gesture, ma sarebbe utile e funzionale poter interagire con l'applicativo stesso attraverso un processo di speech-recognition, ad esempio.
In conclusione è possibile affermare che si è realizzato un nuovo tipo di interfaccia utente, non presente sin ad oggi nel mercato, che potrebbe trovare applicazione in svariati settori. Quello che si è prodotto è difatti una lavagna virtuale, sulla quale vengono posizionati oggetti non reali, navigabile attraverso la rotazione e il movimento della testa o del corpo dell'indossatore degli

smart-glass. Una delle principale caratteristiche che si è riuscito a garantire è il tracciamento di tipo spaziale degli oggetti: presupponiamo che l'utente visualizzi un oggetto virtuale alla sua destra, se esso compiesse un rotazione intorno a se stesso di 360 gradi, l'oggetto virtuale rimane nella stessa posizione in cui era visibile precedentemente. D'altro canto non viene garantita una registrazione di tipo temporale, se infatti è presente un oggetto in una data posizione rispetto all'utente, se quest'ultimo, o la realtà che lo circonda, si spostano o cambiano nel tempo, l'oggetto non sarà più visualizzato nella stessa posizione rispetto alla realtà, ma rispetto all'utente l'oggetto rimarrebbe comunque nella stessa posizione.
Questo tipo di framework trova applicazione soprattuto, secondo la mia opinione, in dei contesti sia di tipo entrateiment, ma potrebbe essere utilizzato anche all'interno delle scuole per migliorare la qualità dell'istruzione stessa, coinvolgendo di più gli studenti all'interno dell'ambito istruttivo. Inoltre troverebbe applicazione soprattutto in ambiti collaborativi, dove più persone all'interno dello stesso team debbano scambiarsi informazioni, senza distrarre l'uno dall'altro dall'attività che si sta svolgendo. Un esempio di questo tipo verrà trattato nel capitolo seguente dove verrà studiato e analizzato nello specifico un caso applicativo di questo framework che si è prodotto.

Capitolo 6

Utilizzo del Framework: Un Esempio Applicativo

Come precedentemente accennato, all'interno di questo progetto di tesi ho voluto analizzare anche un caso di applicazione del framework realizzato, che in questo capitolo tratterò nel dettaglio. L'applicazione realizzata è disponibile presso il seguente repository bitbucket: https://bitbucket.org/feedmari/app_tesi

6.1 L'idea

Dalle considerazioni fatte nei capitoli precedenti, l'innovativa interfaccia utente prodotta all'interno di questo progetto possiede le seguenti caratteristiche:

- Possibilità di visione della realtà con sovraimpressa una lavagna virtuale interattiva
- Immersività dell'utente all'interno del contesto applicativo
- Possibilità di inserire ed eliminare oggetti virtuali all'interno della lavagna virtuale
- Possibilità di interagire con gli oggetti virtuali
- Possibilità di navigare la lavagna virtuale attraverso la rotazione e il movimento della testa
- Sistema di riferimento legato al punto di vista dell'utente e non del dispositivo.
- Semplice ed immediata da utilizzare per un utente.

- Semplice da implementare per uno sviluppatore che vuole utilizzare questo progetto.

Una volta terminato il lavoro per tutto ciò che concerne il framework, si è voluto analizzare, studiare e realizzare un caso applicativo che utilizzasse quest'ultimo, così da poter, in primis, fare delle considerazioni a riguardo e capire meglio quali possano essere i futuri sviluppi ed utilizzi che un'interfaccia utente di questo tipo possa avere. In particolare si è voluto creare un contesto collaborativo all'interno della quale più utenti possono condividere simultaneamente e in real-time la stessa lavagna virtuale. Dovrà essere presente un'amministratore del sistema che invierà specifici comandi per il disegnamento di testo e forme geometriche. Gli indossatori facenti parte del sistema condivideranno la stessa lavagna virtuale e visualizzeranno gli stessi oggetti virtuali che l'amministratore vuole mostrare. E' importante evidenziare che gli oggetti non reali verranno visualizzati rispetto al sistema di riferimento della realtà che circonda l'utente, ma rispetto al sistema di riferimento dell'utilizzatore stesso. L'amministratore quindi sarà in grado, per esempio, di mostrate a tutti gli indossatori facenti parte del sistema, una determinata scritta o una determinata forma geometrica alla destra, oppure di fronte all'utente stesso. Tutti gli utenti, con il movimento e la rotazione della testa saranno in grado di visualizzare questo specifico messaggio grazie alla rotazione della testa e, se vorranno, potranno eliminarlo attraverso un click su di esso con l'utilizzo di un processo di interazione con l'occhiale indossato. Questa applicazione, inoltre, deve funzionare attraverso l'utilizzo di una connessione ad internet, cosi che sia possibile comunicare informazioni alla lavagna, e visualizzare la stessa lavagna virtuale, anche se gli indossatori degli smart-glass e l'amministratore non si trovano nello stesso luogo.
Di seguito viene una mostrata una figura riassuntiva del comportamento del caso di studio.

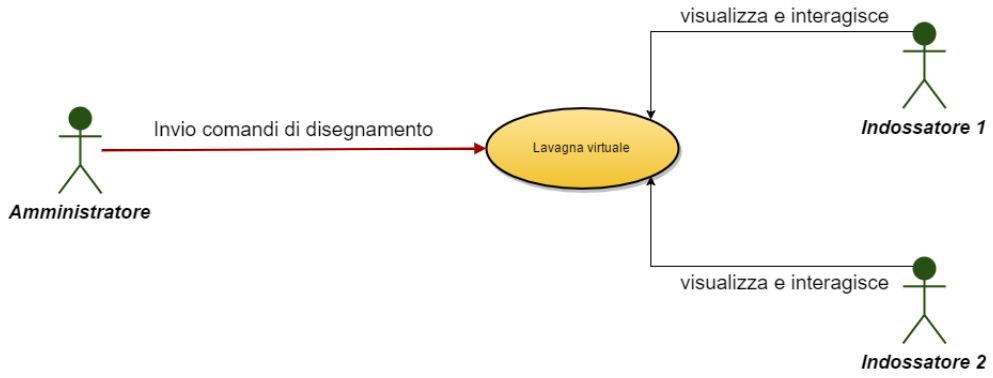

Figura 6.1: Immagine del contesto applicativo condiviso

Qui di seguito verranno elencate le funzionalità che saranno presenti sia dal lato amministratore, che dell'indossatore all'interno del sistema collaborativo e condiviso ideato.

- Amministratore
  - Possibilità di scrivere un determinato testo, con possibilità di scegliere il punto di disegnamento all'interno della lavagna virtuale, la dimensione e il colore del testo.
  - Possibilità di disegnare un rettangolo, con possibilità di scegliere il punto di disegnamento all'interno della lavagna virtuale, la larghezza, l'altezza e il colore.
  - Possibilità di disegnare un cerchio, con possibilità di scegliere il punto di disegnamento all'interno della lavagna virtuale, il raggio e il colore.
  - Possibilità di disegnare una linea, con possibilità di scegliere il punto di disegnamento all'interno della lavagna virtuale, la lunghezza e il colore.
  - Possibilità di disegnare un punto con un determinato colore all'interno della lavagna virtuale, in una qualsiasi posizione.
  - Possibilità di attivare/disattivare l'aggiornamento e navigazione della viewport di conseguenza al movimento della testa dell'utilizzatore
- Indossatori occhiali
  - Visualizzazione della lavagna e degli oggetti al suo interno in maniera condivisa e real-time, rispetto al suo sistema di riferimento.

– Possibilità, per ogni singolo utente, di eliminare un oggetto virtuale dalla propria lavagna attraverso un click su di esso.

Di seguito viene mostrato il diagramma dei casi d'uso relativo al caso applicativo sopra descritto.

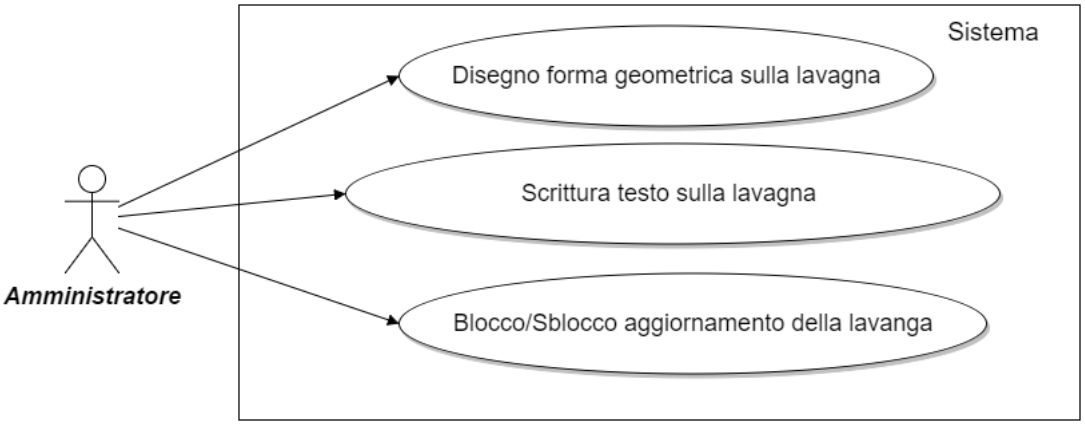

Figura 6.2: Diagramma dei casi d'uso Amministratore

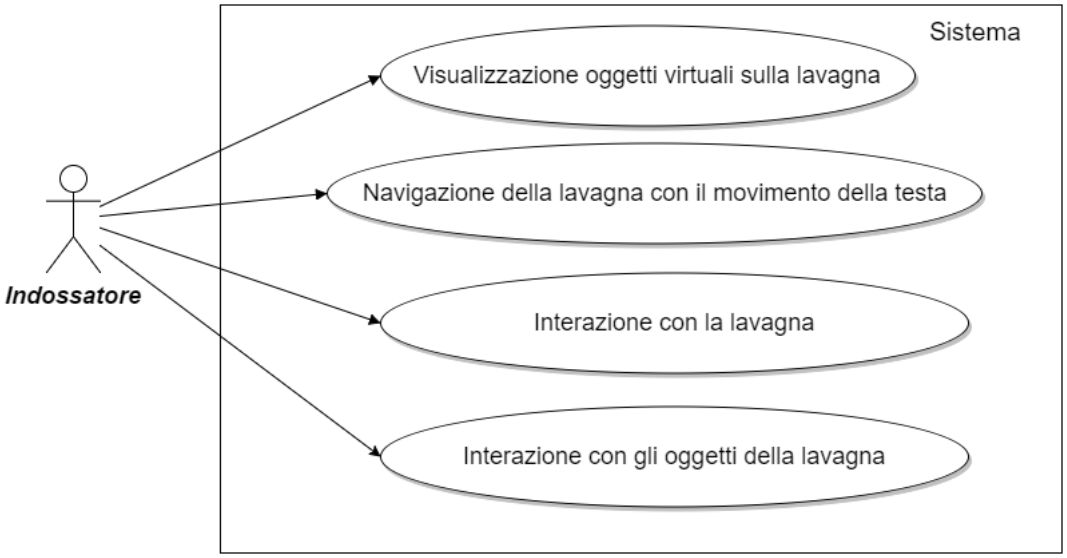

Figura 6.3: Diagramma dei casi d'uso Indossatore Smart-glass

Una sistema di questo tipo vede applicazione in diversi contesti collaborativi enterprise, dove si vogliono mandare, ad esempio, messaggi senza voler distrarre l'indossatore dell'occhiale dall'attività che sta svolgendo. E' infatti possibile, con una lavagna di dimensioni opportunamente configurate, inviare

un messaggio alla destra o alla sinistra del sistema di coordinate dell'utente, e quest'ultimo visualizzerà questo messaggio semplicemente ruotando la testa, navigando quindi la lavagna. Un secondo motivo per la quale si è deciso di produrre questo caso applicativo, è anche quello di mostrare in modo praticato, ad uno sviluppatore che vuole utilizzare il nostro framework, il suo funzionamento e utilizzo.

6.2 Design, progettazione e sviluppo

Il caso applicativo sopra descritto è fondato sulla condivisione di informazioni, inviate dall'amministratore, e visualizzate sulla lavagna virtuale presente in tutti gli gli occhiali connessi al sistema. E' importante che le informazioni arrivino sempre intatte a tutti i destinatari, senza errori o perdite. Si rende necessario quindi l'utilizzo di connessioni affidabili orientate alla connessione e full-duplex.

6.2.1 Architettura logica generale del sistema

Il sistema sarà basato su un'architettura Client-Server, in cui il server corrisponde alla piattaforma che fungerà da gestore delle informazioni visualizzate, mentre il client corrisponderà ad ogni singolo occhiale connesso al server, dove è presente un'applicazione basata sul nostro framework, essa sarà in attesa di ricevere comandi da parte del server per disegnare su di esso gli oggetti virtuali desiderati. Gli smart-glass fungeranno, quindi, da semplice interfaccia per l'utente, su cui verranno visualizzate le informazioni, con le quali potrà interagire attraverso dei click. In particolare il server rimarrà in ascolto, in attesa delle connessioni da parte dei client, alla qual invierà i vari comandi di disegnamento e gestione della lavagna virtuale. Il protocollo di comunicazione utilizzato è il TCP, implementato attraverso l'utilizzo di socket.

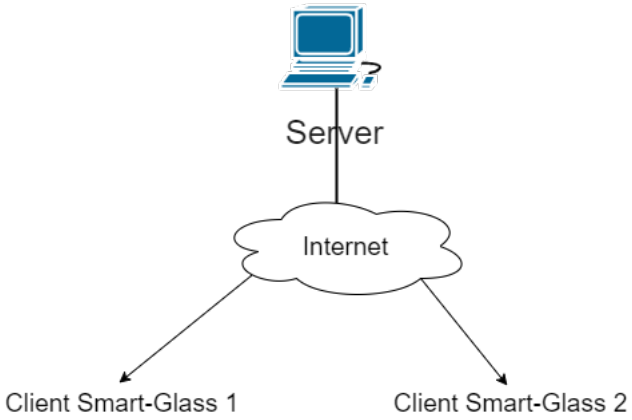

Figura 6.4: Diagramma Server multi client

6.2.2 Progettazione e Sviluppo lato Server

I compiti che il server deve eseguire sono i seguenti:

- Attesa di connessioni da parte dei client
- Salvataggio della socket di connessione con i client che ne fanno richiesta
- Attesa di input di comandi tramite console da parte dell'amministratore
- Invio dei comandi a tutti i client connessi

Gestione connessione Il server, una volta avviato, rimane in attesa di connessioni da parte dei client attraverso l'utilizzo di una ServerSocket. Quando arriva una nuova richiesta di connessione il server accetta quest'ultima e effettua il bind della socket all'interno di una classe chiamata ConnectionManager.

```
private class ClientHandler extends Thread {

     private ServerSocket serverSocket;

     public ClientHandler(ServerSocket serverSocket) {
        this.serverSocket = serverSocket;
     }

     public void run(){
        this.handleClientRequest(serverSocket);
     }

     private void handleClientRequest(ServerSocket serverSocket) {
```

```
        try {
           ConnectionManager.getConnectionManager()
                 .bindSocket(serverSocket.accept());
           System.out.println("New socket connected.");
        } catch (IOException e) {
        e.printStackTrace();
        }
     }
}
```

Listato 6.1: Classe ClientHandler che accetta le connessioni

La ConnectionManager è la classe che ha il compito di gestire tutte le connessioni. E' stata implementata utilizzando il pattern Singleton. Ad ogni nuova connessione in arrivo, all'interno della ConnectionManagar viene salvato il riferimento alla nuova socket, riferita all'ultimo client connesso.

```
public class ConnectionManager extends Thread {

   private List<Socket> socketsList = null;
   private static ConnectionManager connectionManager;

   /**
    * The private constructor for the singleton;
    */
   private ConnectionManager(){
      this.socketsList = new ArrayList<>();
   }

   /**
    * Singleton pattern for the ConnectionManager
    * class.
    *
    * @return the istance of the connectionManager
    */
   public static ConnectionManager getConnectionManager(){
      if( connectionManager == null ){
         connectionManager = new ConnectionManager();
      }
      return connectionManager;
   }

   /**
    * This method is used for bind a
```

```
     * new socket into the connection manager
     * @param socket the socket
     */
    public void bindSocket(Socket socket){
        this.socketsList.add(socket);
    }

    /**
     * This method is used for unregister a
     * new socket into the connection manager
     * @param socket the socket
     */
    public void unbindSocket(Socket socket){
        this.socketsList.remove(socket);
    }

    /**
     * This method is used for broadcasting a
     * JSONObject message to all client sockets connected
     * to the server
     * @param message JSONObject message
     */
    public void broadcastMessage(JSONObject message ){
        System.out.println(message.toString());
        System.out.println("Broadcasting Message to
            "+socketsList.size()+" client sockets");
        ThreadPoolExecutor executor = (ThreadPoolExecutor)
            Executors.newFixedThreadPool(5);
        for ( final Socket socket : socketsList ){
            if (!socket.isClosed()){
                MessageDispatcherTask task = new
                    MessageDispatcherTask(socket, message);
                executor.execute(task);
            } else {
                System.out.println("The socket " + socket.toString() +"
                    is closed!");
            }
        }
        executor.shutdown();
    }

}
```

Listato 6.2: ConnectionManager.java

Invio messaggio L'invio dei messaggi avviene tramite l'utilizzo del terminale, il sistema rimane infatti in attesa di un input dalla tastiera da parte dell'amministratore. Quando l'amministratore vuole inviare un nuovo comando, esso viene inviato alla classe MessageParser, che ha il compito di capire se il messaggio che si vuole inviare è corretto o meno, e viene codificato sotto forma di un oggetto di tipo JSON. Questo oggetto viene inviato a tutti i client attraverso l'utilizzo della classe ConnectionManager; in essa infatti, quando si vuole inviare un messaggio a tutte le socket presenti al suo interno, viene creato un nuovo task per ogni singola socket, la quale ha il compito dell'invio del messaggio all'interno del canale di comunicazione creato in precedenza. Ciò avviene attraverso l'utilizzo di un ThreadPoolExecutor. In particolare, per ogni socket presente all'interno della ConnectionManager viene creato un nuovo Thread che ha il compito dell'invio del messaggio tramite uno stream, quest'ultimo verrà eseguito e gestito attraverso il ThreadPoolExecutor; una volta terminato l'invio del messaggio, ogni singolo task verrà stoppato.

Sviluppo del server Il server è stato sviluppato utilizzando il linguaggio di programmazione Java 7, all'interno dell'ambiente di sviluppo Eclipse.

6.2.3 Progettazione e Sviluppo lato Client

Il compiti che il client deve svolgere sono i seguenti:

- Inizializzazione del framework
- Connessione al server di gestione del sistema
- Connessione al dispositivo di input tramite gesture
- Ricezione dei messaggi da parte del server
- Parsing del messaggio
- Esecuzione del comando sulla vieport a seconda del messaggio arrivato
- Assegnamento di un listener all'oggetto creato nella lavagna per eliminarlo in caso di click dell'utente.

Inizializzazione del framework Per inizializzare il framework all'interno della nostra applicazione client è sufficiente creare un'activity, chiamata MainActivity, che estende la classe ViewportActivity, presente all'interno del framework. Fatto ciò bisognerà creare un oggetto di tipo Viewport, alle quali bisogna assegnare le dimensioni rispetto alla quale deve essere più grande dello

screen dello smart-glass sulla quale girerà l'applciativo. Successivamente verrà richiamato il metodo setContentView, all'interno del metodo OnCreate dell'activity, passandogli come parametro l'oggetto che fa riferimento alla viewport. All'interno dell'onCreate verrà aperta anche la connessione bluetooth verso il dispositivo esterno di input di gesture.

```
@Override
 protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
   //init the viewport, with 20 extraWidth, 0 extraHeight
     viewport = new Viewport(this, 20, 0);
   //setting the content view
     setContentView(viewport);
   //opening the bluetooth connection for input device
     this.openBluetoothConnection();
   //init the MessageHandler istance
     MessageHandler.init(this);
   //init the client thread
     //insert here the server ip
     Thread client = new Client("" , 9742);
     client.start();
 }
```

Listato 6.3: Porzione di codice MainActivity.java

Apertura della connessione Una volta inizializzato il framework, viene creato l'oggetto che fa riferimento alla classe che si occupa di gestire la connessione con il Server. Questa classe, quando inizializzata, prende in ingresso l'ip del server a cui connettersi e la porta sulla quale il server vuole aprire la socket. Questa classe girerà su un thread a parte. All'interno di essa viene aperta una socket di comunicazione con il server ed essa si metterà in ascolto in attesa di messaggi in arrivo dalla socket.

Arrivo di un messaggio, parsing ed esecuzione I messaggi che arriveranno verranno salvati all'interno di un buffer presente all'interno del thread che si occupa di gestire la comunicazione; esso verrà convertito in un oggetto JSON. Dall'oggetto JSON viene estrapolato il nome del comando che il server ha inviato al client. Per la comunicazione tra il Thread che si occupa della gestione della connessione con il server, e il Main Thread che si occupa del rendering degli oggetti, si è utilizzato un Handler. La classe che modella l'handler è stata chiamata MessageHandler, essa infatti estende la classe Hand-

ler. La classe MessageHandler è utilizzabile attraverso il pattern Singleton. Per permettere la comunicazione tra il MessageHandler e l'activity principale si è creata un'interfaccia con il nome VireportCallbackIntarface. All'interno di essa sono presenti tutti i comandi che possono essere inviati alla viewport presente all'interno della MainActivity. La MainActivity implementa questa interfaccia mentre la classe MessageHandler contiene al suo interno un riferimento a questa interfaccia. Il riferimento a questa interfaccia viene inizializzato quando si inizializza l'activity stessa, passandogli se stessa come parametro. Quando arriva un messaggio da parte del server, da esso viene estrapolato il nome del comando. A seconda del nome del comando che arriva, all'interno della classe che si occupa della gestione delle connessione, viene creato un oggetto di tipo Message (classe utilizzata di default per l'invio di messaggi all'handler) nella quale viene messo il riferimento al MessageHandler, quindi il destinatario, un riferimento al codice del metodo che si vuole chiamare e viene allegato al messaggio i valori dei parametri da assegnare, attraverso l'utilizzo di un bundle. L'handler riceverà questo messaggio e lo ridizionerà all'activity principale attravero il riferimento all'interfaccia ViewportCallbackInterface. Di seguito viene mostrato un diagramma UML che mostra l'interazione tra il MessageHandler e la classe MainActivity.

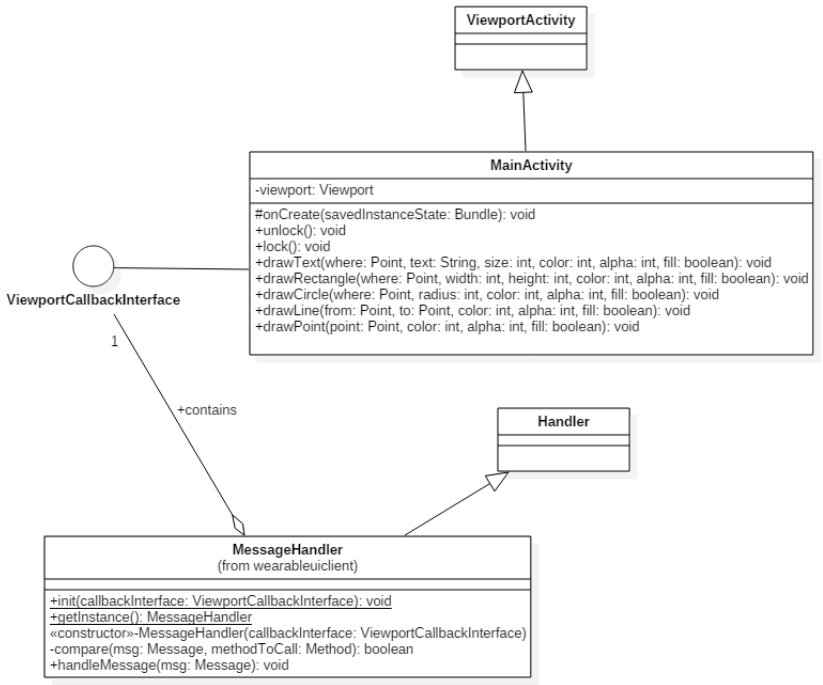

Figura 6.5: Diagramma UML MainActivity - MessageHandler

L'activity principale riceverà il messaggio e lancerà il corrispettivo metodo sulla viewport. Infine, se il comando arrivato si tratta di un comando di disegnamento di un oggetto virtuale, a quest'ultimo verrà assegnato un listener, esso avrà il compito di ascoltare l'eventuale click dell'indossatore sull'oggetto virtuale, ed eliminare l'oggetto stesso se ciò accade.

6.3 Considerazioni finali

Quello che si è voluto produrre attraverso questo caso applicativo è una semplice demo di come il framework potesse essere esteso per essere utilizzato in un contesto collaborativo, dove un gestore del sistema può mostrare specifiche informazione a tutti i dispositivi smart-glass connessi al server dalla quale egli invia i comandi di gestione e disegnamento. L'applicativo è stato infatti sviluppato in base alle fasi di analisi e progettazione avvenute successivamente alla conclusione del lavoro sul framework stesso, sulla quale l'applicativo si basa. Le funzionalità principali del sistema sono state implementate con successo. Esso, dopo una fase di testing, ha risposto bene ai requisiti e agli scenari descritti attraverso i casi d'uso. E' importante comunque affermare che il caso applicativo potrebbe essere migliorato in molteplici aspetti: esso non presenta un controllo adeguato sulla gestione delle connessioni/disconnessione del client al server, inoltre le performance potrebbe essere migliorato in quanto, ora come ora, si è deciso di ottimizzare una fase connessione, da parte del server, per un numero massimo di 5 client. Sarebbe inoltre opportuno, per avere una buona user-experience che l'invio dei messaggi di gestione del client, da parte del server, non avvenisse attraverso l'uso di una cosole di comando, ma attraverso l'utilizzo di un interfaccia utente grafica. Si pensi ad un pannello che rappresenta la lavagna virtuale, visualizzata da tutti i client, sulla quale il gestore del sistema inserisce, attraverso un processo di drag-and-drop, gli elementi virtuali. Questo tipo di user-experience risulterebbe sicuramente molto più efficiente e comoda da utilizzare. Un altro futuro miglioramento potrebbe essere l'utilizzo di una comunicazione peer-to-peer per connettere gli smart-glass l'uno con l'altro, tutti gli utenti potrebbero cosi disegnare sulla medesima lavagna e visualizzare gli stessi contenuti. Sarebbe sicuramente una soluzione molto efficacie ed utile per ambiti e contesti enterprise di tipo collaborativo, dove più persone lavorano all'interno dello stesso team e vogliono scambiarsi messaggi senza distrarre l'attenzione dall'attività che stanno svolgendo. E' importante sottolineare inoltre, che il caso applicativo sopra trattato, dal punto di vista di come è stato modellato e sviluppato lato client, è facilmente inseribile all'interno del framework stesso come funzionalità aggiuntiva per la connessione ad

un server che vuole gestire l'invio di informazioni a tutti i client, quindi agli occhiali, connessi all'interno del medesimo sistema di condivisione.

Conclusioni

E' possibile affermare che l'obiettivo di creare un nuovo tipo di interfaccia utente sovraimpressa alla realtà, navigabile attraverso il movimento della testa, e interattiva attraverso l'utilizzo di input hands-free, in particolare con l'utilizzo di un dispositivo basato sul riconoscimento di gesture, è stato raggiunto.
Inoltre attraverso il caso applicativo è stato possibile verificarne l'utilità anche in un contesto di condivisione delle medesime informazioni tra più utenti.
Il caso applicativo prodotto, inoltre, dal punto di vista di come è stato progettato e sviluppato lato client, è facilmente importabile all'interno del framework stesso, come estensione per permettere la possibilità di controllo della viewport, presente all'interno degli occhiali connessi al sistema, da parte di un utente esterno che funge da controllore.
Il framework è stato progettato, prototipato e sviluppato da un team composto da due persone: me, Federico Marinelli e lo studente Federico Giannoni. Abbiamo partecipato assieme alle fasi di anasili, progettazione e prototipazione del sistema. Successivamente, nella fase di sviluppo io mi sono occupato principalmente del rendering degli oggetti e nello studio e implementazione degli algoritmi applicativi per il tracking degli oggetti virtuali, anche se comunque lo sviluppo è avvenuto in pair-programming sul medesimo computer.
Lavorare in team è stato un'esperienza significativa, che mi ha permesso di realizzare un progetto di cui mi ritengo pienamente soddisfatto. Grazie a questo progetto di tesi ho infatti potuto apprendere importanti nozioni riguardanti la realtà aumenta, l'applicazione della sensor-fusion all'interno del contesto del tracking degli oggetti virtuali, dettagli riguardanti i dispositivi wearable, in particolare smart-glas e infine tutto il processo che riguarda la human-computer interacion hands-free; tutto questo sia da un punto di vista teorico, ma anche e soprattuto pratico.
In definitiva, è possibile concludere affermando che l'interfaccia ottenuta, al termine di questo progetto di tesi, ha tutte le potenzialità per essere ulteriormente studiata e applicata in diversi ambiti, permettendo al fruitore del sistema di interfacciarsi facilmente e in maniera naturale con le tecnologie utilizzata all'interno del contesto applicativo realizzato, aiutandone la diffusione e

lo sviluppo, ottenendo un'applicazione che permette all'utente finale di essere libero di scegliere se visualizzare informazioni aggiuntivo o meno, interfacciandosi con essa attraverso l'utilizzo dispositivi di human-computer interacion non convenzionali, in particolare hands-free.

Ringraziamenti

Al termine di questo percorso di studi vorrei ringraziare chi mi è stato vicino e mi ha sostenuto dandomi un grande supporto emotivo. Per questo vorrei ringraziare i miei familiari, in particolare i miei genitori che hanno sempre creduto in me, dandomi un grandissimo aiuto sia economico ma anche morale. Inoltre, vorrei ringraziare tutti i miei amici più cari che, in questi anni, mi hanno sempre aiutato a rilassarmi dallo studio, ma anche dandomi parole di conforto e sostegno, capendomi quando ero costretto a studiare. Infine vorrei ringraziare tutti i miei compagni di università , con i quali ho condiviso diverse sofferenze, ma altrettante gioie e soddisfazioni.

Bibliografia

[1] L. Sydanheimo, M. Salmimaa, J. Vanhala, and M. Kivikclski. Wearable and ubiquitous computer aided service, maintenance and overhaul. IEEE International Conference, 1999.

[2] Steve Mann. Wearable computing: A first step toward personal imaging. Journal, Computer, Volume 30 Issue 2, 1997.

[3] Microsoft. Microsoft Hololens, Website. http://www.microsoft.com/microsoft-hololens.

[4] Epson. Epson Moverio BT-200. Website. http://www.epson.com/MoverioBT200

[5] ThalmicLabs. Myo. Website, http://www.myo.com, 2013.

[6] F. Frangeskides, A. Lanitis, *A Hands-Free Non-Invasive Human Computer Interaction System*. Chapter. Advances in Systems, Computing Sciences and Software Engineering, pp 235-242, 2006.

[7] P. Mistry and P.Maes. Sixthense: A wearable gesture interface. ACM SIGGRAPH ASIA 2009 Sketches, Article No. 11, 2009.

[8] P. Milgram and F. Kishino. A taxonomy of mixed reality visual display. IEICE Paper, 1994.

[9] Kerri Smith, *Brain implant allows mute man to speak*. Article, nature.com, 2008.

[10] D Drascic, P Milgram. Perceptual issues in augmented reality. Conference, Stereoscopic Displays and Virtual Reality Systems III, 1996.

[11] Google Android Developer. Motion Sensors. Website, https://www.developer.android.com/guide/topics/sensors.