ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

APPLICAZIONI DI MIXED REALITY E MICROSOFT HOLOLENS: STUDIO E RIPROGETTAZIONE DEL CONCETTO DI OLOGRAMMA E DELLA RELATIVA GESTIONE ALL'INTERNO DI UN'APPLICAZIONE OLOGRAFICA

Elaborato in
PROGRAMMAZIONE DI SISTEMI EMBEDDED

Relatore
Prof. ALESSANDRO RICCI

Co-relatore
Ing. ANGELO CROATTI

Presentata da
MATTIA ORIANI

Prima Sessione di Laurea
Anno Accademico 2015 – 2016

PAROLE CHIAVE

Augmented Reality

Mixed Reality

Microsoft Hololens

Holographic Applications

Hologram

"Quando riesci a cambiare il modo in cui vedi il mondo, puoi cambiare il mondo che vedi"
- Microsoft.

Indice

Introduzione **ix**

1 Augmented e Mixed Reality: Panoramica **1**
1.1 Cos'è la realtà aumentata 1
1.2 Modalità di realtà aumentata 2
1.2.1 Applicazioni più diffuse 3
1.3 Cenni di Realtà Virtuale 4
1.4 Confronto tra Realtà Aumentata e Realtà Virtuale 5
1.5 Mixed Reality 6
1.6 Dispositivi e ologrammi 7
1.7 Principali soluzioni per sviluppare Mixed Reality 8

2 Microsoft Hololens **9**
2.1 Cos'è Microsoft Hololens 9
2.1.1 Dettagli Tecnici 10
2.2 Funzionamento Generale 11
2.2.1 Building Blocks: Moduli Principali Hololens 12
2.3 Principi di Base 14
2.3.1 Sistemi di coordinate 15
2.3.2 Gaze 19
2.3.3 Gesture Input 19
2.3.4 Vocal Input 22
2.3.5 Spatial Mapping 22
2.3.6 Spatial Sound 26

3 Creare Applicazioni Microsoft Hololens **29**
3.1 Cos'è un'applicazione per Hololens 29
3.2 Strumenti di Utilizzo 30
3.2.1 Unity 30
3.2.2 DirectX e Windows Holographic API 31
3.2.3 Hololens Emulator 31
3.3 Funzionamento Generale in DirectX 32

3.3.1 Struttura Architetturale 32
3.4 Principi di base in DirectX 36
3.4.1 Sistemi di Coordinate in DirectX 36
3.4.2 Gaze in DirectX 38
3.4.3 Gesture in DirectX 38
3.4.4 Spatial Mapping in DirectX 40
3.5 Esempio di Holographic App 42
3.5.1 Funzionalità principali 42
3.5.2 Considerazioni sull'esempio 43

4 HoloFrameOfMind **47**
4.1 Architettura principale 48
4.2 Progettazione dell'ologramma 50
4.3 Dettagli d'implementazione 54
4.3.1 HoloModel 54
4.3.2 HoloView 56
4.3.3 HoloController 58
4.3.4 HoloRenderer 59
4.3.5 HoloManager 62
4.3.6 Overview utente 66

Conclusioni **69**

Ringraziamenti **71**

Bibliografia **73**

Introduzione

Nel corso degli anni, la tecnologia ha assunto sempre più un ruolo chiave e fondamentale all'interno della vita delle persone. Si usano dispositivi e supporti tecnologici in ogni campo, e per qualsiasi motivo, dalla medicina all'ingegneria, dall'agricoltura fino allo sport, ma soprattutto la vita di ogni persona è supportata da almeno un dispositivo tecnologico, per esempio uno smartphone, che viene utilizzato per distrarsi dalla vita quotidiana, per giocare, per rimanere informati sui fatti del mondo. Quando una persona non conosce qualche informazione, utilizza generalmente lo smartphone o il computer per ottenere tale informazione, e i nuovi sviluppi della tecnologia informatica vanno sempre più in questa direzione cercando di fornire all'utente delle informazioni in tempo reale relative alle sue richieste. Per ovviare a questa "necessità" sono nati i dispositivi di augmented reality, che svolgono le funzioni appena indicate, permettendo l'aggiunta di informazioni anche in momenti critici, magari nei quali non si ha molto tempo a disposizione e si necessita di informazioni in tempo reale. Questi dispositivi potrebbero rappresentare il futuro dell'informatica, o crearne un ramo significativo, tanto che da vari anni le università e i colossi aziendali informatici eseguono ricerche e sviluppo in questa direzione. L'uso dei dispositivi di augmented reality, si sta sempre più specializzando, fino a diventare dispositivi in grado di svolgere qualsiasi funzione, dall'ottenere informazioni, al giocare, fino ad ad essere un supporto fisso alla vita di una persona. Più che i contenuti o le funzionalità che si possono fare con questi dispositivi, cambia il modo di farlo, è questa la vera innovazione.
In questa tesi si andrà ad analizzare le tipologie di tecnologie disponibili allo stato dell'arte attuale, descrivendone le caratteristiche principali, per dare un'overview sulle potenzialità di ogni tipo di trasformazione della realtà umana. In particolare verrà analizzato un caso innovativo: un dispositivo creato da Microsoft, chiamato Hololens, il quale permette ancora di più l'integrazione fra il mondo reale e le informazioni virtuali. Questo dispositivo fornisce una nuova visione del mondo, portando lo stato dell'arte ad un livello più alto, permettendo l'ideazione di applicazioni innovative, di qualsiasi genere, dal supporto alla vita quotidiana al puro svago che necessita una persona.
Saranno descritte tutte le parti fondamentali, incluso il funzionamento del di-

spositivo Microsoft Hololens, per fornire al lettore una panoramica dettagliata su quali potenzialità abbia questo dispositivo e il modo in cui sfruttarle. In particolare viene posta l'attenzione sulla creazione di applicazioni per questo device, spiegandone le componenti. L'obiettivo di questo lavoro è quello di riprogettare a livello ingegneristico l'architettura e le componenti fondamentali di un'applicazione, per raggiungere una creazione semplificata, un migliore sviluppo ed un migliore utilizzo di tale tecnologia.

Capitolo 1

Augmented e Mixed Reality: Panoramica

In questo capitolo verranno descritte le modalità con cui è possibile modificare la realtà, aggiungendo informazioni utili, creando una realtà nuova e facendoci immergere l'utente, fino a descrivere come fare un mix tra le due modalità. Di ogni modalità verrà data inizialmente una definizione, per introdurre all'argomento e comprendere bene cosa si intende. Verranno poi spiegate le modalità di realizzazione di ogni tipo di trasformazione della realtà, fino a descriverne le applicazioni più diffuse allo stato attuale dell'arte. Verranno fatti confronti fra le varie modalità, per permettere di distinguerle in modo chiaro, giungendo così all'esplorazione di Microsoft Hololens con una chiara visione dell'argomento generico, e del campo del quale si sta trattando.

1.1 Cos'è la realtà aumentata

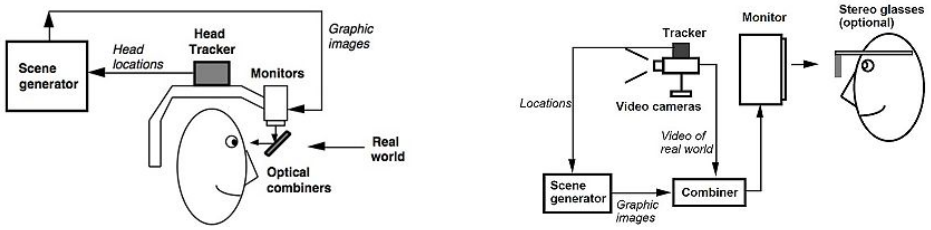
Un tipo di tecnologia interessante, e in elevata fase di sviluppo negli ultimi anni, è la realtà aumentata, cioè un modo nuovo di vedere la realtà. La realtà aumentata permette di aggiungere al mondo reale delle informazioni che una persona normale non sarebbe in grado di percepire con i normali cinque sensi umani. E' una tecnologia in continuo sviluppo, che ha subito un'accelerazione notevole negli ultimi anni, testimoniata dalla presentazione di dispositivi che mettono in atto questa tecnologia da parte delle più grandi aziende informatiche. Il concetto di realtà aumentata è riscontrabile anche in dispositivi più semplici, per esempio uno smartphone nel caso si stesse esplorando una città, oppure degli occhiali che proiettano delle immagini davanti agli occhi, chiamati smartglasses [5]. Le caratteristiche principali di un sistema di realtà aumentata riguardano il fatto di saper combinare oggetti reali insieme ad oggetti virtuali in un unico ambiente reale, unitamente al fatto che questa combinazione venga

fatta in tempo reale, in base alle richieste dell'utente e di ciò che succede nel mondo.
I dispositivi sono dotati di sensori, di vario tipo relativamente all'aspetto che si vuole aumentare della realtà. L'aumento di qualsiasi dei cinque sensi è considerato realtà aumentata, dall'aggiunta di contenuti visivi, all'uso di suoni artificiali, fino alla simulazione di odori. Generalmente le applicazioni di realtà aumentata si occupano della parte grafica, quindi aggiungendo dei contenuti alla realtà, ma nel caso si volesse ampliare il senso dell'udito, potrebbero essere usati degli auricolari per simulare dei suoni. Esistono anche altri tipi di dispositivi che implementano il concetto di realtà aumentata, quali visori head-locked o lenti see-through, cioè dispositivi fissati alla propria testa, o lenti in cui si può guardare attraverso.

1.2 Modalità di realtà aumentata

Come appena accennato, ci sono vari hardware per realizzare esperienze di realtà aumentata, dal semplice smartphone, a veri e propri visori. Usando dispositivi see-through si ha ovviamente un'esperienza più completa, continuativa e immersiva, rispetto ad uno smartphone, dando così la possibilità di svolgere funzioni più complesse. Si effettuano applicazioni di realtà aumentata anche usando dei visori che oscurano la vista della persona. In questo caso ci sono delle telecamere che riprendono la realtà e la combinano con gli elementi aggiuntivi, creando un video che verrà mostrato all'utente sul suo visore. Ci sono vari vantaggi sull'usare delle lenti see-through rispetto ad un visore video di realtà aumentata, come spiegato da vari articoli scientifici [3]. Un vantaggio riguarda la qualità dell'esperienza, dato che usando una riproduzione video della realtà, quest'ultima viene poi visualizzata ad una qualità inferiore rispetto a quella che l'utente potrebbe vedere con i propri occhi, cosa che invece non succede nel caso si utilizzino lenti see-through. E' normale che ci sia questa differenza, in quanto la ripresa e la riproduzione della realtà, non sarà mai qualitativamente valida come invece lo è guardando la realtà direttamente con i propri occhi. E' bene considerare anche un fattore di sicurezza nel caso si utilizzino visori per creare applicazioni di augmented-reality, infatti nel caso la persona utilizzasse delle lenti see-through e il sistema smettesse di funzionare e di conseguenza di aggiungere le informazioni alla realtà, il mondo continuerebbe ad essere visibile attraverso le lenti. Nel caso in cui utilizzasse un visore video e questo smettesse di funzionare, l'utente avrebbe una visione totalmente oscurata, non rendendosi più conto di cosa stia vedendo. E' importante considerare questi aspetti in base al tipo di applicazione che si sta facendo e all'esperienza che si vuole offrire all'utente, perchè nel caso si trattasse di

applicazioni usate in ambito medico, il fatto di non riuscire a vedere nel caso smettesse di funzionare il sistema può rivelarsi fatale, invece nel caso si trattasse di un videogioco, beh, il danno sarebbe decisamente meno grave, se non nullo. Ci sono anche alcune differenze riguardo la computazione, infatti nel caso di dispositivi che riproducono video, ci si dovrebbe occupare di due flussi di informazione, quello riferito al video acquisito, e quello relativo alle informazioni da aggiungere, unitamente alla combinazione fra questi due flussi. E' facilmente intuibile che nel caso di lenti see-through, ci si occuperà prevalentemente delle informazioni da aggiungere e di metterle in relazione con il mondo, dato che non è necessario riprendere la realtà. Nelle figure sottostanti, si può osservare il metodo di funzionamento di un dispositivo con visore see-through e di un dispositivo con riproduzione video [3]. Per approfondire l'argomento sui tipi di dispositivi di realtà aumentata, consultare l'articolo in bibliografia [5].



(a) Funzionamento di un dispositivo con visore see-through

(b) Dispositivo basato su riproduzione video e monitor

Figura 1.1: Raffigurazione dei metodi di funzionamento e principali differenze tra i due tipi di dispositivi

1.2.1 Applicazioni più diffuse

Nell'ultimo decennio, la realizzazione di applicazioni di questo tipo ha subito una notevole accelerazione, e come spiegato prima le grandi multinazionali informatiche hanno già creato i propri dispositivi all'avanguardia. La realtà aumentata è già utilizzata in alcuni ambiti, quali, medico e militare [3]. In ambito militare sono già ampiamente utilizzati dei caschetti sui quali vengono proiettate delle informazioni in tempo reale, per permettere per esempio al pilota di aereo di rimanere aggiornato senza distogliere l'attenzione dal suo compito principale. I dispositivi che sono in commercio e diffusi su larga scala, come iPhone o iPad[1], possono essere già considerati dispositivi di realtà aumentata, in quanto possiedono sensori come accelerometro, GPS, e magne-

[1]Sito Ufficiale Apple, Produttore di iPhone e iPad: `www.apple.com`

tometro, e sono predisposti alla creazione di applicazioni di augmented reality [6].
Google ha commercializzato i Google Glass, nel 2012. Questo dispositivo permetteva l'aggiunta di informazioni alla realtà visualizzandole davanti ai propri occhi, unitamente all'acquisizione di immagini e informazioni sulla realtà stessa. E' un dispositivo che va tenuto in modo fisso sugli occhiali, quindi legato alla testa per usufrire in pieno delle sue funzionalità. Purtroppo questo progetto non ha riscontrato il successo sperato, ma rimane un esempio di dispositivo di realtà aumentata. Nel 2015 Microsoft annuncia un progetto innovativo, chiamato Microsoft Hololens, di cui all'attuale stato dell'arte è stata solo rilasciata una versione per sviluppatori. Questo device è un vero e proprio computer, con un visore di tipo see-through e vari sensori, il quale combina alcuni elementi di augmented reality, insieme ad altri concetti di mixed reality, spiegati in seguito [2] [6].

1.3 Cenni di Realtà Virtuale

Nelle applicazioni di realtà virtuale, ci si pone come obiettivo principale, la creazione di un mondo totalmente virtuale e parallelo a quello reale, in cui far immergere l'utente. Il mondo che verrà creato può essere una riproduzione di un ambiente reale, oppure un mondo tutto nuovo, inesistente nella realtà. All'interno del mondo virtuale l'utente può comportarsi esattamente come se fosse nella realtà, reagendo agli stimoli e interagendo con gli oggetti che si trovano attorno a lui. Non si creano solamente videogiochi per questo tipo di trasformazione della realtà a differenza di come si può immaginare, ma vengono realizzate anche rappresentazioni alternative del mondo stesso, per esempio la simulazione di un volo aereo. Per ottenere lo scopo prefissato, i dispositivi di realtà virtuale sono principalmente dei caschetti che coprono tutto il campo visivo, per avvolgere e coinvolgere totalmente la persona, nella maggior parte dei casi questi dispositivi oltre al visore in cui viene proiettato il mondo, sono dotati anche di auricolari per raggiungere in modo completo l'immersione dell'utente. La virtual reality è abbastanza diffusa e alcuni dei dispositivi più conosciuti al pubblico sono: Oculus Rift (vd. fig.1.2a), HTC Vive (vd. fig. 1.2b), Hdk 2, rispettivamente di proprietà di Facebook, HTC, e Osvr, una comunità open source dedicata alla virtual reality nei videogiochi. Dato che gli smartphone si sono evoluti in modo esponenziale diventando dei piccoli computer, ci sono alcuni dispositivi quali Gear, prodotto da Samsung,

[2] Sito Ufficiale Microsoft Hololens `https://www.microsoft.com/microsoft-hololens/en-us`

che permettono di incorporare lo smartphone all'interno di un visore e usarlo come dispositivo di realtà virtuale.

(a) Oculus Rift (b) HTC Vive

Figura 1.2: Esempi di dispositivi di realtà virtuale.

1.4 Confronto tra Realtà Aumentata e Realtà Virtuale

E' importante distinguere Realtà Aumentata e Realtà Virtuale. A grandi linee, già dalle parole stesse si intuisce la differenza principale. Nel caso ci si occupi di realtà aumentata infatti, si parte dalla realtà vera e propria, aumentandola, con l'aggiunta di nuove informazioni, a correlazione del mondo reale. E' bene intendere che con lo stesso principio con cui si aggiungono dei contenuti o delle informazioni alla realtà, si possono anche togliere o nascondere per renderla migliore, anche in questo caso si parla di realtà aumentata. Nella realtà virtuale invece viene creato un nuovo mondo, nel quale l'utente verrà totalmente immerso, con lo scopo di fare dimenticare tutto ciò che gli sta attorno, come se fosse realmente nel mondo creato e che è in grado di vedere. Non si fa quindi accenno alla realtà vera e propria come nell'augmented reality, ma se ne crea una parallela in cui l'utente dovrà entrare. Le applicazioni e i dispositivi di realtà aumentata hanno requisiti minori rispetto alle applicazioni e ai dispositivi di realtà virtuale[3]. Per quanto riguarda i dispositivi, nel caso di augmented reality basta una lente see-through o un dispositivo mobile per aggiungere informazioni alla realtà, invece nel caso di virtual reality c'è la necessità di avere dispositivi più avvolgenti, non basterà più una lente see-through per proiettare l'utente in una realtà virtuale, ma servirà un caschetto o un dispositivo che avvolga totalmente l'utente. Questo richiede che il dispositivo abbia tecnologie diverse, in quanto deve riprodurre totalmente da zero la realtà: questa necessità si rispecchia nell'utilizzo di una gamma di colori maggiore rispetto ai dispositivi di realtà aumentata, di un utilizzo di rendering delle immagini ancor più alto e preciso, cosa che invece

nella realtà aumentata non è necessaria visto che l'utente vede già la realtà vera e propria e quindi non c'è il bisogno, nè di riprodurre immagini ad una qualità elevata per simularle realmente, nè di utilizzare una vasta gamma di colori per riprodurre tutte le sfumature del mondo. C'è un problema da considerare, riguardo entrambi i due mondi, riguardante l'interazione dei sensori col mondo e il rilevamento della posizione dell'utente, e anche in questo caso ci sono notevoli differenze tra realtà aumentata e realtà virtuale. Nel caso di realtà aumentata c'è la necessita di sapere con precisione dove si trova l'utente e che movimenti sta facendo, questo implica un notevole uso di informazioni rilevate dai sensori, perchè per avere dei contenuti aggiuntivi alla realtà questi dovranno essere messi in relazione con il mondo stesso, avendo quindi ben chiara la posizione dell'utente e la sua visione del mondo in un preciso momento. Nel caso ci trovassimo all'interno di un museo, un'applicazione di realtà aumentata potrebbe indicarci quale sia la storia dell'opera che si trova davanti a noi, e in questo caso serviranno dei sensori per capire bene in quale punto della realtà ci troviamo. Questo implica anche una computazione aggiuntiva per la corretta visualizzazione delle informazioni, cosa che in realtà virtuale non avviene in quanto si tratta già di un mondo parallelo. Nel caso di realtà virtuale quindi non c'è la necessità di sapere con accuratezza cosa stia facendo l'utente nel mondo reale, in quanto ogni suo movimento è rispecchiato all'interno di una realtà virtuale, nella maggior parte dei casi basterà quindi tracciare i suoi movimenti della testa per capire in quale parte del mondo virtuale stia guardando elaborandone poi i contenuti. Ne consegue un diverso utilizzo dei sensori fra i due tipi di realtà. Per approfondire l'utilizzo dei sensori in realtà aumentata (Azuma, 1997).

1.5 Mixed Reality

Dopo aver presentato i concetti di augmented reality e virtual reality, e aver analizzato le principali differenze, in questa sezione verrà discusso il concetto di mixed reality. Milgram definisce la mixed reality come "anywhere between the extrema of the virtuality continuum" [7]. E' interessante analizzare questa definizione in quanto esprime in modo sintetico e chiaro il concetto di mixed reality. La virtuality continuum (vd. fig. 1.3) è un range che pone ai due estremi il mondo reale e il mondo virtuale, indica quindi i passaggi da un mondo all'altro, i quali comprendono la sovra citata realtà aumentata, fino ad arrivare alla realtà virtuale.

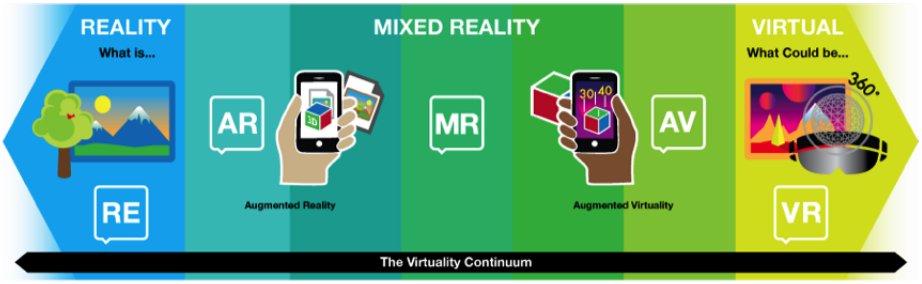


Figura 1.3: Rappresentazione della virtuality continuum

Mixed reality è un insieme di tutto ciò, infatti un'applicazione di mixed reality combina oggetti virtuali, informazioni di realtà aumentata, e oggetti reali, facendoli coesistere in un unico mondo, e mettendoli in grado di reagire in tempo reale [9]. La mixed reality consiste quindi nella generazione di un mondo nuovo, diverso da uno totalmente reale, e diverso da uno totalmente virtuale, ma un mondo che contiene una parte di uno e una parte dell'altro, perfettamente combinate tra loro. A livello storico la mixed reality è un'evoluzione e una derivazione della virtual reality [9], della quale i primi esempi sono stati prodotti intorno al 1960 [10]. Generalmente i dispositivi che permettono l'uso di applicazioni di mixed reality, sono head-locked, cioè caschetti, o comunque dispositivi fissati alla testa della persona. Questi dispositivi sono molto simili a quelli utilizzati per l'augmented reality e descritti in precedenza, con la sola differenza che vengono implementate applicazioni di mixed reality invece che di realtà aumentata. Sono sempre dotati di lenti see-through, e di sensori di vario tipo, quali accelerometro, GPS, giroscopio, e sistemi audio.

1.6 Dispositivi e ologrammi

Una tecnica per visualizzare degli oggetti nella realtà aumentata e nella mixed reality, è quella di produrre ologrammi. Un ologramma non è altro che un incrocio di fasci di luce che riproduce un'immagine o un oggetto in tre dimensioni, osservabile poi da qualunque prospettiva. Questa tecnica, rispetto alla semplice visualizzazione di immagini tridimensionali, rende meno affaticato l'occhio ed è sempre più in via di espansione [11]. La rappresentazione degli oggetti sotto forma olografica è in continuo aumento fra i dispositivi head-mounted, che avranno dei proiettori di fasci di luce, i quali renderizzano l'oggetto desiderato davanti agli occhi dell'utente, oppure nella parte del mondo in cui si vuole inserire l'informazione, aumentandone la realtà o combinandola con elementi virtuali. Un esempio di questi dispositivi è Hololens,

annunciato da Microsoft nel 2015 [2]. Questo dispositivo incarna perfettamente il concetto di mixed reality, in quanto le applicazioni integrano la realtà vera e propria con ologrammi rappresentanti scene o semplici oggetti virtuali, andando così a creare una nuova versione dell'ambiente in cui ci si trova.

1.7 Principali soluzioni per sviluppare Mixed Reality

Ci sono varie soluzioni sia hardware che software per accedere al mondo dell'augmented reality, della virtual reality e della mixed reality. Un'azienda sempre più in via di espansione è Meta, la quale ha prodotto due innovativi dispositivi di augmented reality, chiamati Meta 1 e Meta 2 Development Kit. Questi dispositivi si differenziano da quelli di realtà virtuale in quando è possibile comunque vedere il mondo reale. Meta porta l'augmented reality in campi d'interesse diversi dal solito, come la medicina e le aziende, con lo scopo di rendere più facile l'apprendimento, e più immediato l'aiuto in caso di bisogno per risolvere alcune situazioni [12]. Magic Leap[3], è un'altra azienda che sviluppa tecnologie di realtà aumentata, non sono ancora stati rilasciati dispositivi da questa azienda, ma si occupa principalmente dello sviluppo della tecnologia in generale, e a differenza dei dispositivi comuni di virtual e augmented reality, che pongono la concentrazione dell'utente sul mondo vicino a lui, Magic Leap punta ad allargare sempre di più la visione del mondo aumentato. Per quanto riguarda le piattaforme software per lo sviluppo di augmented reality ne esistono varie: merita una citazione Vuforia AR, che permette lo sviluppo di applicazioni di questo tipo per i dispositivi mobile. Le applicazioni di realtà aumentata realizzate con questa tecnologia, sono eseguibili su ogni dispositivo mobile, che esso sia di produzione Apple o che usi Android, e la tecnologia permette di aggiungere qualsiasi immagine o oggetto alla realtà ripresa da un dispositivo mobile. Vuforia, ha sviluppato anche un supporto per i nuovi dispositivi Microsoft, quali Surface e Hololens, come annunciato a giugno 2016[4]. Come accennato nel paragrafo precedente, anche Microsoft ha realizzato un dispositivo che utilizza questo tipo di tecnologia, attualmente è il dispositivo più completo sul mercato, in quanto ha una vasta quantità di sensori e rende l'esperienza totalmente immersiva al fine di integrare perfettamente il mondo reale con quello virtuale.

[3]Sito Ufficiale Magic Leap `https://www.magicleap.com`
[4]Developer Site Vuforia `https://developer.vuforia.com/`

Capitolo 2

Microsoft Hololens

In questo capitolo verrà analizzato il dispositivo Microsoft Hololens[1]: un dispositivo innovativo fra quelli presenti nel mercato, in grado di rappresentare i concetti di realtà aumentata e mixed reality. Usando sempre un approccio top-down, verrà quindi descritto in generale in cosa consiste questo dispositivo, fino ad arrivare nel dettaglio, prima in termini tecnici e poi in termini di concetti, per capire meglio cosa si nasconda dietro a questo dispositivo e come funziona veramente. Al momento della scrittura del documento, il dispositivo vero e proprio risulta disponibile solo in USA e Canada, al prezzo di 3000 dollari, in una versione solamente per sviluppatori. Il dispositivo deve ancora essere commercializzato a tutte le persone e non è ancora stata annunciata una data di rilascio per l'uso comune.

2.1 Cos'è Microsoft Hololens

Microsoft Hololens è un visore di mixed reality progettato e sviluppato da Microsoft. Il visore non è altro che un caschetto indossabile dotato di una sorta di smart glasses incorporati capaci di far visualizzare all'utente degli ologrammi nel mondo reale (vd. fig. 2.1). Il caschetto viene definito da Microsoft un "computer olografico", in quanto è stato progettato credendo nel fatto che "gli ologrammi siano il futuro nell'evoluzione dell'informatica". Il dispositivo dà la possibilità all'utente di essere libero con entrambe le mani, quindi è un sistema hands-free, ed è un vantaggio non da poco, rendendo Hololens un dispositivo indossabile sempre, per svolgere le azioni di tutti i giorni, dalla più semplice alla più complessa, e non un dispositivo usabile solo in certi momenti della propria giornata.

Figura 2.1: Il dispositivo Microsoft Hololens

Il dispositivo è indossabile da tutti, questo è reso possibile da un regolatore manuale dell'ampiezza degli occhiali, che permette di stringere o di allargare la circonferenza. Per quanto riguarda il peso del dispositivo, risulta distribuito omogeneamente e non maggiormente sulla parte frontale a differenza di come si possa pensare appena si vede il dispositivo. Questo rende Hololens più confortevole, in linea con il pensiero di indossarlo spesso. Il visore crea una sorta di nuova realtà, appunto la mixed reality, raggiungendo così un punto di mezzo, un'intersezione, di un livello quasi superiore, fra augmented reality e virtual reality. Come descritto in precedenza, la virtual reality proietta l'utente in un mondo parallelo, isolandolo dalla realtà, invece la realtà aumentata consente all'utente di vivere la propria realtà, ma avendo informazioni e contenuti digitali davanti alla propria visione reale. Microsoft Hololens fa qualcosa di più: crea oggetti digitali, chiamati appunto ologrammi, rendendoli parte del mondo fisico, quasi senza far accorgere all'utente della differenza fra reale e virtuale, facendo sembrare questi oggetti parte del mondo vero e proprio.

2.1.1 Dettagli Tecnici

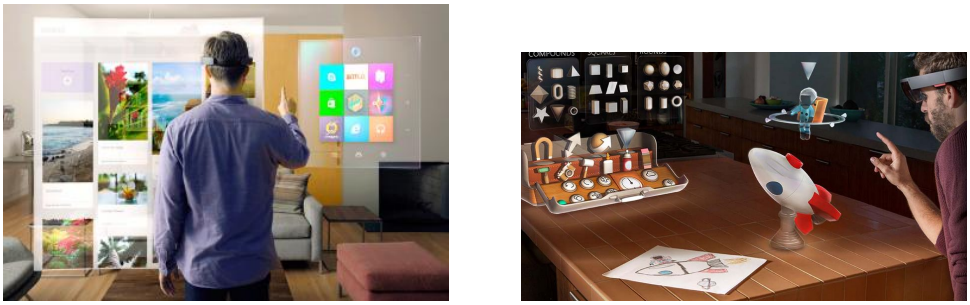
Analizzando i dettagli tecnici di Hololens, esso risulta un dispositivo veramente potente. Non ha bisogno nè di fili, nè di un computer sempre connesso, è quindi un dispositivo totalmente indipendente. Il dispositivo, attualmente, ha Windows 10 come OS, ed è dotato di una batteria di 16.5Wh, CPU e GPU, 2GB di RAM e 64GB di memoria, Wi-Fi 802.11 e Bluetooth per connettersi rispettivamente ad internet e con qualsiasi dispositivo, inoltre sfrutta anche Bluetooth Low Energy per collegarsi al clicker, un accessorio fornito da Microsoft per favorire lo scorrimento e la selezione nelle finestre dell'interfaccia. La vera potenza di Hololens sta nell'analisi del mondo che lo circonda, per svolgere questa funzione Microsoft ha posto nel dispositivo vari sensori, qua-

li accelerometro, giroscopio e magnetometro, racchiusi in una IMU (Inertial Measurement Unit), unitamente a quattro camere per capire il mondo attorno al dispositivo, e una camera di profondità, per ottenere una visione completa di cosa ci sia e cosa succeda attorno ad Hololens. E' stata creata una HPU, cioè una Holographic Processing Unit, al fine di combinare le informazioni raccolte dalle camere di profondità e i dati raccolti dall'IMU. L'Holographic Processing Unit è in grado di gestire anche le gesture di input effettuate dall'utente e gli input vocali. Alex Kipman, in qualità di Hololens Chief Inventor, ha dichiarato che "HPU è in grado di processare terabytes di informazioni" dai sensori di Hololens in tempo reale [2]. Questo dato lascia già immaginare la potenza di tale dispositivo. Oltre ai sensori, una parte cruciale del device sono le lenti. Hololens incorpora delle lenti avanzatissime, precisamente delle lenti olografiche see-through, ovvero delle lenti che permettono di guardare al di là delle stesse, non coprendo il mondo reale come succede in un dispositivo di realtà virtuale. Unitamente alle stesse viene utilizzato un doppio motore luce, ed una risoluzione olografica pari a 2.3M di punti luce totali. Le funzionalità dei sensori, insieme alle lenti ottiche super avanzate, permettono la visione di ologrammi che siano parte vera e propria del mondo. Nel dispositivo è presente una fotocamera da 2.4MegaPixel, per scattare foto in ogni momento, e una videocamera da 1.1MegaPixel, per riprendere la propria visione del mondo olografico, oltre agli altoparlanti per riprodurre i suoni come se appartenessero al mondo reale. Hololens è ricaricabile tramite porta microUSB, ed è utilizzabile anche durante il periodo di carica, infine è dichiarato che il peso del dispositivo sia 579 grammi [16].

2.2 Funzionamento Generale

Hololens funziona esattamente come un computer, semplice, solo che si tratta di un computer olografico. All'accensione, verrà reso visibile all'utente un menù principale, dal quale potrà lanciare la fotocamera o la videocamera, accedere al menu con tutte le app, e lanciarne una. Ogni applicazione progettata per Microsoft Hololens è basata su UWP. Universal Windows Platform è un modello di applicazione e un ambiente per le applicazioni Windows. Il modello di applicazioni UWP stabilisce come vengono trattate le applicazioni, dall'installazione all'aggiornamento fino alla disinstallazione, e questo si occupa del ciclo di vita delle applicazioni quanto vengono lanciate e terminate. Questo vuol dire che ogni applicazione basata sul modello UWP è in grado di essere eseguita su Hololens, quindi applicazioni UWP per desktop o mobile potranno essere scaricate dal Windows Store e lanciate sul dispositivo. Ci sono due categorie di applicazioni, una comprende le applicazioni 2D, cioè

quelle appena accennate, e l'altra le applicazioni olografiche. Per quanto riguarda le applicazioni 2D invece di essere lanciate in una finestra come in un computer, vengono lanciate, a livello grafico, su una superficie verticale che simula una finestra (vd. fig. 2.2a), è facile provare applicazioni non progettate per Hololens su questo dispositivo, infatti Microsoft ne dà un esempio palese installando già di base Skype all'interno del dispositivo. Per quanto riguarda le applicazioni olografiche, inizialmente si presentano come una finestra, per permettere all'utente di piazzarla nel mondo e lanciare il programma, una volta dato l'ok per l'esecuzione la finestra scompare e appare lo spazio olografico fuso completamente con il mondo reale (vd. fig. 2.2b). La differenza e la novità fra le semplici applicazioni e quelle olografiche è proprio in questo punto, le applicazioni olografiche non sono una semplice finestra piazzata nel mondo reale, come fosse un quadro, ma creano un vero e proprio spazio olografico per tutto l'ambiente circostante all'interno del quale l'applicazione svolgerà le sue azioni. Microsoft, come in tutti i suoi ultimi sistemi, aggiunge anche a Hololens, l'assistente per l'utente Cortana. In questo caso dato che sono molto frequenti i comandi vocali, ci si potrà rivolgere a Cortana chiedendo di aprire o chiudere un'applicazione o semplicemente chiedendo come fare qualcosa su questo dispositivo.



(a) App 2D (b) App olografica

Figura 2.2: Raffigurazione dei due tipi di applicazioni descritti.

2.2.1 Building Blocks: Moduli Principali Hololens

In questa sezione saranno descritti i moduli principali che caratterizzano il dispositivo Microsoft Hololens, con un approccio bottom-up. Partendo quindi dalla base, cioè dal sistema operativo, verranno descritte le parti fondamentali fino a raggiungere i principi di base che consentono l'esperienza innovativa su Hololens.

Sistema Operativo Microsoft Hololens usa Windows 10 come sistema operativo. Questa scelta implica che la gestione del sistema sia effettuata allo stesso modo di un altro device che ha come sistema operativo Windows 10. La grafica a blocchi caratteristica di Windows 10, come si può notare nel menù principale e in tutte le finestre utilizzabili con questo sistema operativo, è la stessa di un pc, o uno smartphone, con Windows 10. Essendo un sistema operativo conosciuto, è facilitato l'utilizzo da parte dell'utente medio, che sarà in grado così di sapersi già orientare tra le relative funzionalità messe a disposizione. E' stato già accennato che il sistema operativo di questo dispositivo, cioè Windows 10, utilizzi lo stesso modo di gestire le applicazioni di un altro dispositivo, che non siano gli occhiali olografici, che utilizza il medesimo sistema operativo. Le applicazioni sono gestite secondo lo schema Universal Windows Platform, cioè una piattaforma che indica per ogni applicazione come verrà gestito il suo ciclo di vita e la sua interazione con il sistema operativo. E' bene ricordare che questa scelta, consente l'utilizzo su Hololens di applicazioni progettate con UWP, rendendolo così un dispositivo in grado di eseguire anche altre applicazioni non necessariamente olografiche.

Windows Holographic Microsoft, per raggiungere lo scopo prefissato di creare una mondo misto fra realtà e virtuale, ha creato una piattaforma di mixed reality basata sul sistema operativo Windows 10. Questa piattaforma a livello concettuale si pone sopra il sistema operativo, in quanto si basa su di esso. La piattaforma in questione è chiamata Windows Holographic ed è la vera innovazione, dato che consente la creazione e la gestione di ogni aspetto olografico. Microsoft ha annunciato lo sviluppo ad inizio 2015 [2], e questa piattaforma è attualmente in continuo sviluppo, dato che al momento il progetto Hololens è in fase di testing avanzata, essendo disponibile solo una versione per sviluppatori. Windows Holographic è la piattaforma che permette la creazione di un mondo fatto di oggetti olografici e reali, quindi un mondo di mixed reality, e si occupa di tutto ciò che riguarda il mondo olografico, dalla gestione dei sensori che forniscono informazioni per una corretta esecuzione del sistema olografico, fino all'implementazione dei vari principi olografici che corrispondono all'utilizzo delle Windows Holographic API. Le applicazioni olografiche utilizzano tutte questa piattaforma per quanto riguarda ogni contenuto olografico.

Concetti Fondamentali I principali concetti fondamentali, per quanto riguarda un'applicazione olografica, si trovano al livello superiore dello stack, in quanto sono le parti che andranno implementate per realizzare un'applicazione di questo tipo. Riguardano la gestione delle posizioni nel mondo, cioè i sistemi di coordinate, la gestione dello spazio intorno a sè, spatial mapping, la gestio-

ne dell'input, gaze e gesture, e la gestione dei suoni, spatial sound. Questi concetti sono racchiusi nelle Windows Holographic API, le quali mettono a disposizione le nuove funzionalità per la gestione di questi concetti innovativi, sempre basati sulla piattaforma precedentemente spiegata. Ogni concetto verrà approfondito nel dettaglio nelle sezioni successive.

2.3 Principi di Base

La domanda principale, dettata dalla curiosità, rimane, come fa Hololens a fare tutto ciò. Gli ologrammi non sono altro che rappresentazioni tridimensionali fatte da luci e suoni, di qualsiasi cosa, come per esempio personaggi, finestre di programmi, o oggetti. Si capisce quindi che ogni cosa è rappresentata come un ologramma. Inizialmente il dispositivo acquisisce più informazioni possibili riguardo al mondo reale, capendo come è fatta la stanza attorno all'utente, se sono presenti o meno e in che posizione si trovano eventuali oggetti veri e propri. Le informazioni sul mondo sono captate in qualsiasi momento in modo continuativo dal dispositivo, facendo una vera e propria mappatura dello spazio. La mappatura continua in tempo reale è dettata fondamentalmente dal fatto che l'utente si possa e deve muoversi avendo gli occhiali addosso, questo implica che se gira la testa, o va in una stanza diversa, sarà in una posizione differente dalla sua precedente, e si troverà in un punto diverso dell'ambiente, quindi gli occhiali, per continuare a rendere l'esperienza reale, hanno bisogno di sapere dove si trova l'utente e cosa lo circonda in ogni momento, e ciò è reso possibile dalle camere d'ambiente e dai sensori presenti sul dispositivo. Il dispositivo sarà quindi in grado in ogni momento di avere dettagli sull'ambiente circostante, e ad ogni rilevazione aggiornerà i suoi dati. Avendo una visione del mondo, Hololens è in grado di piazzare gli ologrammi nel mondo stesso, questa azione è chiamata "pinning", che significa appunto fissare. Ogni volta che l'utente vorrà posizionare un ologramma nel mondo, la prima azione che gli viene richiesta è proprio questa, cioè indicare al dispositivo dove vuole piazzare il suo ologramma. L'ologramma potrà rimanere fisso in quella posizione, mentre l'utente si sposta da un'altra parte oppure potrà anche seguire l'utente durante i suoi spostamenti. L'utente sarà in grado di dare degli input al device facendo dei gesti con le mani oppure usando la voce, e il dispositivo è in grado di rilevarli. Questo permette di interagire con gli ologrammi svolgendo delle azioni su di essi, oppure di interagire con il dispositivo, chiedendogli per esempio di aprire il menù principale, oppure con il mondo stesso, per piazzare appunto gli ologrammi. E' questo il modo di interagire con il dispositivo, viene abbandonato così il concetto di mouse e di click su uno schermo, e saranno le mani a fare click o qualsiasi altra azione. Nonostante sia possibile utilizza-

re il clicker, Hololens è in grado di farne a meno per cliccare su un oggetto, ma può diventare utile se si vuole per esempio scorrere una pagina web. Per indicare al dispositivo in che punto si vogliono eseguire tali azioni sarà sempre visibile un puntatore, che indica dove verrà eseguita l'azione, il puntatore segue la direzione dalla parte frontale del dispositivo, quindi il cursore sarà mosso direttamente in relazione ai movimenti della testa dell'utente, per poi eseguire delle azioni con le mani. Questi concetti corrispondono al nome di gaze e gesture input, spiegati in seguito nei dettagli. L'utente sarà avvolto totalmente nell'esperienza di mixed reality anche grazie ai suoni emessi dagli altoparlanti del dispositivo, i quali sono veri e propri suoni spaziali, percepibili da ogni direzione, e associabili ad ogni oggetto olografico. Un ologramma è in grado di utilizzare anche le informazioni acquisite dal dispositivo sullo spazio, questo permette agli ologrammi di muoversi nel mondo e di diventare ancora più realistici. La descrizione dei principi di base è approfondibile e basata sulla documentazione ufficiale [15].

2.3.1 Sistemi di coordinate

Per tutte le funzioni che riguardano il posizionamento di un ologramma, il suo orientamento, oppure la sua identificazione, Hololens usa diversi sistemi di coordinate per rappresentarli. Non è una cosa che viene visualizzata all'utente, ma è solo un modo di Hololens di trattare gli oggetti e le azioni, per rapportarsi con il mondo in modo autentico, e rendere ancora di più, tutto questo, reale. A differenza delle normali applicazioni di grafica tridimensionale, che usano sistemi di coordinate cartesiane, Hololens usa anche dei sistemi di coordinate spaziali: questi sistemi non sono altro che sistemi di coordinate che hanno un significato realmente interpretabile e collegabile al mondo reale. Infatti ogni unità in un sistema di coordinate spaziale corrisponde ad un metro, e questo facilita le misure, la scalatura degli oggetti, essendo le unità le stesse del mondo reale, anche a livello di programmazione. I sistemi di coordinate cartesiani sono di due tipi: left-handed e right-ended, la differenza fra loro si trova nel posizionamento degli assi (vd. fig. 2.3).

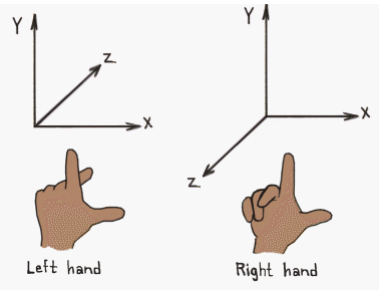


Figura 2.3: Rappresentazione dei sistemi left-handed e right-handed

Per quanto riguarda i sistemi di riferimento left-handed l'asse X va verso destra e l'asse Y verso l'alto, l'asse Z invece, in un sistema left-handed punta in avanti. Queste direzioni sono intese in riferimento ai punti positivi di ogni asse. In un sistema right-handed, l'asse X e Y sono identiche ad un sistema left-handed, quindi puntano rispettivamente verso destra e verso l'alto, invece l'asse Z punta all'indietro. Il sistema di coordinate spaziali utilizzato da Hololens è right-handed. L'applicazione Hololens sarà comunque in grado di trattare entrambi i sistemi di coordinate, nonostante prevalga quello spaziale. Ogni cosa in Hololens è riferita al sistema di coordinate stabilito, entrando nel dettaglio, durante la progettazione degli ologrammi si utilizza il sistema di coordinate spaziale, per farne usi diversi: piazzare l'ologramma in un punto fisso nel mondo, oppure piazzare l'ologramma ad una distanza fissa dall'utente, quindi rendendolo mobile e collegato alla posizione della persona.

Stationary Frame Of Reference

Stationary frame of reference è un punto del mondo sul quale si costruisce il sistema di coordinate di un'applicazione per Hololens. Questo punto generalmente viene calcolato in relazione alla posizione del dispositivo, ed è anche un punto di riferimento per lo stesso. Nel caso in cui si volesse piazzare un ologramma in un punto fisso nel mondo, andrà usato appunto uno stationary frame of reference, cioè un frame di riferimento fisso. Questo significa che l'ologramma avrà bisogno di conoscere un punto fisso nel mondo, in cui dovrà rimanere anche se il dispositivo si muove nell'ambiente. Viene calcolata una distanza da quel punto, espressa in un vettore contenente le tre dimensioni del sistema di coordinate, nella quale verrà piazzato l'ologramma, che generalmente si trova davanti all'utente o comunque nel suo campo visivo. L'applicazione, continuerà a scansionare l'ambiente, ed elaborando le informazioni ottenute, potrà aggiornarne la posizione appena riconosce dati più completi sull'ambiente, andando così a perfezionare la posizione calcolata per l'ologramma. L'aggiornamento dei dati relativi allo spazio potrebbe produrre

degli spostamenti rispetto alla posizione originale dell'ologramma, diventando così un problema, e Microsoft Hololens sfrutta le spatial anchor per risolvere questi errori.

Spatial Anchor Come appena accennato, le Spatial Anchor servono per eliminare i leggeri spostamenti dalla posizione originale dell'ologramma in seguito all'aggiornamento dei dati sull'ambiente circostante. L'ologramma può essere piazzato usando una spatial anchor e rimarrà fisso in quel punto, come se fosse ancorato. Lo Spatial Anchor viene creato partendo dallo Stationary Frame Of Reference, e nel punto che in precedenza era stato calcolato come distanza, viene creato un sistema di riferimento nuovo, che rimarrà fisso in quel punto. Dal momento in cui si posiziona un'ancora, non si potrà più spostarla. La differenza principale fra un sistema di riferimento basato sulle ancore e uno stationary frame of reference riguarda il metodo di aggiornamento della posizione. E' inevitabile ed è anzi naturale che il dispositivo aggiorni le posizioni dei propri oggetti, in quanto non conosce perfettamente l'ambiente al momento del posizionamento dell'ologramma, ma ne acquisisce informazioni in modo dinamico mano a mano che l'utente si muove nel mondo. Quando necessario, le ancore si aggiornano in relazione ad altre ancore utilizzate o in relazione ai sistemi di riferimento spaziali, questo garantisce che non venga nettamente modificata la posizione causando un effetto di drift, ma che l'ologramma rimanga in modo preciso nello stesso posto. Nello stationary frame of reference viene aggiornata la posizione relativamente a quella dell'utente, invece nelle spatial anchor viene aggiornata la posizione relativamente alla sua origine, ottenendo così risultati più precisi. E' quindi un metodo di posizionamento accurato per svolgere una funzione di questo tipo. Le ancore sono un sistema di posizionamento ottimo anche per altri due motivi. E' possibile salvare le ancore e quindi una certa posizione nel mondo, prima dello spegnimento del dispositivo, in modo da conservarne i dati e ripristinarli al momento dell'accensione, trovando l'ologramma nello stesso punto in cui l'avevamo lasciato. I dati relativi alle ancore vengono salvati in uno store ed identificati da una chiave univoca. E' possibile anche condividere fra più dispositivi le informazioni relative ad un'ancora. In questo modo i due dispositivi saranno in grado di conoscere le stesse informazioni relativamente a quella parte di ambiente, e nel caso visualizzassero lo stesso ologramma, sarà possibile vederlo nello stesso punto per entrambi.

Attached Frame Of Reference

Questa modalità viene utilizzata nel caso in cui non volessimo fissare l'ologramma in un punto stabile del mondo, ma nel caso in cui volessimo tenerlo

legato all'utente. L'ologramma sarà in grado di seguire l'utente ad ogni suo spostamento. Ci sono due tipi di aggancio, uno body-locked, e un altro head-locked. Un ologramma body-locked segue la posizione del dispositivo, rimanendo ad una distanza fissa dallo stesso. Un ologramma head-locked invece rimane ad una distanza fissa dall'utente, ma risulterà sempre visibile nel suo campo visivo, perchè legato alla posizione della testa. E' sconsigliato utilizzare ologrammi head-locked, in quanto rendono l'esperienza meno reale, dato che nella realtà non si avrà mai un oggetto fisso nel proprio campo visivo. Un ologramma body-locked può avere invece molteplici utilizzi: se ci trovassimo in un videogioco, e volessimo giocare in squadra con un ologramma, esso potrebbe seguirci in una missione durante tutti i nostri spostamenti. In questo sistema di riferimento l'origine è sempre posta nella posizione del dispositivo. Tale caratteristica, rende il sistema di riferimento l'unico utilizzabile nel caso in cui Hololens non sappia in che punto si trova del mondo, e quindi non sia più in grado di tenere le posizioni degli ologrammi.

Problemi di riconoscimento dello spazio

Ci sono alcuni casi in cui il dispositivo potrà fare fatica a riconoscere e tenere traccia dell'ambiente circostante. Nel caso in cui il dispositivo non sia più in grado di capire in che punto si trova, è ovvio che non sarà più in grado di aggiornare le distanze relative alle ancore o agli stationary frame of reference. Sarà solamente in grado di utilizzare l'attached frame of reference come detto in precedenza, ed è buona norma visualizzare all'utente, nel caso ce ne fosse bisogno, un messaggio o dei contenuti informativi su come ovviare a questo problema. I contenuti mostrati in queste circostanze dovranno essere posizionati usando l'attached frame of reference. Un altro caso in cui potrebbe essere necessario mostrare all'utente dei messaggi d'errore, potrebbe verificarsi nel caso in cui ci si trovi in due spazi uguali fra loro, non è insolito di trovarsi in due stanze identiche all'interno di un albergo, o anche nella propria casa. Il dispositivo rilevando lo stesso ambiente attorno a lui, potrà confonderlo, e quindi capire che sia lo stesso spazio già conosciuto e non uno nuovo ancora da scoprire. Nel caso in cui riconoscesse l'ambiente come quello già conosciuto in precedenza potrebbe verificarsi la comparsa degli ologrammi che erano presenti nell'ambiente precedente. Nel caso succedesse ciò, è possibile eliminare i dati acquisiti su quello spazio, per far si che il dispositivo rinizi il riconoscimento della stanza, però eliminando i dati su uno spazio viene eliminato anche tutto ciò che esso conteneva, quindi anche degli ologrammi. Allo stesso modo se l'utente torna in un ambiente conosciuto, ma che ha subito dei cambiamenti significativi, gli ologrammi potrebbero posizionarsi in punti diversi, oppure se si trova in un ambiente affollato, il dispositivo farà fatica a tracciare l'ambiente

in quanto ci sono dei cambiamenti enormi in brevissimi lassi di tempo, è quindi consigliato utilizzare il dispositivo in ambienti non affollati.

2.3.2 Gaze

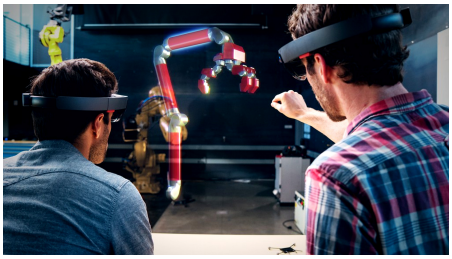
Il gaze è un punto cruciale e fondamentale di ciò che stiamo analizzando. E' la prima forma di input ed è usato per molteplici funzioni con le applicazioni olografiche. Il gaze indica la direzione in cui l'utente sta guardando mentre vuole fare qualcosa, analogamente alla realtà in cui quando noi vogliamo interagire con qualcosa generalmente lo guardiamo. A differenza della realtà però, la funzione di gaze si basa sulla posizione della testa e non sulla direzione degli occhi, quindi in base alla posizione e ai movimenti della testa rilevati dal caschetto, cambierà la direzione identificata al momento dal gaze. E' come se fosse una sorta di raggio laser, che parte dalla propria testa e va verso l'oggetto che si sta guardando, questo concetto riprende quello di cursore, in modo più completo, ed è proprio questo uno dei principali utilizzi del gaze. E' utilizzato nella totalità delle applicazioni in quanto elemento fondamentale, anzi, è un elemento fondamentale delle stesse, perchè, come sarà chiarito in seguito, da esso dipendono tutti i gesture input effettuati all'interno di un'applicazione. Per identificare dove l'utente stia guardando precisamente, viene effettuato un raycasting da parte dell'applicazione, praticamente si traccia una linea immaginaria nella direzione specificata dalla testa della persona, esattamente come un raggio laser, andando a intersecarla con gli oggetti lungo quella linea, e il cursore si fermerà quando incontra l'oggetto più vicino in quella direzione. Generalmente, le applicazioni che ne fanno uso, mostrano il cursore in modo fisso e ben visibile, così l'utente è in grado di riconoscere in ogni momento dove esso si trova e sarà in grado di riconoscere in modo chiaro l'oggetto identificato dal gaze sul quale verranno effettuate le azioni. Allo stesso modo in cui l'applicazione è in grado di sapere quando l'utente guarda un oggetto, essa è in grado di sapere anche quando non lo sta guardando, e nel caso fosse necessario mandare un segnale audio. Il gaze è abbinato, oltre che alle gesture di input, anche ai comandi vocali, essendo la forma primaria e fondamentale di input.

2.3.3 Gesture Input

La modalità innovativa di input in Microsoft Hololens, è l'utilizzo delle gesture. Una gesture non è altro che un movimento della mano, il quale specifica una precisa azione da eseguire. Questo movimento della mano viene riconosciuto dai sensori del dispositivo e poi elaborato per eseguire l'azione. E' una modalità di input molto comoda per l'utente in quando lo rende libero da qualsiasi forma di telecomando o pad, interagendo direttamente con il

mondo. E' fondamentale per il dispositivo riconoscere le mani dell'utente, a supporto di questo, i movimenti vanno eseguiti all'interno del gesture frame, cioè nella porzione di spazio attorno al dispositivo predisposta a rilevare le gesture, la zona di riconoscimento è davanti a se stessi e a fianco, sia a destra che a sinistra. Hololens riconosce due movimenti precisi delle mani, e quando le mani non compiono questi due movimenti, ma qualsiasi altro gesto, non viene riconosciuto, anche se è all'interno del gesture frame. Il primo movimento, è chiamato "ready state" ed è formato dalla mano chiusa a pugno rivolta in avanti, con l'indice alzato (vd. fig. 2.4a). Il secondo movimento, definito "pressed state", è dato dalla mano chiusa a pugno, con l'indice abbassato in avanti (vd. fig. 2.4b). La combinazione di questi due movimenti permette di svolgere varie azioni.

(a) Ready State



(b) Pressed State

Figura 2.4: I due tipi di movimento riconosciuti da Hololens.

Press and Release Gesture La gesture più comune è quella di select, che va a sostituire il click del mouse, per fare ciò, è necessario partire da uno stato di ready della mano, abbassare l'indice come per cliccare, passando così in pressed state, e ritornare subito in ready state, come se si fosse rilasciato il bottone del mouse. La gesture di selezione, è definita da Microsoft "press e release" e solitamente si associa ad azioni di selezione di un ologramma, ed è combinata ovviamente con le informazioni del gaze, per capire dove si sta guardando in quel momento e su quale ologramma o in che punto eseguire l'azione di press e release.

Hold Gesture Questa gesture è molto simile a quella di press e release, in quanto si parte sempre da uno stato di ready della mano, per poi abbassare l'indice in pressed state, mantenendo però questo stato per un periodo più lungo, e concludendo l'azione tornando in ready state. L'azione può essere usata per far svolgere un comportamento specifico all'ologramma nel caso venga mantenuto lo stato di pressed, che equivale al tener premuto il click del mouse.

Manipulation Gesture La gesture di manipolazione è generalmente utilizzata nelle applicazioni per spostare un oggetto, ridimensionarlo o ruotarlo. Anche in questo caso, è necessario partire dalla direzione del gaze, per sapere su quale ologramma eseguire l'azione. Appurato ciò, si esegue una press da un ready state, per poi compiere dei movimenti con la mano lungo le tre dimensioni del mondo. La manipolazione dell'ologramma viene gestita dai movimenti della mano, e ci si basa solo su tali movimenti dal momento in cui si inizia la gesture, l'utente potrà quindi anche guardare altrove mentre esegue questi gesti.

Navigation Gesture La navigazione è intuibile già dal nome della gesture, può essere usata nelle applicazioni per scorrere un menù o eseguire azioni di scorrimento, come se si fosse su un'interfaccia a due dimensioni.

Bloom Un gesto che non dipende da questi due movimenti della mano è quello di "bloom", ed è un gesto di sistema (vd. fig. 2.5). Fare questa gesture è esattamente come premere il tasto Windows sulla tastiera, e come si può ora intuire, il suo effetto è quello di aprire il menù di avvio del sistema. La gesture bloom è riconosciuta ponendo la mano con il palmo in su, e le punte delle dita che si toccano insieme, come se fosse un fiore chiuso, per poi aprire la mano, esattamente come si aprirebbe un fiore. Si può tornare al menù principale chiedendolo anche a Cortana, non solo utilizzando questa gesture. La gesture di bloom è una gesture di sistema, quindi non è utilizzabile nelle applicazioni perchè è gestita dal sistema stesso e non dalle specifiche applicazioni.

Figura 2.5: Bloom Gesture in Microsoft Hololens

Clicker Microsoft ha introdotto un dispositivo chiamato Clicker, che permette di cliccare e di selezionare un oggetto, oppure di scorrere una pagina web, o un menù. E' un accessorio che verrà fornito insieme al dispositivo, presente nella scatola. Quando si compiono delle gesture con il clicker non è necessario avere il pulsante all'interno del frame gesture, ma si può tenere in qualsiasi posizione.

2.3.4 Vocal Input

Gli input vocali sono una delle forme di input presenti su Hololens. E' una modalità molto pratica, potente ed efficace, dato che per l'utente, parlare per scegliere cosa fare, essere ascoltato e compreso totalmente dal dispositivo, è una cosa quasi magica. I comandi vocali su Hololens sono trattati dallo stesso engine che supporta il riconoscimento vocale in altre applicazioni di tipo UWP. Con questa modalità di input, l'utente non compierà nessuna gesture, e nel caso voglia interagire con un ologramma, gli basterà guardarlo, sfruttando il puntatore del gaze, e pronunciare l'azione che vuole eseguire, a patto che sia preimpostata in modo che l'ologramma sappia cosa fare. L'effetto è il medesimo di una gesture, infatti pronunciando la parola "SELECT" mentre si fissa un ologramma, si seleziona allo stesso modo con cui si seleziona compiendo una gesture di "press e release". I comandi vocali sono utili per interagire con Cortana, basta richiamarla dicendo "Hey Cortana", e l'assistente di Windows si renderà disponibile per chiedergli quasi qualsiasi cosa, dal riavvio del dispositivo, allo scatto di una foto o di un video, fino a chiedergli che ora sono, oppure lanciare un'applicazione, o chiamare un contatto su Skype. Come accennato in precedenza, è possibile per esempio tornare al menù di avvio dicendo la frase "Hey Cortana, Go Home", e si aprirà il menù principale di Windows. I comandi vocali funzionano in ogni momento, sia quando non c'è nessuna applicazione in esecuzione, sia su applicazioni olografiche e applicazioni bidimensionali.

2.3.5 Spatial Mapping

Come accennato in precedenza, Hololens vuole fondere il mondo virtuale con quello reale, per far sembrare che siano una cosa sola, questo per far si che l'esperienza con Hololens diventi il più autentica possibile. Questo è l'obiettivo che si pone lo Spatial Mapping, e questa funzione viene svolta egregiamente dal dispositivo, mappando totalmente l'ambiente circostante. Per fare ciò inizialmente vengono definite a livello di applicazione le zone dello spazio delle quali l'applicazione vuole ricevere gli aggiornamenti. Vengono così generati dei dati cartografici inviati all'applicazione. Queste zone dello spazio sono chiamate volumi: un volume non è altro che una forma geometrica, come un cubo o una

sfera, che delimita una regione di spazio. Per ogni volume definito, verranno fornite all'applicazione una serie di superfici spaziali, chiamate spatial surface. Ogni spatial surface rappresenta una superficie del mondo reale, limitatamente ad un piccolo volume di spazio, ed è rappresentata da una fitta rete di triangoli uniti tra loro come a formare una rete sulla superficie. Questi dati serviranno all'applicazione per delimitare gli spazi e rendersi conto di cosa sia presente nel mondo circostante. Ogni applicazione che usa Spatial Mapping, appena riceve dei dati sull'ambiente o appena gli vengono comunicati dei cambiamenti nell'ambiente stesso, aggiorna le proprie spatial surface, scomparendo momentaneamente, per aggiornarsi e ricomparire subito dopo. E' possibile verificare all'avvio, che la prima cosa che fa il dispositivo dopo aver lanciato il sistema operativo, è proprio quella di provare a mappare il mondo circostante, per conoscere già l'ambiente appena Hololens viene messo in funzione. Ovviamente questi dati inizialmente saranno generici dato che l'utente non ha ancora iniziato a muoversi nell'ambiente, ed è per questo che c'è bisogno di aggiornare i dati in modo costante, al fine di avere una visione sempre più chiara del mondo. Nella figura 2.6 una rappresentazione della mappatura dello spazio da parte del dispositivo.

Figura 2.6: Mappatura dell'ambiente circostante al dispositivo

Usi Principali di Spatial Mapping

Sono molteplici gli usi che un'applicazione può fare con i dati sullo spazio circostante: può usarli per facilitare il posizionamento di un ologramma o nel caso di ologrammi dinamici, per farli muovere in relazione alle superfici presenti nello spazio. Nei paragrafi successivi verranno approfonditi questi aspetti [15].

Ostruzione L'utente si aspetta la giusta interazione fra gli ologrammi e l'ambiente reale, infatti se un ologramma si avvicina ad una parete, ci si aspetta che venga ostruito dato che c'è un muro, e non che l'ologramma ci passi attraverso senza problemi, altrimenti l'utente non avrà neanche il dubbio che l'ologramma sia una cosa vera o meno, ne consegue che uno degli usi principali di spatial mapping sia l'ostruzione degli ologrammi. L'utilizzo dei dati relativi allo spatial mapping per raggiungere questo scopo aumenta la percezione che un ologramma sia effettivamente una cosa reale e faccia parte dello stesso ambiente dell'utente. L'uso di spatial mapping per l'ostruzione degli ologrammi, fornisce anche dei feedback all'utente relativamente ad azioni da compiere: infatti quando un ologramma risulterà essere ostruito da una superficie reale, questo, oltre a dargli la conferma della sua posizione nel mondo reale, gli mostrerà che è bloccato e non può andare oltre in quanto è presente una superficie che lo ostacola, l'utente così sarà in grado eventualmente di agire in modo diverso sull'ologramma. Se fosse necessario che l'utente debba interagire con gli ologrammi anche se si trovano dietro una superficie (e questi non la oltrepassano dato che viene usato il concetto di ostruzione) è consigliato visualizzare l'ologramma in modo diverso, per esempio abbassandone la sua luminosità, in modo che l'utente sia in grado di visualizzare l'ologramma ma rendendosi conto allo stesso tempo che questo si trova dietro una superficie del mondo reale.

Visualizzazione I dati relativi alla mappatura dello spazio, non vengono visualizzati generalmente durante l'esecuzione di un'applicazione al fine di lasciare la visuale pulita e il più reale possibile, non sarebbe così se il mondo fosse totalmente coperto dalla rete di triangoli creata dal dispositivo. Ci possono essere alcuni casi in cui la visualizzazione delle superfici sia necessaria nonostante le superfici reali a cui si riferiscono siano già visibili. Supponendo che l'utente voglia posizionare un ologramma sulla parete, gli potrà essere utile vedere una sorta di ombra dietro l'ologramma per rendersi conto della vicinanza dell'ologramma alla parete, in modo da capirne la distanza fra l'oggetto ed eventualmente avvicinarlo ancora di più. Oppure se si sta usando un'applicazione che usa per esempio un tavolo sul quale l'utente deve interagire, può

essere utile mostrargli la superficie orizzontale del tavolo, per delimitargli la zona in cui agire.

Posizionamento I dati raccolti per lo spatial mapping, all'interno dell'applicazione, possono anche rendere più semplice e realistica la collocazione degli ologrammi nel mondo. Quando fissiamo un ologramma in un punto del mondo reale, indichiamo un punto nel mondo a tre dimensioni, invece è possibile definire la posizione di un ologramma relativamente ad una superficie, questo genera una correlazione tra un punto nello spazio tridimensionale ed un punto su una superficie bidimensionale. La riduzione della quantità di informazioni che l'utente deve dare all'applicazione, unitamente al tempo, alla facilità e alla precisione di interazione dell'utente con il mondo, sono tutti vantaggi di questa correlazione. Questo concetto viene usato ed applicato in quanto è un comportamento comune anche nella realtà, infatti quando puntiamo il dito per indicare dove mettere un oggetto, stiamo indicando una direzione e non una distanza. L'applicazione, seguendo questo principio reale, esegue un raycast lungo il gaze dell'utente, il quale come spiegato prima conosce la direzione verso la quale si sta puntando, per trovare la superficie più vicina, ricavando così la distanza dalla direzione specificata. Generalmente la procedura di raycasting viene eseguita più volte di seguito, questo perchè facendo più di un raycast lungo il gaze, e combinando i risultati, si ottengono dei risultati più precisi, dettagliati e ragionevolmente sicuri.

Simulazione della fisica Un altro uso dei dati dello spatial mapping riguarda la simulazione dei comportamenti degli ologrammi secondo i principi della fisica. Alcuni esempi sono il rotolamento di una palla su una superficie, o il rimbalzo sul muro o sul pavimento della stessa, unitamente alla forza di gravità nel caso si stesse raffigurando un oggetto che cade a terra. Nel caso si creino tipi di ologrammi che nella realtà avrebbero comportamenti che usano le leggi della fisica, sarebbe bene implementarli, andando così a rendere ancora più realistica l'esperienza, senza lasciare nulla al caso. Per fare ciò, a livello di sviluppatore, può essere necessario eseguire un'elaborazione della rete di triangoli relativa alla superficie, cioè per esempio rendere lisce delle superfici ruvide o riempire dei buchi nella rete di triangoli, questo per avere risultati più concreti e reali. E' importante considerare la grandezza del volume di scansionamento del mondo, infatti se una superficie non è stata scansionata e quindi mancano dei dati, quella parte del mondo non sarà conosciuta dall'ologramma ed esso non si scontrerà con quella superficie, ma penserà che non ci sia nulla. Tenendo conto di questi esempi e tali principi, se utilizzati quando necessario, aumentano la credibilità che tutto ciò che viene visualizzato sia reale.

Navigazione Come ultimo utilizzo, ma non per importanza, le mappe possono essere anche utili allo scopo di permettere all'applicazione di creare ologrammi che rappresentano dei personaggi con la capacità di muoversi liberamente nel mondo reale allo stesso modo in cui si muoverebbe una persona vera. La cosa importante per effettuare questo comportamento è identificare e delimitare bene le superfici in cui è possibile, per il personaggio, camminare o muoversi, al fine di non permettere all'ologramma di camminare su un tavolo o sul frigorifero, ma di camminare solo su superfici percorribili realmente, sennò non sarebbe più reale l'esperienza vissuta.

2.3.6 Spatial Sound

Gli spatial sound servono per rendere completa l'esperienza con Hololens, questo concetto, molto diffuso in mixed reality, vale per la maggior parte degli ologrammi. Se rappresentiamo un personaggio di un videogioco sotto forma di ologramma, l'utente non lo considererà mai veramente reale se non emette suoni e se non provoca rumori nei suoi movimenti, per questo, nel caso di vari ologrammi i suoni spaziali sono considerati parte fondamentale di un'esperienza realistica. Dato che gli ologrammi sono solamente oggetti fatti di luce, l'aggiunta di suoni aiuta a renderli più credibili al fine di creare un'esperienza olografica coinvolgente. Su Microsoft Hololens il motore audio si occupa della parte relativa ai suoni nell'esperienza della mixed-reality simulando il suono tridimensionale combinando fattori come direzione e distanza. Queste componenti fanno si che i suoni sembra che escano dagli oggetti fisici del mondo reale o dagli ologrammi, indistintamente. Dato che gli ologrammi sono visibili solo all'interno del campo visivo dell'utente attraverso la lente see-through del visore e dato che non è possibile vedere un ologramma se si trova dietro di noi, può essere utile, se non necessario, utilizzare degli spatial sound per richiamare l'attenzione sull'ologramma che non è visibile, indicando per esempio all'utente che deve girarsi e guardare l'ologramma, oppure fare qualche azione su di esso. Nel caso si utilizzino ologrammi dinamici, come durante un videogioco, si usano gli spatial sound per simulare l'avvicinamento di un ologramma verso l'utente, magari aumentandone il volume, oppure in caso contrario diminuendolo se l'ologramma si allontana. I suoni possono provenire da qualsiasi direzione, da sopra, da sotto, dietro, di lato rispetto all'utente. Questa funzione rende veramente coinvolgente l'esperienza, perchè rende gli ologrammi realistici, coinvolgendo pienamente l'utente, e dà la possibilità agli sviluppatori di creare qualcosa di veramente autentico. Oltre agli usi appena descritti, puramente scenici, vengono raccomandati da Microsoft agli sviluppatori altri utilizzi degli spatial sound, per esempio quando la mano dell'utente entra ed esce nel gesture frame, un suono potrà indicare che il dispositivo ha rilevato

che si sta per compiere una gesture, dando una certezza all'utente, oppure quando l'utente seleziona un ologramma un suono può confermare l'avvenuta ricezione del comando. Infine vengono caldamente raccomandati dei suoni per l'immersione dell'utente nell'ambiente circostante. Alcuni esempi di suoni ambientali possono essere il movimento dell'acqua nel caso l'utente stia agendo su una scena olografica che la contiene, o qualunque altro suono che provenga dall'ambiente, ed è molto utile aggiungerli per perseguire lo scopo di realismo, in quanto anche nella realtà ci sono sempre dei suoni di sottofondo non necessariamente prodotti da un oggetto. Fondamentale per gli sviluppatori è ricordare che se i suoni spaziali insieme a quelli ambientali stereo sono utili a creare un ambiente realistico, è bene tenere i suoni stereo ad un volume più basso rispetto a quelli spaziali, o comunque tenerli in secondo piano rispetto agli spatial sound, questo per lasciare spazio ai suoni spaziali di cui il volume può e deve essere tenuto più alto, altrimenti risulterebbe difficile per l'utente percepire uno spatial sound da una certa direzione se si trova in un ambiente molto rumoroso con suoni stereo generati dal dispositivo. Per simulare la distanza e la posizione dalla quale è percepito il suono, Hololens cerca di comportarsi esattamente come il cervello umano fa quando giunge un suono all'orecchio. Molto sinteticamente, il nostro cervello stabilisce la distanza e la direzione dalla quale proviene il suono, analizzando come i suoni raggiungono entrambe le orecchie, in questo caso il motore audio usa degli HRTF (Head Related Transfer Functions) per simulare i suoni e renderli spaziali. HRTF simula questa funzione modellando una risposta spaziale che caratterizza il modo in cui l'orecchio riceve i suoni da un punto lontano nello spazio.

Capitolo 3

Creare Applicazioni Microsoft Hololens

Dopo aver analizzato ogni elemento di Microsoft Hololens, in questo capitolo verrà spiegato come si crea un'applicazione olografica o bidimensionale, dagli strumenti necessari per svilupparla, fino a quelli per testarla, entrando nel dettaglio di ogni aspetto. La descrizione verrà effettuata in base alle prove effettuate e tenendo conto dello stato dell'arte al momento della scrittura.

3.1 Cos'è un'applicazione per Hololens

Un'applicazione per Hololens è una rappresentazione olografica dei contenuti, e sfrutta tutti i principi di base analizzati nel capitolo precedente, quindi si occuperà della gestione dello spazio, combinando le informazioni per svolgere le funzioni previste, si occuperà di quali azioni svolgere quando verrà riconosciuta una gesture, e si occuperà di gestire il gaze, tenendo traccia della sua posizione e direzione.
E' importante tenere ben distinto che tipo di applicazione si vuole creare in quanto cambiano totalmente sia l'approccio sia il modo di sviluppare l'app. Nel caso si voglia sviluppare un'applicazione bidimensionale, vuol dire che si sviluppa un'applicazione universale windows, che può essere eseguita oltre che su hololens, anche su un computer, o su uno smartphone. Per fare applicazioni bidimensionali bastano i comuni tools di sviluppo per applicazioni Windows, e il prodotto può essere testato anche su un semplice computer, questo è possibile dato che Hololens è in grado di eseguire anche applicazioni normali oltre a quelle olografiche. A prodotto ultimato si può lanciare l'applicazione sul dispositivo la quale funzionerà perfettamente come su qualsiasi altro dispositivo che utilizza Windows.
Per sviluppare applicazioni olografiche, lo sviluppatore dovrà utilizzare degli

strumenti appositi per sfruttare ogni funzione delle Windows Holographic API. E' consigliato usare Unity come strumento di sviluppo, nel caso si voglia partire dall'architettura già predisposta dell'applicazione e creare la propria scena e le proprie funzionalità. Per gli sviluppatori che vogliono creare il proprio motore centrale dell'applicazione, è meglio scendere ad un livello più specifico e quindi utilizzare Visual Studio, usando DirectX insieme alle Windows API, per implementare direttamente il modo in cui è progettata l'applicazione e come verrà gestito ogni dato rilevato dal dispositivo. Indipendentemente dallo strumento utilizzato per la realizzazione, per testarla, si può trasferire direttamente sul dispositivo, oppure lanciarla sull'emulatore, passando per Visual Studio.

3.2 Strumenti di Utilizzo

Dato che è necessario per testare l'applicazione, indipendentemente dal fatto che si usi Unity o DirectX, la prima cosa che uno sviluppatore dovrà installare è Visual Studio, al momento si utilizza Visual Studio 2015, Update 2, scaricabile dal sito ufficiale: `https://developer.microsoft.com/en-us/windows/downloads`.
Siccome attualmente il dispositivo è disponibile solo in USA e Canada, diventa fondamentale installare anche l'emulatore. Questo permetterà di testare comunque le applicazioni, che gireranno su una virtual machine senza avere Hololens a portata di mano. La build più recente disponibile è la 10.0.14342.1018, e si basa su Hyper-V, quindi tale tecnologia deve essere supportata dal proprio sistema. E' importante tenersi aggiornati sulle versioni rilasciate, essendo lo strumento ancora in fase di sviluppo. La stessa cosa vale per Unity, del quale è creata una versione specifica, denominata Unity Hololens Technical Preview, e non è altro che una versione personalizzata dello Unity Editor, scaricabile al link: `http://unity3d.com/pages/windows/hololens`.

3.2.1 Unity

Unity è sicuramente il modo più veloce per creare un'app Hololens, ma prima di sviluppare applicazioni olografiche su Unity, è importante avere delle conoscenze di base su questo ambiente di sviluppo. L'applicazione richiede che sia presente una camera la quale rappresenta la visione dell'utente, e un game object, il quale rappresenta l'insieme degli oggetti e dei loro comportamenti. Ogni cosa utilizzata all'interno della scena andrà aggiunta al game object. E' possibile inserire degli oggetti, che rappresentano gli ologrammi, e Unity mette a disposizione uno store in cui è possibile scaricare qualsiasi tipo di oggetto per

aggiungerlo alla propria scena. I comportamenti e le funzioni sono specificate tramite gli script, che possono essere aggiunti direttamente al game object, oppure agganciati all'ologramma. Per ogni ologramma è possibile specificare ogni funzionalità e proprietà, quale la zona in cui si può cliccare, oppure i suoni che esso emette. Unity è quindi uno strumento completo per sviluppare applicazioni olografiche senza occuparsi del motore dell'applicazione.

3.2.2 DirectX e Windows Holographic API

Creando l'applicazione direttamente su Visual Studio, si usano in modo diretto le API fornite da Microsoft, le quali rispondono al nome di Windows Holographic API e sono utilizzabili con DirectX 11. DirectX 12 non è attualmente supportato. Nello specifico viene usato HolographicSpace per creare la scena olografica che rende l'utente immerso nel mondo e vengono implementati i principi di base di Hololens in DirectX, riguardanti il rendering della scena olografica, il gaze, le gesture, lo spatial mapping, e gli input vocali. Le applicazioni possono essere scritte sia in C++, che in C#, e nel caso si usasse C# come linguaggio è possibile sfruttare una libreria open source per la gestione di alcune parti di grafica tridimensionale, di input e di suoni, chiamata SharpDX. Per creare un'applicazione, è possibile aprire la versione di Visual Studio indicata per lo sviluppo, selezionando il tipo di applicazione come olografica, unitamente al linguaggio in cui si vuole scrivere l'app.
La struttura dell'applicazione è un semplice game loop, nel quale vengono eseguiti tutti gli aggiornamenti della scena olografica. C'è un metodo nel main delle applicazioni, chiamato Update, in cui viene aggiornata la scena e gli ologrammi, il quale restituisce un HolographicFrame, che verrà usato per aggiornare la vista dell'utente. L'aggiornamento della vista è svolto dal metodo Render, tenendo ovviamente conto della posizione dell'utente, in modo da visualizzare solo gli ologrammi presenti nel campo visivo verso il quale è focalizzato. Il ciclo si ripeterà in modo continuo per tenere costantemente aggiornato lo stato e la vista dell'app.

3.2.3 Hololens Emulator

E' doverosa una piccola overview per quanto riguarda lo strumento di testing utilizzato. L'emulatore del dispositivo Microsoft Hololens è uno strumento utile e pieno di funzionalità per testare in modo corretto le applicazioni olografiche. Non c'è bisogno di modificare il codice per eseguire l'app o sul dispositivo o sull'emulatore, perchè quest'ultimo è in grado di eseguirla perfettamente. La scena rappresentata simula in modo completo quella reale, è possibile usare delle stanze simulate per testare l'applicazione in vari ambienti,

oppure modificare l'altezza della persona tramite i tasti PagSu/PagGiu, per avere una diversa prospettiva di visione all'interno dello spazio olografico. Ci si muove all'interno della stanza usando le frecce direzionali da tastiera, cioè i tasti W, A, S, D, o in alternativa è perfino possibile usare un controller XBox. Per quanto riguarda lo spostamento del gaze, cioè il movimento della testa, se ne occupa il click sinistro del mouse, il quale tenendo premuto e trascinando in qualsiasi direzione simula lo spostamento. E' possibile effettuare un'air tap gesture cliccando con il tasto destro del mouse, oppure tornare al menu principale, cioè eseguire una bloom gesture, premendo il tasto Windows sulla tastiera. Nell'emulatore è presente anche un pannello di opzioni in cui si può cambiare la stanza, o verificare i parametri relativi alla simulazione, quali le coordinate in cui ci si trova in quel momento e la posizione della testa.

3.3 Funzionamento Generale in DirectX

Un'applicazione olografica sviluppata usando DirectX e le Windows Holographic API, permette di occuparsi di ogni aspetto relativo ai concetti olografici. In linea generale la prima cosa che un'applicazione di questo tipo fa è creare e definire lo spazio olografico utilizzato, andando poi a rappresentare i contenuti olografici all'interno dello stesso. Viene gestita la posizione, l'aggiornamento, e la renderizzazione di ogni ologramma, oltre alla gestione degli input prodotti dall'utente. Saranno analizzati ora i principali aspetti che caratterizzano la struttura dell'applicazione a livello architetturale.

3.3.1 Struttura Architetturale

Quando viene lanciata l'applicazione sul dispositivo, la prime cose che vengono effettuate sono la creazione di un IFrameworkView, cioè un interfaccia che rappresenta la vista dell'applicazione, e il lancio di CoreApplication. Quest'ultimo si occupa della gestione dell'applicazione, come creare una nuova vista oppure uscire dalla stessa. Queste funzioni di base, vengono scritte nel file Program, e di seguito è riportato il frammento di codice necessario.

```
1 internal class Program
2     {
3         /// <summary>
4         /// Punto d'accesso dell'applicazione.
5         /// </summary>
6         [MTAThread]
7         private static void Main()
```

```
8       {
9           var exclusiveViewApplicationSource = new AppViewSource();
10          CoreApplication.Run(exclusiveViewApplicationSource);
11      }
12  }
```

Da questo momento, dopo essere passati per AppViewSource e AppView, le quali si occupano del collegamento tra l'app e Windows da parte del framework, verrà utilizzato il file main principale dell'applicazione, il quale implementa ogni funzionalità della stessa.

Gestione Spazio Successivamente viene creato lo spazio olografico, uno per ogni applicazione, usando HolographicSpace, che è il "portale all'interno del mondo olografico" come spiega Microsoft. HolographicSpace è ciò che permette di passare dalla visione di un finestra nella user interface, ad una visione full screen dell'applicazione, integrandola con il mondo. Per quanto riguarda la creazione dello spazio olografico, è la prima funzione da implementare per un'applicazione olografica. Essendo il portale all'interno del mondo olografico, è questo il momento in cui si specificano tutte le caratteristiche dello spazio olografico, inizializzando ogni elemento che lo caratterizza. Bisognerà inizializzare tassativamente i seguenti elementi:

- Ologrammi da rappresentare
- Gestore degli input
- Controllore della posizione del dispositivo.
- Creare il sistema di riferimento delle coordinate del mondo

Dal momento in cui è stato definito lo spazio olografico dell'applicazione, l'applicazione è in grado di eseguire le proprie funzionalità e di avere una visione full screen. Questo è possibile grazie all'Holographic Frame che verrà ricavato successivamente da HolographicSpace. HolographicFrame non è altro che una rappresentazione del contenuto olografico, che dovrà essere visualizzata dalle camere. Si usa quindi un approccio dal generale al particolare, in quanto da HolographicSpace si crea una visione del mondo chiamata HolographicFrame, la quale verrà visualizzata all'utente. E' per questo fondamentale creare e settare un HolographicSpace appena viene creata l'applicazione.

Ciclo di controllo Per quanto riguarda il ciclo di controllo di un'applicazione olografica, questo non è altro che un loop. L'applicazione ha bisogno di controllare ogni aspetto ad intervalli di tempo brevissimi, ed eventualmente

modificare i contenuti. Questa necessità implica il bisogno di creare un loop, che rappresenta il ciclo principale di controllo dell'applicazione, all'interno del quale vengono principalmente svolte la gestione dello spazio, l'aggiornamento delle informazioni, relative anche agli ologrammi, e la renderizzazione del contenuto visibile all'utente. Nel momento in cui è stata creata la finestra dell'applicazione ed essa viene visualizzata, parte il loop caratteristico dell'applicazione che termina solo nel momento in cui la finestra viene chiusa. Nel loop principale, viene richiamato il metodo Update, il quale restituisce un HolographicFrame, cioè una visione dello spazio olografico in un preciso istante, la quale viene poi renderizzata per essere visibile all'utente. Le condizioni fondamentali affinchè queste due funzionalità siano eseguite, sono che la finestra sia appunto creata e visibile all'utente, e che lo spazio olografico, cioè HolographicSpace sia stato definito. Infatti è dal HolographicSpace che il metodo Update ricava il frame che verrà renderizzato. Di seguito viene riportata la porzione di codice relativo al ciclo appena descritto, in cui si possono notare i dettagli spiegati.

```
1  public void Run()
2          {
3              while (!windowClosed)
4              {
5                  if (windowVisible && (null != holographicSpace))
6                  {
7                      CoreWindow.GetForCurrentThread().Dispatcher.
8      ProcessEvents(CoreProcessEventsOption.ProcessAllIfPresent);
9
10                     HolographicFrame frame = main.Update();
11
12                     if (main.Render(ref frame))
13                     {
14                         deviceResources.Present(ref frame);
15                     }
16                 }
17                 else
18                 {
19                     CoreWindow.GetForCurrentThread().Dispatcher.
20     ProcessEvents(CoreProcessEventsOption.ProcessOneAndAllPending);
21                 }
22             }
23         }
```

All'interno del metodo Update, solitamente vengono gestiti tre aspetti cri-

tici, la creazione di un oggetto di tipo HolographicFrame, il controllo che non siano stati rilevati input, e l'aggiornamento degli ologrammi visualizzati nello spazio, come è stato accennato, partendo da un oggetto di tipo HolographicSpace, si ricava tramite una funzione apposita un HolographicFrame che sarà restituito poi dal metodo. Ad ogni creazione del frame c'è il bisogno di controllare che non siano state eseguite delle azioni di input che potrebbero modificare il frame da renderizzare, e questo controllo viene fatto controllando se ci sono input nel gestore degli eventi di input. La modifica del frame nel caso degli input, normalmente avviene sugli ologrammi, quindi è importante prima eseguire il controllo di eventuali input, e successivamente, aggiornare le informazioni relative agli ologrammi.
Ogni ologramma ha una struttura basata anch'essa sui metodi Update e Render. Il metodo Update di un ologramma viene richiamato per aggiornarne le informazioni, all'interno del metodo Update dell'applicazione, e analogamente il metodo Render di un ologramma viene richiamato all'interno del metodo Render dell'applicazione.

Aggiornamento Ologramma La parte di gestione dell'ologramma viene effettuata nel ciclo di controllo principale dell'applicazione, in quanto nel metodo Update dell'applicazione viene richiamato lo stesso dell'ologramma. Il metodo Update viene richiamato una sola volta per ogni frame e al suo interno si dovranno specificare i cambiamenti che avvengono nell'ologramma ogni volta che viene creato un frame. La stessa cosa vale per il metodo Render, esso viene richiamato con la stessa frequenza, dal metodo Render dell'applicazione, e si occupa della renderizzazione dell'ologramma.

Intercettazione Input Ogni comando di input, viene rilevato da un Handler, il quale si occupa della rilevazione degli input da parte dell'utente. L'handler si compone principalmente di due parti fondamentali, quale la creazione di un Recognizer in grado di mettersi in ascolto e rilevare gli eventi di un eventuale input, e una funzione che permette al ciclo di controllo dell'applicazione di controllare se siano presenti input per poi agire di conseguenza. Il rilevatore di movimenti della mano è di tipo SpatialInteractionManager, ed è in grado di rilevare gli input, invece il Recognizer in questione è un oggetto di tipo SpatialGestureRecognizer, fornito nelle Windows Holographic API, che si occupa di riconoscere il tipo di input rilevato dal dispositivo. L'handler, come specificato in precedenza, va inizializzato nel momento in cui si setta lo spazio olografico, per permettere che ci sia fin da subito una rilevazione di eventuali input.

3.4 Principi di base in DirectX

Verranno ripresi ora i concetti riguardanti i principi di base di Hololens, per vedere come le varie funzioni possono essere implementate all'interno di un'applicazione. Si analizzeranno le funzioni principali che un'applicazione olografica non può non implementare: i sistemi di coordinate riferiti agli ologrammi, la gestione del gaze, la rilevazione delle gesture, e l'acquisizione dello spatial mapping. Verranno descritte inoltre le modalità con cui è possibile implementarli e le principali API che vengono utilizzate per ogni funzione. La descrizione dei principi di base utilizzando le API di DirectX, si rifà alla documentazione fornita dai produttori del dispositivo [4] [13] [14].

3.4.1 Sistemi di Coordinate in DirectX

Come accennato a livello teorico in precedenza, i contenuti dell'applicazione, cioè gli ologrammi, devono essere posizionati secondo uno specifico sistema di coordinate al fine di essere visualizzati. Questa funzionalità è messa a disposizione dalle Windows Holographic API: la definizione del sistema di coordinate spaziali è fattibile usando un oggetto di tipo SpatialCoordinateSystem, per quanto riguarda invece la creazione di uno stationary frame of reference c'è un tipo di oggetto chiamato appunto SpatialStationaryFrameOfReference, e ne esiste uno analogo per l'attached frame denominato SpatialLocatorAttachedFrameOfReference.
Per creare uno StationaryFrameOfReference e quindi fissare l'ologramma in punto fisso del mondo, bisogna inizialmente usare una funzione dell'oggetto di tipo SpatialLocator, che a livello concettuale sta a rappresentare un tracciante della posizione del device Microsoft Hololens e segue i suoi movimenti. Dato che il frame di riferimento viene creato nella posizione in cui si trova il dispositivo in quel momento, inizialmente si ottiene la posizione corrente del device, e successivamente si crea un sistema di riferimento la cui origine sarà collocata nella posizione appena rilevata. Di seguito le righe di codice utili a svolgere questa funzionalità.

```
1 SpatialLocator                    locator;
2 SpatialStationaryFrameOfReference referenceFrame;
3 referenceFrame =
      locator.CreateStationaryFrameOfReferenceAtCurrentLocation();
```

Basandosi sul sistema di riferimento appena creato è necessario impostare un'istanza del sistema di coordinate, ogni volta che si crea un frame nel metodo Update, che poi viene utilizzato per aggiornare l'ologramma e per specificare il

punto focale del frame. Di seguito la riga di codice necessaria per la creazione del sistema di coordinate

```
SpatialCoordinateSystem currentCoordinateSystem =
    referenceFrame.CoordinateSystem;\\
```

Per quanto riguarda l'ancoraggio nel mondo olografico, utile ad evitare il drifting, esiste un oggetto di tipo SpatialAnchor. Per crearne un'istanza, si parte prendendo come riferimento il sistema di coordinate spaziali in uso nell'applicazione, lo stesso che era stato creato in precedenza. In base a tale sistema di coordinate si crea una nuova ancora. Durante la creazione dell'ancora, è richiesto di specificare come parametri il sistema di riferimento in uso, e un eventuale scostamento sugli assi, indicato da un vettore a tre dimensioni, il quale indica la posizione in cui verrà creata l'ancora. Dal momento in cui viene creata l'ancora è possibile accedere al sistema di coordinate dell'ancora, e poi è possibile posizionare un ologramma su tale sistema di coordinate, in modo da evitare l'effetto di drifting che si verrebbe a creare. L'ologramma posizionato su questo sistema di riferimento sarà relativo all'ancora e non al sistema di riferimento principale dell'applicazione. Negli esempi di codice qui sotto viene mostrato ciò che è stato appena descritto.

```
SpatialCoordinateSystem currentCoordinateSystem =
    referenceFrame.CoordinateSystem;
SpatialAnchor anchor =
    SpatialAnchor.TryCreateRelativeTo(currentCoordinateSystem, new
    Vector3(3.0f, 0.0f, 0.0f));
SpatialCoordinateSystem anchorCoordinateSystem =
    anchor.CoordinateSystem;\\
```

Nel caso volessimo invece rappresentare un ologramma che segua l'utente, usiamo un AttachedFrameOfReference. La classe che rende possibile la creazione di un sistema di coordinate di questo tipo è SpatialLocatorAttachedFrameOfReference. Si parte anche qui dal Locator, classe utilizzata per lo StationaryFrameOfReference, per ottenere la posizione del device in quel momento e creare un sistema che abbia inizialmente origine in quella posizione. Inizialmente, perchè, si dovrà tenere conto della posizione dell'utente ad ogni momento dato che l'ologramma dovrà seguirlo, questo sta a significare che questo tipo di sistema di coordinate non è fisso come i precedenti, ma risulta dinamico e attento ai movimenti. Si usa una variabile di tipo HolographicFramePrediction con lo scopo di fare una previsione della posizione dell'utente al momento del frame

successivo.

```
SpatialLocator                    locator;
SpatialLocatorAttachedFrameOfReference attachedReferenceFrame;
attachedReferenceFrame =
    locator.CreateAttachedFrameOfReferenceAtCurrentHeading();
currentCoordinateSystem = attachedReferenceFrame.
GetStationaryCoordinateSystemAtTimestamp(prediction.Timestamp);
```

L'ultima istruzione come nel caso di un frame stazionario, va eseguita nel momento in cui si va a creare il frame nel metodo Update.

3.4.2 Gaze in DirectX

Il gaze rappresenta la direzione in cui l'utente sta guardando, il puntatore del mouse, e indica ciò su cui si vuole eseguire qualche azione. Per ottenere queste funzioni vengono combinate la direzione del dispositivo e il suo orientamento, sempre in un sistema di coordinate specificato dall'app. Il punto di accesso per ottenere le informazioni relative al gaze è un oggetto di tipo SpatialPointerPose, istanziabile invocando un metodo della stessa classe. Dal momento in cui questo oggetto viene creato si possono ottenere le informazioni relative al gaze: principalmente viene calcolata la posizione della testa, in un vettore di tre elementi, e la direzione del gaze in avanti, specificata in un vettore unitario che ne descrive appunto la direzione stessa, ForwardDirection, è possibile ottenere anche la direzione del gaze verso l'alto, chiamata UpDirection, fornita anche quest'ultima in un vettore. I vettori che rappresentano le informazioni del gaze, essendo relativi al pointerPose, sono forniti nello stesso sistema di coordinate con il quale esso era stato richiesto.

```
SpatialPointerPose pointerPose =
    SpatialPointerPose.TryGetAtTimestamp(currentCoordinateSystem,
    prediction.Timestamp);
Vector3 position = pointerPose.Head.Position;
Vector3 forwarddirection = pointerPose.Head.ForwardDirection;
Vector3 updirection = pointerPose.Head.UpDirection;
```

3.4.3 Gesture in DirectX

E' stato descritto che i movimenti della mano rilevati sono solamente due, e che le gesture si compongono di questi due movimenti in un diverso lasso di

tempo o sequenza, quindi in base alla pressione più o meno prolungata, oppure diretta, il dispositivo capisce che tipo di gesture sta eseguendo l'utente. Questa differenza è presente anche a livello di sviluppatore, infatti c'è una parte che si occupa di rilevare le mani e i suoi movimenti, e un'altra in grado di distinguere i movimenti rilevati identificandoli nelle gesture. La prima parte, si occupa di rilevare i movimenti della mano, in particolare quando la mano entra ed esce nello spazio utile a rilevare i movimenti, e accorgendosi di quando la mano si trova in uno stato di ready o pressed state. Il manager di questi eventi, chiamato SpatialInteractionManager, ha un evento che notifica all'applicazione quando un movimento della mano viene rilevato. Alla rilevazione dell'evento di pressed da parte della mano, questo viene comunicato all'applicazione in modo asincrono. Di seguito le righe di codice per istanziare l'InteractionManager e agganciargli gli eventi.

```
1 SpatialInteractionManager interactionManager;
2 interactionManager = SpatialInteractionManager.GetForCurrentView();
3 interactionManager.SourcePressed += this.OnSourcePressed;
```

L'evento OnSourcePressed si occupa di cambiare lo stato della rilevazione della pressione o meno, come riportato.
Sarà l'applicazione ad occuparsi di controllare quando sono presenti degli input, e lo farà come specificato in precedenza, all'interno del metodo Update.
E' quindi necessario implementare un metodo che restituisce il tipo di interazione e se è stata rilevata o meno.
In SpatialInteractionManager sono presenti anche due eventi che vengono notificati quando una mano entra o esce dallo spazio in cui vengono rilevati i gesti, questi eventi sono chiamati SourceDetected e SourceLost. Per quanto riguarda l'interpretazione dei gesti effettuati dalla mano, se ne occupa una parte diversa di API, chiamata SpatialGestureRecognizer. Il suo scopo è quello di distinguere le varie gesture, per non ottenere risultati ambigui dato che i tipi di movimenti riconosciuti sono solo due. Bisogna definire uno SpatialGestureRecognizer per ogni tipo di gesture che si vuole riconoscere, specificandone il tipo di Gesture fra quelle riconosciute dal dispositivo e specificate in un enum chiamato SpatialGestureSettings delle Windows Holographic API.

```
1 SpatialGestureRecognizer tapGesture;
2 tapRecognizer = new
     SpatialGestureRecognizer(SpatialGestureSettings.Tap);
```

E' fondamentale la gestione dell'evento InteractionDetected dello SpatialInteractionManager, in cui all'interno dell'evento si va a catturare l'interazione

voluta tramite il metodo CaptureInteraction. Generalmente nel metodo dell'evento Tapped, si settano i valori dei dati relativi alla gesture come la gesture registrata e il gaze, e l'aggiornamento della variabile che rappresenta se è stata rilevata una gesture.
Di seguito un esempio esplicativo.

```
1  private void InteractionDetected(SpatialInteractionManager sender,
       SpatialInteractionDetectedEventArgs args)
2          {
3              this.tapRecognizer.CaptureInteraction(args.Interaction);
4          }
5
6  private void Tapped(SpatialGestureRecognizer sender,
       SpatialTappedEventArgs args)
7         {
8             this.interaction.First = SpatialGestureSettings.Tap;
9             this.interaction.Second = this.gaze;
10            this.interactionOccurred = true;
11         }
```

Dal momento in cui è stata catturata l'interazione, è necessario combinare tale informazione con lo SpatialPointerPose, che contiene le informazioni del gaze, per poi eseguire il tipo di modifica o di cambiamento che l'applicazione prevede nel punto in cui l'utente ha specificato l'interazione.

3.4.4 Spatial Mapping in DirectX

Per l'implementazione delle funzionalità dello spatial mapping con DirectX e le Api di Windows, si usano i tipi di oggetti definiti sotto il nome di Windows.Perception.Spatial. I principali tipi forniti per lo sviluppo della mappatura spaziale sono i seguenti [4]:

- SpatialSurfaceObserver: il quale fornisce oggetti di tipo SpatialSurfaceInfo contenenti le informazioni sulle superfici relative alle zone specificate vicino all'utente.

- SpatialSurfaceInfo: il quale descrive una singola superficie esistente: questa è identificata da un ID univoco, contiene il bounding volume e il tempo dall'ultimo cambiamento. Fornirà anche una SpatialSurfaceMesh se viene richiesta.

- SpatialSurfaceMesh: rappresenta i dati della mesh per una singola superfice spaziale. I dati per la posizione dei vertici, i vertici normali e

gli indici dei triangoli sono contenuti negli stati degli oggetti di tipo SpatialSurfaceMeshBuffer.

- SpatialSurfaceMeshBuffer: rappresenta un singolo tipo di dato della mesh.
- SpatialSurfaceMeshOptions: contiene i parametri usati per personalizzare gli oggetti di tipo SpatialSurfaceMesh richiesti da SpatialSurfaceInfo.

SurfaceObserver è il punto di partenza per quanto riguarda l'utilizzo dello spatial mapping, e Microsoft indica le linee guida per usare tale elemento[4]: Per prima cosa bisogna creare un oggetto di tipo SpatialSurfaceObserver. E' importante controllare che l'utente abbia dato i permessi all'applicazione di utilizzare le funzionalità di scansione dell'ambiente al dispositivo, altrimenti tutta la procedura di spatial mapping non funzionerà. Generalmente i permessi vengono impostati nel file xml dell'applicazione. Successivamente si definiscono uno o più volumi spaziali per definire le regioni di interesse di cui si vuole ricevere i dati di mappatura dello spazio. Questa funzione viene eseguita utilizzando uno spatial volume, chiamato SpatialBoundingVolume, con un sistema di coordinate world-locked per identificare una zona fissa del mondo fisico. Per specificare le zone di spazio delle quali si vogliono rivedere le informazioni relative, si usa SetBoundingVolumes, i volumi definiti possono anche essere cambiati successivamente. Il passo successivo è ricevere le informazioni, e si può fare polling sul SurfaceObserver in qualsiasi momento, oppure è possibile registrare l'evento del SurfaceObserver chiamato ObserverSurfacesChanged il quale si attiva ogni volta che sono disponibili delle informazioni sulle superfici spaziali nelle regioni specificate in precedenza. E' possibile ricevere una mappa di oggetti di tipo SpatialSurfaceInfo, con la quale si possono aggiornare i dati delle superfici spaziali presenti nell'ambiente intorno all'utente. Per ogni superficie spaziale si può per esempio chiamare il metodo TryGetBound, per provare a stabilire le dimensioni della superficie espressa in un sistema di coordinate scelto. E' possibile anche richiedere una mesh per una superficie, usando il metodo TryComputeLatestMeshAsync. Ricevuta la mesh, di tipo SpatialSurfaceMesh, bisogna processarla. A partire da questo oggetto, come specificato in precedenza, è possibile avere gli oggetti contenuti nello SpatialSurfaceMeshBuffer, i quali sono in un formato di dati elaborabile anche dalle API di Direct3D. Sarà necessario scorrere la serie di superfici fornite e classificare le superfici in aggiunte, cambiate o rimosse. Per ogni superficie aggiunta o cambiata, se è utile si può fare una richiesta asincrona per ricevere gli aggiornamenti della mesh che rappresenta lo stato della superficie ad un certo livello di dettaglio. A questo punto si possono utilizzare le mesh per vari scopi all'interno dell'applicazione, in primis vengono usate per il rendering delle stesse, ma anche per gli scopi descritti in precedenza nello spatial mapping.

3.5 Esempio di Holographic App

A corredo delle informazioni e dell'analisi effettuata su ogni aspetto che caratterizza un'applicazione olografica, si riporta un esempio basilare che riprende i concetti appena descritti mettendoli in pratica. L'applicazione è sempre caratterizzata da un loop principale, dai metodi di gestione degli ologrammi appena descritti, e ha la modalità di intercettazione degli input descritta in precedenza. Viene analizzata ora l'applicazione delle funzionalità di base ad un semplice contesto dimostrativo.

3.5.1 Funzionalità principali

In questa applicazione, nello specifico, viene rappresentato un cubo come unico ologramma, il quale ad ogni frame, ruota, producendo così un effetto di rotazione continuo data la composizione di un loop e il succedersi dei frame uno dietro l'altro. La struttura architetturale dell'applicazione rispecchia quella descritta in precedenza. All'interno dell'applicazione viene utilizzato un frame di riferimento fisso nel mondo per posizionare l'ologramma, senza l'utilizzo di ancore. Un'altra funzionalità utilizzata è una gesture di tipo "Press e Release", che eseguirà un'azione sull'ologramma, e l'utilizzo del Gaze per eseguire l'azione stessa. Nel caso venga rilevata una gesture di tipo press e release, quindi corrispondente ad un tap, l'ologramma verrà riposizionato davanti alla visuale dell'utente. La gesture può essere effettuata in qualsiasi parte dello spazio olografico per essere rilevata. L'esecuzione di questa funzionalità apparentemente semplice implica l'utilizzo di vari principi di base specificati di seguito in ordine d'esecuzione:

- Rilevazione dell'input
- Setting del riposizionamento dell'ologramma
  - Uso del gaze per rilevare lo sguardo dell'utente
  - Reimpostazione della posizione dell'ologramma
- Aggiornamento dell'ologramma.
- Render dell'ologramma.

Queste funzioni principali, dopo aver settato lo spazio olografico, sono eseguite all'interno del main dell'applicazione, tranne quelle relative al riposizionamento dell'ologramma, che vengono rimandate all'ologramma stesso, infatti all'interno del loop, quando viene rilevato un input viene richiamato il metodo che

all'interno della classe dell'ologramma si occuperà dell'azione relativa alla gesture effettuata. Di seguito è riportato uno screenshoot dell'applicazione in esecuzione, in cui si può vedere come viene rappresentato l'ologramma.

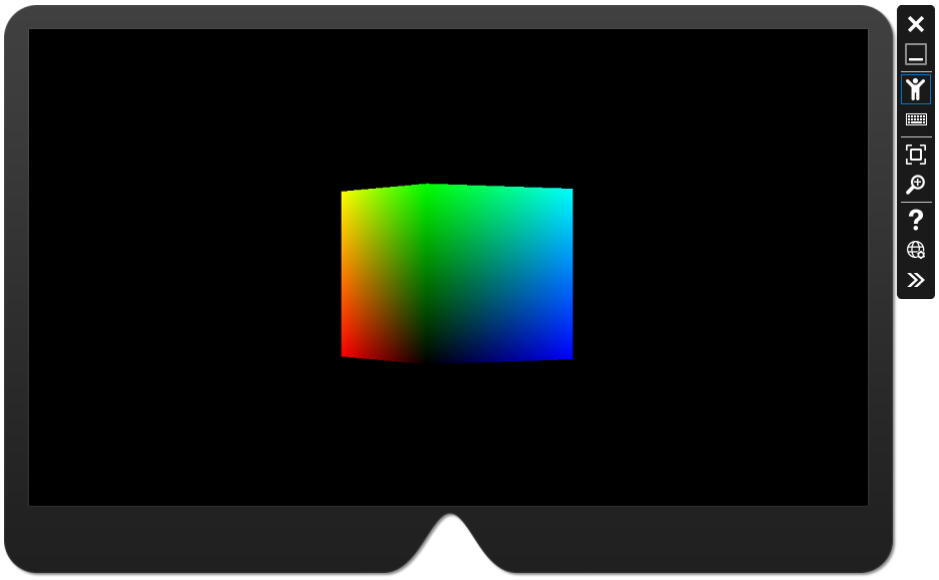


Figura 3.1: Esempio di applicazione dei concetti di base di Microsoft Hololens

3.5.2 Considerazioni sull'esempio

Verranno analizzati nei paragrafi successivi i vari aspetti dell'applicazione, andando a studiare le varie scelte implementative e a fare considerazioni a livello di progettazione e implementazione, al fine di capire bene cosa comporta ogni scelta effettuata. L'applicazione sovra citata, rappresenta un esempio dell'utilizzo di ogni concetto spiegato in questo capitolo. Il programma si svolge in un unico ciclo di controllo. Questa modalità implica che ad un punto del loop l'applicazione dipenda dall'esecuzione dei metodi riferiti all'ologramma, di modo che se si facesse un'azione bloccante all'interno dell'ologramma, anche tutta l'applicazione risulterebbe bloccata. E' evidente che non ci sia separazione tra le varie parti di controllo dell'applicazione. Supponendo di avere più ologrammi all'interno dell'applicazione, è auspicabile che nel caso ci fosse un'azione bloccante su uno di essi, gli altri ologrammi continuino ad eseguire il loro comportamento senza dipendere da quello bloccato, in modo che anche l'applicazione continui il suo funzionamento, avendo si una parte in attesa ma essendo comunque in grado di svolgere altre funzioni allo stesso tempo. In questa modalità quindi nel caso ci fossero più ologrammi e uno fosse bloccante ad un certo punto, anche gli altri ologrammi, magari pur essendo concettualmen-

te indipendenti, non potrebbero proseguire il proprio aggiornamento, creando così uno svantaggio non da poco.
Per quanto riguarda gli ologrammi veri e propri, Microsoft pone ogni aspetto all'interno della stessa classe: dall'applicazione verrà creata un'istanza di tale classe per creare un ologramma, e qualsiasi tipo di operazione riferita all'ologramma verrà eseguita agendo su questo oggetto. La classe viene dichiarata come un Renderer, quindi non è espresso il concetto di ologramma vero e proprio, ma viene indicata come una semplice renderizzazione di elementi geometrici. Trattandosi di applicazioni olografiche, in cui ogni cosa è olografica, era auspicabile che il concetto del più semplice oggetto olografico, cioè l'ologramma, fosse espresso, invece ogni oggetto appartenente alla scena olografica non è altro che un insieme di punti geometrici, i quali vengono collegati tra loro a formare l'oggetto desiderato e poi posti all'interno della scena in una determinata posizione. Non esistendo il concetto di ologramma vero e proprio, a livello concettuale e implementativo, si può affermare che l'applicazione olografica sia formata da uno spazio olografico il quale rappresenta la zona in cui l'applicazione mixa realtà e virtuale, all'interno del quale sono renderizzate delle forme geometriche sotto forma di ologramma.
In un'applicazione olografica, all'interno della classe che rappresenta l'ologramma, sono inseriti tutti gli aspetti dello stesso, dalle informazioni relative alla sua visualizzazione grafica, alle azioni da intraprendere nel caso fossero rilevate azioni di input. Sono presenti anche i metodi che si occupano dell'aggiornamento e del render dell'ologramma stesso, rispettivamente i metodi update e render, richiamati nei medesimi metodi dell'applicazione. Questa scelta implementativa non distingue i vari aspetti dell'oggetto da renderizzare, mescolando aspetti grafici con aspetti comportamentali. Assumiamo come esempio, il caso in cui l'ologramma da rappresentare fosse una persona e non un cubo: la persona ha una varietà maggiore di comportamenti che può assumere, in quanto oltre a ruotare come il cubo, può muoversi nello spazio in base allo spatial mapping rilevato, può emettere suoni spaziali, e assumere molteplici comportamenti in relazione a diverse gesture che vengono eseguite dall'utente. Oltre al numero elevato di comportamenti e reazioni che può avere l'ologramma della persona, questo al fine di essere reale e non fare distinguere le differenze tra reale e virtuale, deve essere rappresentato con enormi informazioni grafiche, oltre a specificare ogni punto che ne disegna la forma, potrebbero essere necessarie alcune texture per implementare le diverse sfumature dell'ologramma e i tratti somatici della persona a livello grafico. Si intuisce che ci sia una quantità enorme di informazione riferita allo stesso ologramma, e seguendo l'esempio, si verrebbe a creare un'unica ed enorme classe, con tutte le informazioni relative alla parte grafica e alla parte comportamentale racchiuse insieme. Si verrebbe così a creare una notevole difficoltà di comprensione del codice, rendendo

difficile anche un'eventuale modifica perchè per andare a modificare un dettaglio della persona olografica, si dovrebbe andare a cercare all'interno dell'unica classe. Con questa scelta implementativa ne risente anche il riuso del codice e la sua estendibilità, infatti nel caso si dovesse creare un'altra persona, sarebbe difficile utilizzare lo stesso codice e andare a modificarne qualche aspetto o qualche comportamento, perchè non facilmente identificabili. La stessa cosa vale nel caso in cui si volessero aggiungere dei comportamenti alla persona, che significa aggiungere delle funzionalità all'ologramma da eseguirsi al verificarsi di determinati eventi, non sarebbe immediato come lo è l'idea di farlo, in quando all'interno dell'ologramma si dovrebbe andare a gestire ogni aspetto, aumentando ancora di più la confusione fra i vari aspetti dell'ologramma e il volume della classe stessa che lo rappresenta.
Seguendo i principi dell'ingegneria del software, sarebbe stato più utile, a favorire la comprensione del codice, la manutenzione, l'estendibilità o il riuso, dividere ogni aspetto in moduli ed eventualmente sottomoduli, per separare bene ogni aspetto dell'applicazione, mettendoli poi in relazione fra loro.

Capitolo 4

HoloFrameOfMind

HoloFrameOfMind è un'applicazione che si pone come obbiettivo principale, la riprogettazione del concetto di ologramma all'interno di una applicazione olografica, separando in modo netto e chiaro i vari aspetti, a differenza di quanto viene effettuato nell'esempio citato in precedenza.
Questa applicazione rappresenta lo stato d'animo di una persona sotto forma di un cubo il quale viene rappresentato all'interno dello spazio olografico come ologramma, assumendo diverse forme in base allo stato d'animo della persona alla quale si riferisce. La rappresentazione è ovviamente fittizia e a puro scopo raffigurativo, in quanto l'obiettivo dell'applicazione è quello descritto in precedenza, e non quello di rappresentare in modo concreto lo stato d'animo di una persona reale nel mondo, questa idea viene usata a scopo scenico e come esempio di una applicazione olografica. Per quanto riguarda lo stato d'animo della persona, se questa è felice, viene rappresentato con il colore verde e una rotazione tranquilla, a differenza dello stato d'animo arrabbiato raffigurato con un rosso acceso e una rotazione furiosa dell'ologramma stesso. Nel caso invece lo stato d'animo fosse triste il cubo assumerebbe una posizione statica, senza più ruotare ed un colore grigio, invece nel caso la persona fosse affamata questo diverrebbe di colore giallo e con una flebile rotazione, come se avesse necessità di essere ricaricato.
In questo senso è stato deciso il titolo, combinando l'idea di rappresentare lo stato d'animo di una persona sotto forma di ologramma. Ogni persona in base allo stato d'animo, ha comportamenti e reazioni diverse, e dato che si sta parlando di applicazioni olografiche le quali renderizzano un frame ogni volta, cioè forniscono una rappresentazione dello spazio olografico in un certo momento, si è scelto di combinare le due cose. E' come se si creasse una rappresentazione della mente della persona, cioè un frame della mente della persona, raffigurato come un ologramma, da cui appunto HoloFrameOfMind.
Saranno analizzate ora le scelte di progettazione e implementative per appor-

tare i vantaggi prefissati come obiettivo.

4.1 Architettura principale

In questa sezione verrà descritta in primis la nuova architettura dell'applicazione, spiegando i vari moduli che la compongono, evidenziando così la separazione delle parti principali del programma. L'applicazione precedente si presentava come un insieme di funzioni senza distinzione fra loro, HoloFrameOfMind invece distingue le varie parti come verrà spiegato nel paragrafo successivo. La nuova applicazione sfrutta sempre le WindowsHolographic API e le funzioni in DirectX per la creazione degli elementi olografici.

Funzionamento generale L'applicazione è caratterizzata sempre da un loop, il quale a patto che la finestra della stessa sia visibile e sia stato creato uno spazio olografico, ricava un frame che verrà poi renderizzato all'utente. L'ottenimento del frame avviene sempre richiamando il metodo update del main del programma, e la renderizzazione del frame ottenuto viene sempre eseguita dal metodo render del main del programma. Dal momento in cui ci si trova all'interno del main del programma, vengono comunque settate le impostazioni principali, come per esempio il settaggio dello spazio olografico relativo.
E' possibile notare una prima notevole differenza per quanto riguarda la gestione dei contenuti dell'applicazione. E' stata applicata infatti una netta separazione fra la gestione dell'applicazione e degli ologrammi al suo interno, per questo è stato creato un HoloManager, il quale si occupa della gestione dei suddetti. HoloManager viene usato all'interno dell'applicazione per racchiudere tutti gli ologrammi presenti nella scena, gestendo sia l'aggiunta che la rimozione degli stessi, occupandosi dell'aggiornamento delle informazioni e della renderizzazione di ogni ologramma, fino alla rilevazione delle gesture relative effettuate eventualmente su un ologramma della scena.
Da parte dell'applicazione ci sarà una aggiunta di un ologramma nel momento in cui si setta lo spazio olografico, richiamando il metodo apposito della classe HoloManager. Quando l'applicazione crea il frame da renderizzare, richiamerà il metodo Update implementato all'interno di HoloManager, il quale si occupa dell'aggiornamento di tutti gli ologrammi che in precedenza sono stati aggiunti alla scena. Analogamente quando l'applicazione deve renderizzare il frame ottenuto, oltre a verificare i parametri della camera dell'utente, richiamerà il metodo Render di HoloManager che si occupa della renderizzazione degli ologrammi presenti nello spazio.
Ogni oggetto affidato ad HoloManager è di tipo HoloRenderer, il quale si oc-

cupa della gestione del singolo ologramma, fino a renderizzarlo, come descrive il nome della classe. HoloManager quindi implementerà una lista di gestori del singolo ologramma, i quali si occupano del singolo oggetto rappresentato. Siamo giunti quindi ad una divisione dal generale al particolare in quanto l'applicazione gestisce un manager di tutti gli ologrammi della scena, e a sua volta il manager degli ologrammi, HoloManager, si occupa dei vari gestori del singolo ologramma, HoloRenderer. HoloRenderer, apprese le informazioni relative alle risorse del dispositivo e alla grafica riferita all'ologramma, è in grado di renderizzare un oggetto trasformandolo in ologramma. HoloRenderer, per ogni ologramma, utilizza i metodi Update e Render, nel primo vengono settati eventuali cambiamenti di posizione a livello grafico, e nel secondo viene renderizzato l'ologramma dopo gli aggiornamenti che potrebbero essere stati eseguiti. HoloManager all'interno del suo metodo Update richiamato dal main, andrà quindi a richiamare il medesimo metodo però riferito all'ologramma singolo, per ogni istanza di HoloRenderer che era stata aggiunta alla scena olografica.
Questa distinzione dal generale al particolare porta ad avere un'esecuzione tipo: l'applicazione, per generare un frame, necessita di aggiornare il gestore degli ologrammi, il quale a sua volta aggiorna ogni renderizzatore degli ologrammi, ottenendo così alla fine il frame pronto per essere renderizzato. La fase seguente implica che l'applicazione, una volta ottenuto il frame, richiami il metodo Render del gestore degli ologrammi, affinchè esso per ogni Renderer aggiunto, richiami il medesimo metodo relativo ad ogni ologramma. Con un approccio inverso, è la definizione delle informazioni di renderizzazione di ogni ologramma, che verrà incorporata all'interno di un unico frame, il quale poi verrà renderizzato dall'applicazione tenendo conto delle camere attive e della posizione dell'utente come specificato in precedenza. Questa distinzione di responsabilità fra le parti del programma, porta un maggior controllo su quali ologrammi sono presenti all'interno della scena, rendendo un'azione semplicissima l'aggiunta e la rimozione di ologrammi dalla scena, garantendo inoltre sempre un quadro chiaro sulle parti relative all'applicazione e quelle relative esclusivamente agli ologrammi. Effettuando questa scelta, viene aggiunto così anche il concetto di vero e proprio ologramma presente nella scena olografica: questo concetto è stato ritenuto fondamentale fosse inserito all'interno di un'applicazione di questo tipo. Non è stato ancora affrontato come vengono rappresentate le informazioni che il renderer di ogni ologramma riceve, cioè come viene rappresentato l'ologramma vero e proprio. Nella sezione successiva sarà descritto nel dettaglio come può essere progettato e implementato un ologramma all'interno di un'applicazione olografica.

4.2 Progettazione dell'ologramma

Principi fondamentali di progettazione Viene interpretato l'ologramma come un'entità esterna al sistema, supponendo poi che l'applicazione sia in grado di raffigurare olograficamente tale entità all'interno del proprio spazio olografico, permettendo interazioni con la stessa, ed è stato ridefinito e riprogettato il concetto di ologramma e la gestione dello stesso ologramma all'interno di HoloFrameOfMind, avendo questo concetto come idea di base. L'ingegneria del software suggerisce alcuni principi per la progettazione, quali ad esempio separazione degli argomenti e anticipazione dei cambiamenti. Per quanto riguarda la separazione degli argomenti, è suggerito dalla materia di individuare i vari aspetti del problema che si pone, e trattarli in modo ben distinto per ottenere così una soluzione più semplice e intuitiva. Per applicare il concetto di anticipazione dei cambiamenti invece bisogna progettare il sistema non solo occupandosi del problema che si pone al momento, ma tenendo conto di possibili sviluppi futuri e di come il sistema potrebbe cambiare, in modo da rendere poi meno dispendiose e più semplici tutte le modifiche che verranno apportate in futuro. La gestione e il concetto dell'ologramma sono stati progettati basandosi su questi principi, e a livello implementativo viene fatto uso di un design pattern riconosciuto e ampiamente utilizzato. Un design pattern non è altro che uno schema progettuale, il quale definisce a livello generale la modalità di applicazione dei principi sopra descritti. Esistono vari design pattern, ognuno in base alle necessità di sviluppo del software, il design pattern utilizzato per la riprogettazione dell'ologramma in questo caso è MVC: Model View Controller.
Il design pattern MVC è generalmente un modello di progettazione utilizzato nella programmazione ad oggetti, in questo caso viene applicato al concetto di ologramma, e non all'intera applicazione. MVC suggerisce la separazione in tre parti ben distinte: una chiamata "controller" la quale si occuperà della logica di controllo specificando le azioni da eseguire in caso di rilevazione di eventi sul sistema, una parte definita "view" che si occupa della gestione degli aspetti di visualizzazione, e una parte di "model" cioè di modello la quale conterrà i dati necessari e riferiti all'oggetto.

Uso dei principi nell'applicazione Applicando questo pattern al concetto di ologramma, e ricordando l'obiettivo fissato ad inizio sezione, la distinzione dell'entità esterna al sistema che dovrà essere rappresentata dall'applicazione, avviene nel modo seguente.
L'entità avrà delle caratteristiche che la contraddistinguono, come per esempio uno stato, dovrà poi essere rappresentata graficamente in un certo modo e

reagire agli eventi che si potrebbero verificare. Viene definita così una parte di model riferita all'entità, in cui si modellano gli aspetti relativi alle sue caratteristiche di base, in questo caso la parte di modello terrà traccia dello stato dell'entità. La parte di view, interesserà la rappresentazione grafica dell'entità tenendo conto del suo stato attuale, e la parte di controller, reagirà agli eventi e implementerà la logica di controllo dell'ologramma. Per quanto riguarda la comunicazione tra le parti, la parte di modello deve essere osservabile dalla parte di view, questo significa che in caso di cambiamento di stato, il modello informa la view sul fatto che sia cambiato lo stato, e poi la view richiede lo stato al modello, tramite il metodo apposito. La stessa cosa vale tra view e controller, nel caso infatti si verifichi per esempio un input sulla view, essa informerà il controller notificandogli l'evento catturato, il controller in base all'evento verificatosi sarà poi in grado di modificare direttamente lo stato dell'entità di cui si tiene traccia nel modello.
Si riporta di seguito lo schema UML di progettazione delle classi riferite all'ologramma, il quale rappresenta i concetti appena descritti sulla composizione delle varie parti e l'interazione fra di esse.

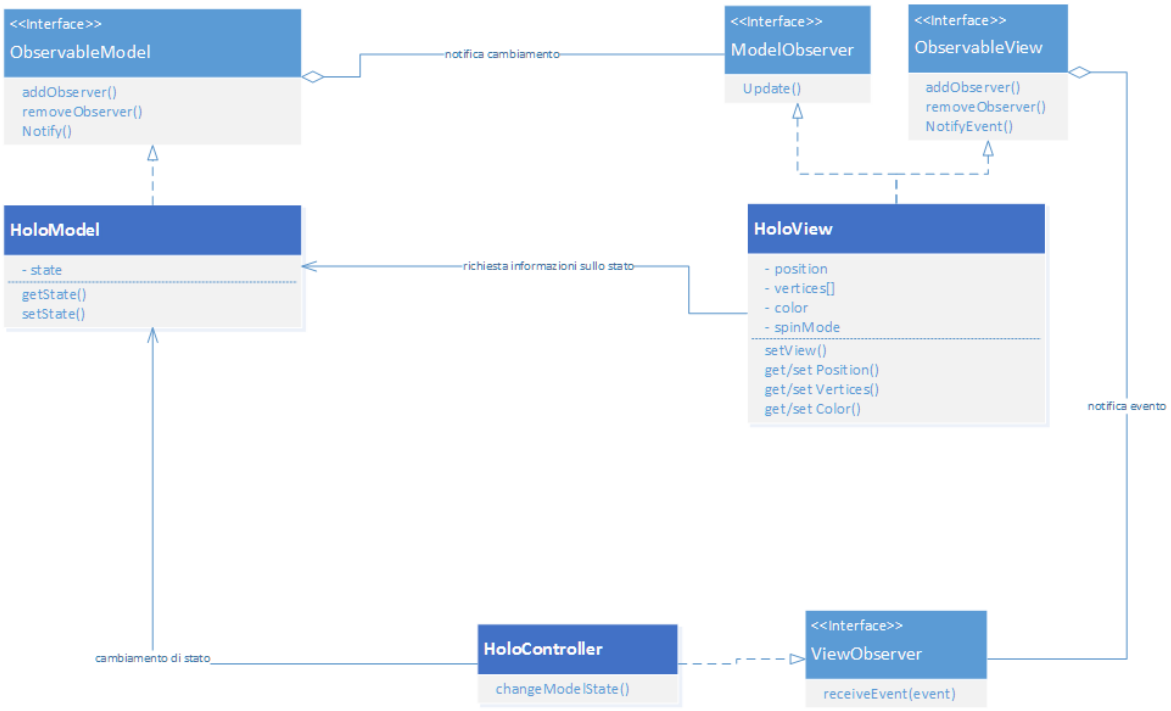


Figura 4.1: Diagramma UML raffigurante le varie parti dell'ologramma.

Come si nota dal diagramma sono state implementate le tre classi rappresentanti la parte di model, view, e controller. Per la comunicazione fra le parti

si usa il pattern Observable/Observer, questo pattern è utilizzato quando vari componenti reagiscono in funzione ai cambiamenti dello stato di un altro componente, e le parti che osservano quella principale si identificano in Observer, e il componente contenente lo stato è Observable. E' il pattern adatto per questo caso in cui il Model contiene uno stato e la view deve rimanere aggiornata sui suoi cambiamenti per rappresentarlo di conseguenza. L'interfaccia observable contiene dei metodi per aggiungere osservatori e rimuoverli unitamente al metodo per notificare i cambiamenti. L'interfaccia Observer invece descrive la necessità di avere un metodo di aggiornamento, il quale verrà richiamato ad ogni cambiamento di stato per ogni osservatore registrato. L'interazione tra model e view corrisponde a questo pattern come si evince dal diagramma, nello stesso modo in cui comunicano view e controller, la view dell'ologramma infatti può ricevere degli input su di essa, e il controller dovrà essere in grado di rilevarli. Per questo la view dovrà essere osservabile dal controller, in quanto quando si verifica un evento, la view notificherà al controller la cattura dello stesso, comunicandogli il tipo di evento registrato, richiamando il metodo NotifyEvent, il controller poi sarà in grado di interpretare l'evento e cambiare direttamente lo stato nel model. Dal diagramma è possibile notare anche la distinzione netta delle parti, infatti il modello dell'entità contiene solamente la proprietà "state" che rappresenta il suo stato, mentre la view contiene tutte le informazioni di rappresentazione grafica come la sua posizione nel mondo, i vertici del cubo, il colore, e la modalità di rotazione. Il controller rappresenta semplicemente la logica di controllo dell'ologramma stesso, il quale cambia in funzione degli input verificatisi.
Viene ora analizzato in modo sequenziale come avviene l'interazione fra le parti, model view controller, dell'ologramma, nel caso in cui fosse rilevata una gesture sull'ologramma in questione. Il diagramma di sequenza sottostante, riporta graficamente le varie fasi.

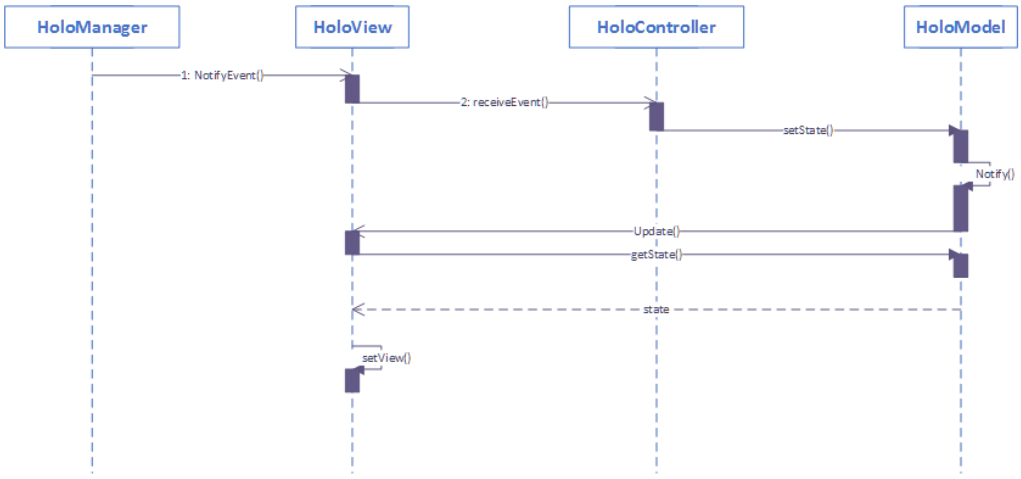


Figura 4.2: Diagramma di sequenza al verificarsi di un input sull'ologramma.

HoloManager, come detto in precedenza, si occupa di rilevare gli input sugli ologrammi, ed è in grado di identificare su quale ologramma della scena viene eseguita la gesture. Nel caso in cui HoloManager rilevi che la gesture sia stata eseguita sul cubo, questo richiama il metodo NotifyEvent della view relativa all'ologramma. Quando all'interno della view viene richiamato questo metodo, sarà la view che si occupa di informare il controller chiamandone il metodo receiveEvent. Il controller, nel caso in cui la gesture rilevata rispecchi quella aspettata, cioè una tap gesture sull'ologramma, andrà a chiamare il metodo setState del modello per cambiare lo stato dell'entità. Dal momento in cui all'interno del modello avviene un cambiamento di stato, il modello stesso richiamerà il metodo Notify, al fine di informare la view. Per ogni view registrata viene chiamato il proprio metodo update. A questo punto la view farà una richiesta al model per conoscere lo stato dell'entità, e il model risponderà restituendo il valore dello stato. La view ha così ottenuto l'informazione necessaria e si aggiornerà di conseguenza. Questo è lo schema di interazione fra le varie componenti dell'ologramma, stando all'utilizzo dei due design pattern, e ogni qualvolta HoloManager rilevi una gesture o un qualsiasi altro tipo di input, verranno eseguiti i seguenti passi per aggiornare la rappresentazione grafica, cioè per cambiare lo stato del cubo.

Vantaggi dell'utilizzo dei pattern L'utilizzo del pattern MVC, oltre a garantire un'architettura basata sulla distinzione fra le varie parti del software, presenta anche altri vantaggi.
E' possibile innanzitutto raffigurare il modello con view diverse, dato che le due parti sono separate, e nel caso si volesse rappresentare lo stato dell'entità in modo diverso, sarà quindi necessario solo riscrivere le caratteristiche grafi-

che senza andare ad intaccare la parte di model. Questo permette anche una manutenzione agevolata, in quanto si saprebbe a priori su quale parte agire, sapendo di non creare cambiamenti anche nelle altre parti del codice. La parte di modello potrebbe essere totalmente cambiata, indicando altre proprietà dell'entità, e apportando varie modifiche, senza toccare le altre parti del codice. L'utilizzo del pattern permette anche che il cambiamento sia fatto senza conoscere come sono state implementate diverse parti fra loro, quindi la parte di view, per esempio, potrebbe essere riscritta da una persona specializzata in grafica tridimensionale, mentre il model da un programmatore, senza conoscere l'uno i dettagli dell'altro, se non per le informazioni da scambiarsi, e nel caso si estendesse l'applicazione ad un livello più alto, la suddivisione del lavoro sarebbe fondamentale, questo tipo di progettazione dell'ologramma permetterebbe tale suddivisione senza problemi. L'applicazione del pattern MVC garantisce inoltre una totale estendibilità del codice, anzi, più il progetto è grande, più i vari vantaggi si notano, in quanto ci sarebbero varie proprietà all'interno di ogni sezione di model, di view e di controller, e il codice diventerebbe illeggibile senza una distinzione adeguata. I vantaggi sovra elencati non erano presenti nell'esempio mostrato in precedenza, e la progettazione si è svolta anche dandosi come obiettivo una futura estendibilità, manutenzione e riuso del codice in modo semplice.

4.3 Dettagli d'implementazione

Entrando nel dettaglio, dopo aver analizzato a livello generale come viene applicato il pattern MVC a livello di ologramma all'interno dell'applicazione, verranno spiegate di seguito nel dettaglio le varie parti in HoloFrameOfMind. La parte di model come spiegato in precedenza, racchiude le informazioni riguardanti lo stato dell'entità, e necessita di essere osservabile dalla parte di view. Il modello implementerà quindi l'interfaccia definita sulla base di Observable, e all'interno della classe è contenuta l'informazione relativa al suo stato e una lista contenente gli osservatori registrati. Per identificare lo stato d'animo della persona, è stato dichiarato un tipo di dato enum, chiamato Mood, il quale può assumere le proprietà "Happy", "Sad", "Angry", e "Hungry".

4.3.1 HoloModel

HoloModel implementa i metodi per aggiungere o rimuovere un osservatore del modello, rispettivamente visibili alla riga 28 e 34 del codice sottostante, e il metodo per notificare i cambiamenti di stato, (vd. riga 40).
Model permette anche la modifica dello stato, la quale sarà effettuata da parte

del controller, tramite un metodo per il settaggio dello stato, chiamato setState (riga 15), e analogamente permette di richiedere lo stato attuale tramite un metodo getState, il quale sarà utilizzato dalla view nel momento in cui viene informata di un cambiamento di stato (riga 21). Di seguito si riporta il codice relativo all'HoloModel.

```
1  class HoloModel : ObservableModel<HoloView>
2      {
3          //stato dell'oggetto
4          private Mood state;
5          //lista di osservatori dei cambiamenti
6          private List<HoloView> observers = new List<HoloView>();
7
8          public HoloModel()
9          {
10             this.state = Mood.Hungry;
11         }
12
13         /// Setter dello stato dell'oggetto
14         /// Al cambiamento di stato, vengono informati gli
               osservatori.
15         public void setState(Mood state)
16         {
17             this.state = state;
18             Notify();
19         }
20
21         /// Getter dello stato dell'oggetto.
22         public Mood getState()
23         {
24             return this.state;
25         }
26
27         /// Aggiunge un osservatore alla lista.
28         public void addObserver(HoloView observer)
29         {
30             observers.Add(observer);
31         }
32
33         /// Rimuove un osservatore dalla lista.
34         public void removeObserver(HoloView observer)
35         {
36             observers.Remove(observer);
37         }
```

```
38
39          /// Per ogni osservatore registrato, viene aggiornato,
                passandogli il nuovo stato dell'oggetto
40          public void Notify()
41          {
42              foreach(HoloView s in observers)
43              {
44                  s.Update();
45              }
46          }
47      }
```

4.3.2 HoloView

Per quanto riguarda la parte di view, gestisce le informazioni relative al disegno del modello sotto forma di ologramma, per fare ciò utilizza le librerie di Direct3D di grafica tridimensionale. All'interno della classe HoloView, come verificabile dal diagramma uml precedente, si tiene traccia di tutte le proprietà grafiche quali la posizione dell'ologramma, i vertici del cubo, il colore, e la modalità di rotazione. Dato che la view implementa le interfacce ModelObserver per essere osservatrice di HoloModel, e ObservableView, per essere osservabile dal controller, al suo interno sono implementati anche i metodi di queste due interfacce. In questo caso è possibile creare una view relativa al modello, specificando direttamente la posizione dell'ologramma e il modello al quale è riferita (riga 1), oppure specificare solo il modello al quale si riferisce, che la posizione verrà impostata automaticamente a due metri davanti all'utente (riga 10).

```
1  public HoloView(HoloModel obj, Vector3 position)
2  {
3     this.position = position;
4     this.model = obj;
5     this.holoState = this.model.getState();
6     //in base allo stato del modello inizialmente si setta la view
7     setView();
8  }
9  /// Nel caso non venga specificata la posizione, viene impostata
       automaticamente a due metri.
10 public HoloView(HoloModel model) : this(model, new Vector3(0.0f,
       0.0f, -2.0f))
11 {
12 }
```

E' bene ricordare che la posizione è specificata in un vettore, il quale in ordine specifica lo scostamento sull'asse x, y, e z, in questo caso il cubo viene scostato solamente due metri nella parte negativa dell'asse z. Dato che Hololens usa un sistema di coordinate rigth-ended in cui l'asse z punta all'indietro, lo scostamento verso la parte negativa dell'asse z, significa che l'ologramma sarà posizionato in avanti di due metri rispetto all'utente.
Per la ricezione degli aggiornamenti quando avviene un cambiamento di stato nel HoloModel, la view implementa il metodo Update dell'interfaccia ModelObserver, come specificato di seguito.

```
1       public void Update()
2       {
3           this.holoState = this.model.getState();
4           setView();
5
6       }
```

Questo metodo viene richiamato dal Notify di HoloModel per ogni view registrata, la view procederà richiedendo informazioni sullo stato del model e di conseguenza aggiorna la rappresentazione grafica dello stesso.
Sono implementati anche i metodi per essere osservabile dal controller, per l'aggiunta e la rimozione di osservatori della view, i metodi sono analoghi a quelli implementati nel model, solo che in questo caso si aggiungono elementi di tipo HoloController, e non di tipo HoloView come nel modello. Gli osservatori vengono salvati in una lista contenente elementi di tipo HoloController. Quando, alla rilevazione di un input, è richiamato il metodo NotifyEvent della view, ad ogni controller della lista viene comunicato il tipo di evento catturato.

```
1       public void NotifyEvent(HoloEvent ev)
2       {
3           foreach (HoloController s in observers)
4           {
5               s.receiveEvent(ev);
6           }
7       }
```

La parte grafica, come detto in precedenza viene realizzata e settata in base ai cambiamenti dello stato, come si nota nel metodo Update, ad ogni aggiornamento dello stato si richiama il metodo setView, per settare le raffigurazioni grafiche di conseguenza. Ogni stato d'animo è comunque rappresentato da un cubo, il quale cambierà colore e modalità di rotazione rispecchiando lo stato d'animo della persona.
Di seguito si riporta la porzione di codice relativa ad uno stato d'animo come

esempio.

```
1  private void setHappyView()
2          {
3              this.color = Color.Green;
4              this.spinMode = SpinningMode.Normal;
5              Vector3[] vertices =
6              {
7                  new Vector3(-0.1f, -0.1f, -0.1f),
8                  new Vector3(-0.1f, -0.1f, 0.1f),
9                  new Vector3(-0.1f, 0.1f, -0.1f),
10                 new Vector3(-0.1f, 0.1f, 0.1f),
11                 new Vector3( 0.1f, -0.1f, -0.1f),
12                 new Vector3( 0.1f, -0.1f, 0.1f),
13                 new Vector3( 0.1f, 0.1f, -0.1f),
14                 new Vector3( 0.1f, 0.1f, 0.1f),
15             };
16             this.vertices = vertices;
17         }
```

Viene impostato il colore, in questo caso verde (riga 3), la modalità di rotazione del cubo, normale (riga 4), e i vertici del cubo. Quest'ultimi sono uguali per il cubo felice e per il cubo triste, invece per il cubo arrabbiato la dimensione del cubo raddoppia, quindi i vertici saranno scostati di 0.2 invece che di 0.1, a differenza del cubo affamato il quale avendo bisogno di mangiare, ha dimensioni ridotte dallo 0.1 allo 0.05. Per quanto riguarda la rotazione, nel caso del cubo arrabbiato questa sarà più veloce, e nel caso di cubo affamato, più lenta, fino al cubo triste che non ruota ma mantiene una posizione fissa. Per la definizione delle modalità di rotazione, è stato dichiarato un tipo di dato enum chiamato SpinningMode, il quale può assumere come valori i tipi "None", "Fast", "Normal", "Slow".

4.3.3 HoloController

Il controller implementato riferito alla gestione dell'ologramma, può essere creato indicando il model del quale specifica la logica di controllo. HoloController implementa l'interfaccia ViewObserver, diventando così un osservatore della parte di view. Al suo interno viene quindi implementato il metodo receiveEvent, il quale analizza l'evento catturato dal sistema. Se la gesture effettuata dall'utente è di tipo "Tap", allora il controller cambia lo stato del model, scegliendo fra uno stato d'animo in modo casuale. E' possibile nel codice seguente vedere il metodo in cui si analizza l'evento (riga 9) e il metodo in cui si setta lo stato del model dopo aver analizzato l'evento (riga 19).

```
class HoloController : ViewObserver
    {
        HoloModel holoModel;
        public HoloController(HoloModel holoModel)
        {
            this.holoModel = holoModel;
        }

        public void receiveEvent(HoloEvent ev)
        {
            SpatialGestureSettings detectedGesture;
            detectedGesture = ev.getDetectedGesture();
            if(detectedGesture.Equals(SpatialGestureSettings.Tap))
            {
                changeModelState();
            }
        }

        public void changeModelState()
        {
            Array moods = Enum.GetValues(typeof(Mood));
            Random random = new Random();
            Mood newMood =
                (Mood)moods.GetValue(random.Next(moods.Length));
            this.holoModel.setState(newMood);
        }
    }
```

4.3.4 HoloRenderer

HoloRenderer rappresenta la parte del programma, che si occupa della renderizzazione di un ologramma all'interno dello spazio olografico. E' necessario definire un HoloRenderer per ogni ologramma che si vuole aggiungere alla scena. All'interno della classe vengono combinate le informazioni relative alle risorse del dispositivo e quelle relative alla parte grafica dell'ologramma da renderizzare, cioè ad HoloView. La creazione di un oggetti di tipo HoloRenderer, necessita quindi delle deviceResource, e di una HoloView da renderizzare. Come è stato descritto nella sezione di funzionamento generale di HoloFrameOfMind, un HoloRenderer implementa i metodi Update e Render di ogni ologramma, i quali si occupano degli aggiornamenti dello stesso e della sua renderizzazione.

In questo caso specifico si riporta il codice del metodo Update: metodo di ogni HoloRenderer richiamato da HoloManager ogni volta che l'applicazione crea un frame da renderizzare.

```
1  public void Update(StepTimer timer)
2          {
3              Matrix4x4 modelTranslation =
                   Matrix4x4.CreateTranslation(this.hologram.Position);
4
5              SpinningMode mode = this.hologram.getSpinningMode();
6              switch (mode)
7              {
8                  case SpinningMode.Fast:
9                      this.degreesPerSecond = 180.0f;
10                     break;
11                 case SpinningMode.Slow:
12                     this.degreesPerSecond = 15.0f;
13                     break;
14                 case SpinningMode.Normal:
15                     this.degreesPerSecond = 60.0f;
16                     break;
17                 default:
18                     this.degreesPerSecond = 0.0f;
19                     break;
20             }
21             float radiansPerSecond = this.degreesPerSecond *
                   ((float)Math.PI / 180.0f);
22             double totalRotation = timer.TotalSeconds *
                   radiansPerSecond;
23             float radians =
                   (float)System.Math.IEEERemainder(totalRotation, 2 *
                   Math.PI);
24             Matrix4x4 modelRotation =
                   Matrix4x4.CreateFromAxisAngle(new Vector3(0, 1, 0),
                   -radians);
25             modelTranslation = modelRotation * modelTranslation;
26
27             this.modelConstantBufferData.model =
                   Matrix4x4.Transpose(modelTranslation);
28
29             if (!loadingComplete)
30             {
31                 return;
32             }
```

```
33
34          var context = this.deviceResources.D3DDeviceContext;
35
36          context.UpdateSubresource(ref
                this.modelConstantBufferData,
                this.modelConstantBuffer);
37      }
```

Inizialmente si setta la posizione dell'ologramma (riga 3), e successivamente (riga 5) in base alla modalità di rotazione definita nella view relativa si setta la variabile che indica di quanti gradi verrà ruotato l'ologramma. L'oggetto chiamato "hologram" è la view riferita all'ologramma che si sta renderizzando. Le righe seguenti servono a settare la rotazione dell'ologramma ruotato, in relazione alla sua posizione di base, e vengono aggiornati i buffer contenenti i dati della posizione dell'ologramma, i quali poi verranno utilizzati per renderizzare effettivamente l'ologramma stesso.
Nel metodo Render la prima cosa effettuata è il riempimento del buffer dei vertici, ad ogni vertice viene associata una posizione ed un colore. Il colore è specificato nel tipo di dato enum Color citato in precedenza, e per la renderizzazione è stata perfezionata una classe apposita che converte l'enum in un vettore a tre dimensioni, leggibile dai metodi di renderizzazione di Hololens, il quale specifica il colore nel formato standard RGB. Di seguito si riporta il codice relativo al riempimento del buffer contenente i vertici del cubo.

```
1  private void FillVertexBuffer()
2          {
3              ICollection<Vector3> vertices = this.hologram.getVertices;
4              VertexPositionColor[] cubeVertices = new
                   VertexPositionColor[vertices.Count];
5              int index = 0;
6              foreach (Vector3 vertex in vertices)
7                  cubeVertices[index++] = new
                       VertexPositionColor(vertex,
                       RGBColors.ConvertColorInRGB(hologram.getColor()));
8
9              this.vertexBuffer =
                   this.ToDispose(SharpDX.Direct3D11.Buffer.Create(
10                 deviceResources.D3DDevice,
11                 SharpDX.Direct3D11.BindFlags.VertexBuffer,
12                 cubeVertices));
13         }
```

Nel dettaglio viene inizialmente richiesto alla view l'insieme di vertici che for-

mano il cubo. Di conseguenza, ogni vertice, specificato dalla posizione, viene completato col colore che esso dovrà assumere, utilizzando la classe di conversione del colore descritta in precedenza (riga 7). Successivamente viene riempito il buffer dei vettori usato per la renderizzazione (riga 9).
Dopo aver eseguito questa operazione, all'interno del metodo render, si analizzano i vertici ottenuti, per concludere disegnando l'oggetto da rappresentare.

4.3.5 HoloManager

Questa classe è il punto di collegamento tra l'applicazione e gli ologrammi presenti nell'applicazione. Essa infatti viene richiamata dalla parte dell'applicazione che gestisce la creazione del frame e la sua renderizzazione, e si occupa della gestione degli ologrammi stessi. All'interno della classe vengono gestiti degli oggetti di tipo HoloRenderer, quindi dovrà essere presente una collezione di oggetti di questo tipo. All'interno di HoloManager sono implementati i metodi Update, per aggiornare tutti gli ologrammi presenti nella scena, e il metodo Render, che richiama il metodo Render di ogni ologramma aggiunto allo spazio olografico.
Ad ogni aggiornamento degli ologrammi, HoloManager si occupa del controllo degli input, eventualmente eseguiti sugli ologrammi, nel caso in cui questo venga rilevato, viene informata la view che è stato rilevato un evento su quell'ologramma. Gli input in Hololens sono catturati dallo SpatialInputHandler, che come dichiarato in precedenza, si occupa di rilevare i movimenti della mano con un SpatialInteractionManager, ed interpretarli grazie ad uno SpatialGestureRecognizer. All'interno di HoloManager sarà presente quindi un'istanza di SpatialInputHandler per avere accesso alle informazioni di rilevazione dell'input. Di seguito si riporta il codice del metodo Update di HoloManager.

```
1  public void Update(StepTimer timer)
2          {
3              if (this.inputHandler.isInteractionOccurred())
4              {
5                  SpatialGestureSettings gestureType =
                       inputHandler.Interaction.First;
6                  SpatialPointerPose pointerPose =
                       inputHandler.Interaction.Second;
7
8                  SharpDX.Ray gaze = new SharpDX.Ray
9                 (this.Vector3ToSharpDX(pointerPose.Head.Position),
10             this.Vector3ToSharpDX(pointerPose.Head.ForwardDirection));
11
12                 HoloView gazedHologram;
```

```
13              if (this.IntersectHologram(gaze, out gazedHologram))
14              {
15                  gazedHologram.NotifyEvent(new
                        HoloEvent(gestureType));
16              }
17
18          }
19          foreach (HoloRenderer hologram in this.hologramsQueue)
20              hologram.Update(timer);
21      }
```

Viene controllato inizialmente che non vi sia un input rilevato, quindi si richiede all'inputHandler se questo è stato catturato o meno. Nel caso lo fosse, grazie ai metodi getter e setter relativi all'interazione, si ricava il tipo di gesture rilevata (riga 5) e uno spatial pointer pose che rappresenta il gaze nel momento in cui è stata rilevata la gesture (riga 6).

Successivamente si crea, grazie alla libreria SharpDX, un raggio, combinando la posizione e la direzione del gaze in quel momento. Questo raggio sarà utilizzato nel metodo IntersectHologram, il quale esegue il raycasting lungo il gaze, per vedere se e quale ologramma è stato intersecato. All'interno del metodo Update, si richiama tale metodo, il quale se rileva un'intersezione, indica su quale tipo di View è stata eseguita (riga 12), e viene richiamato il metodo della view relativo per avviare il processo di aggiornamento dell'ologramma. Dopo gli aggiornamenti relativi agli input, vengono eseguiti gli aggiornamenti propri dell'ologramma, richiamando di ognuno il relativo metodo Update (riga 18).
Per il raycasting viene utilizzato il metodo IntersectHologram, del quale di seguito si riporta l'implementazione relativa.

```
1   public bool IntersectHologram(SharpDX.Ray gaze, out HoloView
        gazedHologram)
2       {
3           gazedHologram = null;
4           float minimumDistance = float.MaxValue;
5           float distance;
6           //per ogni ologramma presente nella scena..
7           foreach (HoloRenderer hologramRenderer in
                this.hologramsQueue)
8           {
9               //..si ottiene la sua view..
10              HoloView hologram = hologramRenderer.Hologram();
11
12              //.. e si crea un bounding box
```

```
13              SharpDX.BoundingBox boundingHologram = new
                    SharpDX.BoundingBox(
14              this.Vector3ToSharpDX(hologram.getVertices.First()
                    + hologram.Position),
15              this.Vector3ToSharpDX(hologram.getVertices.Last()
                    + hologram.Position));
16
17              //se la esiste un intersezione fra il raggio del gaze
                    e il bounding box
18              //viene restituita la distanza
19              if (SharpDX.Collision.RayIntersectsBox(ref gaze, ref
                    boundingHologram, out distance))
20              {
21                  //la distanza piu' piccola e' associata
                        all'ologramma piu' vicino
22                  //quindi si setta l'ologramma da restituire
23                  if (distance < minimumDistance)
24                  {
25                      minimumDistance = distance;
26                      gazedHologram = hologram;
27                  }
28              }
29          }
30          //se e' stato trovato l'ologramma, viene restituito
31          return (gazedHologram != null);
32      }
```

In questo metodo viene effettuato il raycasting lungo il gaze rilevato al momento della gesture, per vedere se viene intersecato un ologramma o meno. Quindi per ogni ologramma presente nella scena (riga 7), si ottiene la view al quale ci si riferisce (riga 10), e utilizzando i vertici specificati all'interno della view, viene creato bounding box. Un bounding box non è altro che un contenitore, il quale incapsula totalmente a livello immaginario l'oggetto rappresentato. Grazie alla libreria SharpDX, si utilizza la funzione RayIntersectBox (riga 19), per verificare se il raggio creato in precedenza lungo il gaze, interseca il box creato attorno all'ologramma. Nel caso fosse intersecato il bounding box, si ricava la distanza, fornita sempre dal metodo precedente, e alla fine la distanza minima calcolata fra le varie intersezioni, rappresenterà l'ologramma più vicino. A questo punto il metodo restituisce la view relativa all'ologramma che è stato intersecato (riga 31).
In HoloManager sono presenti anche i metodi per aggiungere un ologramma, cioè un HoloRenderer, alla collezione (riga 9), per rimuoverlo (riga 14), e per cancellare ogni elemento dalla collezione e quindi anche dalla scena

olografica (riga 19). Di seguito si riportano i metodi appena citati, unitamente al costruttore della classe HoloManager (riga 3), mostrandone la loro implementazione.

```
1        //collezione che include tutti gli ologrammi della scena
2        private ICollection<HoloRenderer> hologramsQueue;
3        public HoloManager()
4        {
5            this.hologramsQueue = new List<HoloRenderer>();
6            this.inputHandler = new SpatialInputHandler();
7        }
8        /// Metodo per aggiungere un ologramma alla scena.
9        public void AddHologram(HoloRenderer hologram)
10       {
11           this.hologramsQueue.Add(hologram);
12       }
13       /// Metodo per rimuovere un ologramma dalla scena.
14       public void RemoveHologram(HoloRenderer hologram)
15       {
16           this.hologramsQueue.Remove(hologram);
17       }
18       /// Permette la cancellazione di ogni elemento nella coda.
19       public void Clear()
20       {
21           this.hologramsQueue.Clear();
22       }
```

La creazione delle varie parti del programma, va effettuata all'interno del main dell'applicazione. Si dovrà creare inizialmente un manager degli ologrammi, cioè un istanza di HoloManager (riga 1), e poi per ogni ologramma da aggiungere alla scena si deve creare il rispettivo model (riga 3), la view (riga 5) e il controller (riga 4), unitamente alla creazione del HoloRenderer da aggiungere alla collezione in HoloManager (riga 9). Alla riga 6 e 7, la view e il controller vengono aggiunti come osservatori rispettivamente di model e view. Nel codice successivo viene mostrata l'implementazione di queste funzioni.

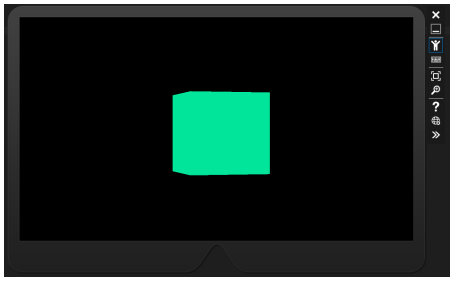
```
1            this.manager = new HoloManager();
2
3            holoModel = new HoloModel();
4            holoController = new HoloController(holoModel);
5            holoView = new HoloView(holoModel);
6            holoModel.addObserver(holoView);
7            holoView.addObserver(holoController);
8
```

```
9           this.manager.AddHologram(new
                HoloRenderer(this.deviceResources, holoView));
```

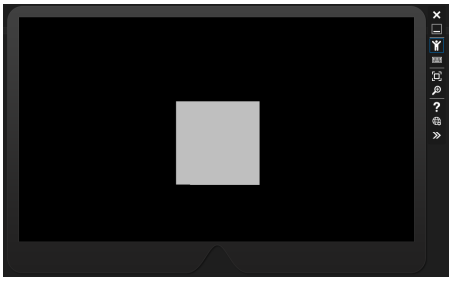
Nel caso di questa applicazione, viene aggiunto un solo ologramma, rappresentante lo stato d'animo di una persona.

4.3.6 Overview utente

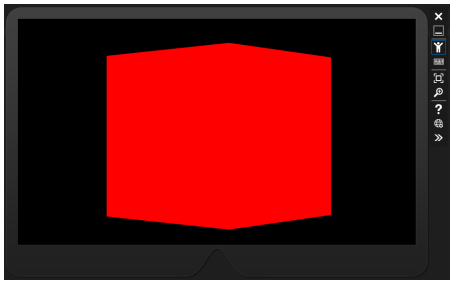
L'applicazione inizialmente sceglie uno stato d'animo random, non essendo collegata in modo effettivo con una persona ma essendo solo un'ipotetica rappresentazione. L'utente potrà cliccare sull'ologramma per cambiare lo stato d'animo, il quale verrà scelto come spiegato in precedenza in modo casuale fra tutti quelli ipotizzati in Mood. Di seguito, a conclusione, sono riportati gli screenshot relativi ad ogni stato d'animo raffigurato nell'applicazione.



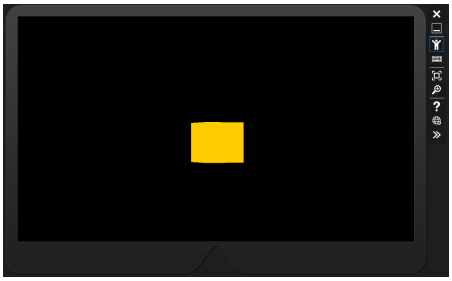
(a) Stato d'animo: Happy.



(b) Stato d'animo: Sad.



(c) Stato d'animo: Angry.



(d) Stato d'animo: Hungry.

Figura 4.3: Didascalia globale

La gesture riconosciuta per effettuare la transizione da uno stato all'altro è quella di "Tap" che dovrà essere effettuata solo sull'ologramma, il sistema sarà ovviamente in grado di rilevarla anche se viene eseguita fuori dall'ologramma, ma la parte di gestione degli ologrammi HoloManager, non troverà corrispondenza fra la direzione del gaze e l'ologramma e quindi non procederà al cambiamento di stato. Questa applicazione, di base, esprime i concetti generici di questa tecnologia e può essere utilizzata come base per future estensioni

della stessa, magari collegando la parte di modello e/o di controller ad agenti esterni in modo che l'ologramma rappresenti veramente una persona. Sarebbe suggestivo indossare il dispositivo Hololens e visualizzare un ologramma sul tavolo il quale rappresenta lo stato d'animo di una persona a noi conosciuta che magari in quel momento è fuori casa.

Conclusioni

La possibilità di studiare questo ramo dell'informatica, relativo ai dispositivi che implementano augmented e mixed reality, oltre ad aver arricchito le mie conoscenze personali, mi ha permesso di toccare con mano una piccola parte di quella che probabilmente sarà una notevole parte della tecnologia futura. Tale possibilità, unitamente al fatto di studiare e svolgere ricerche su un dispositivo al momento non ancora in commercio e quindi sconosciuto alla maggior parte delle persone, è stimolante. Studiando dispositivi nuovi e progettando applicazioni per gli stessi, date le poche documentazioni e conoscenze disponibili al riguardo, risulta essere certamente più difficile ma allo stesso tempo, a risultato ottenuto, è altamente gratificante. L'essere entrati all'interno del mondo della mixed reality, mi ha permesso di ragionare sulle possibili necessità delle persone, fino a produrre una base di applicazione per Hololens. Lo studio della tecnologia essendo al momento ancora in fase di sviluppo non è stato subito immediato, ma dal momento in cui si riesce a padroneggiare gli elementi che la costituiscono, sono riuscito a combinarli con elementi di progettazione, per rendere in futuro, la creazione di applicazioni, più semplice. Il lavoro svolto ha raggiunto gli obiettivi prefissati inizialmente, e può essere usato come documento d'accesso alla tecnologia, o in alternativa come base per espansione di lavori futuri, come spiegato nella sezione successiva. In ogni caso, penso che il lavoro eseguito sia un'indagine esaustiva per chiunque voglia affacciarsi al dispositivo Microsoft Hololens, e iniziare lo sviluppo di applicazioni per questo dispositivo, cioè applicazioni olografiche.

Sviluppi Futuri

La riprogettazione del concetto di ologramma, unitamente all'idea di rappresentare lo stato d'animo di una persona sotto forma di ologramma, può essere certamente esteso in varie direzioni, amplificando lo stesso progetto e rendendolo concreto quando il dispositivo sarà disponibile al pubblico, oppure considerare la parte di progettazione degli ologrammi per sviluppare nuove idee di nuove applicazioni.

Certamente la parte di gestione degli ologrammi da parte dell'applicazione e la progettazione degli ologrammi stessi secondo lo schema specificato può essere utile ad altre persone che vogliano sviluppare applicazioni per Hololens, come schema di base già pronto su cui continuare la progettazione ed estenderla fino al completamento di una nuova applicazione.
Per quanto riguarda l'applicazione HoloFrameOfMind, sarebbe interessante testarla sul dispositivo vero e proprio quando sarà disponibile, ed ampliarla con varie funzionalità. La prima sarebbe la rappresentazione grafica dello stato d'animo, il quale potrebbe essere rappresentato con un cuore, o assumere diverse forme in base allo stato d'animo stesso. Oltre alla parte grafica sarebbe interessante, se non necessario, che l'ologramma fosse "collegato" ad una persona vera, cioè rappresentasse veramente lo stato d'animo di una persona, e si aggiornasse di conseguenza. La persona al quale si riferisce potrebbe definire il suo stato d'animo tramite un ologramma, e questo potrebbe venire condiviso ad altri dispositivi usando la medesima applicazione, in modo che le persone abilitate a conoscere lo stato d'animo, possano vederlo ed informarsi in tempo reale. Oppure la persona potrebbe inserire il suo stato d'animo su un dispositivo wearable, o su un'app per smartphone, e le due parti potrebbero comunicare raffigurando poi lo stato d'animo come ologramma. Insomma, rendere effettivamente collegato lo stato d'animo alla persona, e quindi rendere la parte relativa allo stato dell'ologramma, modificabile anche da entità esterne all'applicazione stessa. Una base di applicazione del genere potrebbe essere poi usata anche in ambito medico, rappresentando le condizioni di ogni paziente di un reparto sotto forma di ologramma, in modo che il medico, nel caso sia nel suo studio, o fuori dall'ospedale, indossando Hololens, possa informarsi sullo stato delle persone che segue. Sarebbe veramente interessante continuare lo sviluppo dell'idea dell'applicazione in questa direzione.

Ringraziamenti

A conclusione di questo primo percorso universitario, nonostante questo verrà proseguito negli anni successivi, penso sia doveroso ringraziare i principali contribuenti a questo lavoro e al raggiungimento del conseguimento di questo titolo universitario.
In primis, ringrazio il Prof. Alessandro Ricci, per avermi permesso di svolgere questo tipo di lavoro nell'ambito della sua materia di sviluppo, unitamente all'Ing. Angelo Croatti, i quali hanno fornito un supporto decisionale e sono stati fonte di conoscenze al fine di svolgere il lavoro di tesi.
Successivamente, la mamma, e tutta la famiglia, che mi ha dato un supporto a 360 gradi in questi tre anni, con lo scopo di tenere sempre il piede sull'acceleratore e non perdersi troppo in cose superflue.
Ringrazierei anche gli amici e i compagni di corso, fonte di svago dai momenti doverosi di studio e di apprendimento, ma necessari anche loro per compensare gli sforzi effettuati.
Infine ringrazio ogni persona incontrata in questi tre anni, in quanto ogni incontro non è casuale e mi ha sicuramente arricchito umanamente.

Bibliografia

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

[2] Satya Nadella, Terry Myerson, Joe Belfiore, Phil Spencer, Alex Kipman, *Windows 10 Briefing*, http://news.microsoft.com/speeches/satya-nadella-terry-myerson\\-joe-belfiore-and-phil-spencer-windows-10-briefing/ Microsoft News Center, 2015.

[3] R. Azuma, *A survey of Augmented Reality*, 1997.

[4] Microsoft, *Spatial Mapping in DirectX*, https://developer.microsoft.com/en-us/windows/holographic/spatial_mapping_in_directx, 2016.

[5] D.W.F. Vankrevelen, *A surver of AR technologies, applications and limitations*, 2010.

[6] C. Arth, *The History of Mobile Augmented Reality*, 2015.

[7] P. Milgram, F. Kishino, *A taxonomy of mixed reality virtual displays*, 1994.

[8] A. Ricci, A. Croatti, *Programming Abstractions for Augmented World* 2016.

[9] A. Kunx, E. Costanza, M. Fjeld. *Mixed reality: A Survey*, 2009.

[10] I. Sutherland, *A head mounted three dimensional display*, 1968.

[11] E. Moon, M. Kim, J. Roh, H. Kim, J. Hahn *Holographic head-mounted display with RGB light emmitting diode light source*, 2014.

[12] Meta Company, *Official Website*, https://www.metavision.com/meet-meta 2016.

[13] Microsoft, *Coordinate System in DirectX*, https://developer.microsoft.com/en-us/windows/holographic/coordinate_systems_in_directx, 2016.

[14] Microsoft, *Gaze and Gesture in DirectX*, https://developer.microsoft.com/en-us/windows/holographic/gaze_and_gestures_in_directx, 2016.

[15] Microsoft, *Windows Holographic Documentation*, https://developer.microsoft.com/en-us/windows/holographic/documentation, 2016.

[16] Microsoft, *Hololens Hardware Details*, https://developer.microsoft.com/en-us/windows/holographic/hardware_details, 2016.