ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

STUDIO E PROTOTIPAZIONE DI UN'ARCHITETTURA SOFTWARE DI ALTO LIVELLO PER LO SVILUPPO DI APPLICAZIONI DI MIXED REALITY SU PIATTAFORMA HOLOLENS

Elaborato in
PROGRAMMAZIONE DI SISTEMI EMBEDDED

Relatore
Prof. ALESSANDRO RICCI

Co-relatore
Ing. ANGELO CROATTI

Presentata da
MANUEL PERUZZI

Prima Sessione di Laurea
Anno Accademico 2015 – 2016

PAROLE CHIAVE

HoloLens

Mixed Reality

Ologramma

Augmented World

Indice

Introduzione

Lo sviluppo tecnologico è in grado di cambiare completamente le nostre vite, rivoluzionandone ogni tipo di aspetto. Pensiamo ad esempio all'introduzione degli smartphone, che in breve tempo sono passati dall'essere considerati un oggetto opzionale fino a rappresentare una parte fondamentale della nostra esperienza quotidiana. Un percorso simile può essere intrapreso dai dispositivi di realtà aumentata.

Quante volte guardando un film di fantascienza siamo stati catturati dalla raffigurazione di ologrammi? La possibilità di creare oggetti composti interamente di luce ed in grado di sviluppare un proprio comportamento ha affascinato l'uomo fin dagli albori dello sviluppo tecnologico. Nonostante ciò abbiamo sempre avvertito gli ologrammi come qualcosa di difficilmente applicabile, qualcosa che sarà possibile utilizzare in un futuro indefinito. Quel futuro, che ci appare ancora così lontano, è in realtà dietro l'angolo.

Le più importanti aziende d'informatica a livello globale sono, da anni, concentrate sulla realizzazione di questi sistemi e il duro lavoro effettuato sta, recentemente, producendo i primi frutti. Sono disponibili sul mercato vari dispositivi di realtà aumentata, che permettono all'utente di estendere la sua percezione attraverso l'integrazione di informazioni digitali nel mondo reale. Mi riferisco ad esempio ai più comuni smartglasses, strumenti rivoluzionari, che però non esprimono il concetto di ologramma tanto atteso nel panorama tecnologico.

La svolta è stata avviata da Microsoft, tramite l'introduzione della tecnologia HoloLens. Il dispositivo relativo si presenta come un semplice caschetto, comodamente indossabile, in grado di generare ologrammi e di collocarli nel mondo reale intorno all'utente. HoloLens esprime le potenzialità per introdurre il concetto di ologramma nella vita quotidiana, rivoluzionando completamente il modo di svolgere le azioni di tutti i giorni.

Questa tesi ha lo scopo di fornire un'esplorazione del mondo HoloLens, in modo da comprendere in che modo questa innovativa tecnologia possa diventare parte della vita comune.

Nella prima parte dell'elaborato è necessario inquadrare HoloLens nel contesto informatico di competenza, soffermandosi inizialmente sulle tecnologie

che individuano l'interazione di elementi reali e virtuali, appartenenti alla categoria di mixed reality. Dopo aver effettuato una prima analisi dell'ambito generale, è possibile immergersi nella perlustrazione degli aspetti chiave individuati dal HoloLens, prestando particolare attenzione allo sviluppo di applicazioni olografiche.

La seconda parte dell'elaborato consiste nella realizzazione di un progetto, suddiviso in due fasi, che ha lo scopo di risolvere le principali criticità incontrate nel processo di sviluppo di un'applicazione HoloLens. Nella prima fase viene effettuata la progettazione e la realizzazione di una serie di API che possano favorire lo sviluppo di applicazioni HoloLens, attraverso l'introduzione di un livello di astrazione. Nella seconda fase viene definita ulteriormente la necessità di fornire ai futuri sviluppatori una gestione facilitata degli ologrammi, attraverso un ulteriore livello di astrazione. Si tratta, in questo caso, dell'introduzione del concetto di mondo aumentato, definito come un sistema nel quale sono presenti entità virtuali dotate di una certa posizione e di uno stato ben definito, accessibile da una vasta gamma di dispositivi diversi, fra cui HoloLens.

Infine sono trattate le considerazioni finali sul lavoro svolto e i possibili sviluppi futuri derivati della realizzazione del progetto.

Capitolo 1

Mixed Reality, un'integrazione di elementi reali e virtuali

Al giorno d'oggi in un sistema informatico ci si trova spesso a dover gestire sia elementi di un ambiente reale, sia oggetti virtuali. Il mondo reale e il mondo virtuale non sono più così distinti, ma possono intersecarsi per creare nuove realtà alternative, delle cosiddette mixed reality. La relazione fra reality e virtuality è in continuo aggiornamento e costituisce un argomento di studio interessante per comprendere i principi fondanti di tecnologie importanti come augmented reality e virtual reality.

In questo capitolo saranno esaminate le tecnologie definite in ambito reale e virtuale, con un'attenzione particolare riguardo alla mixed reality.

1.1 Reality-virtuality continuum

La mixed reality, definita anche come hybrid reality, è un argomento che sta diventando sempre più ricorrente al giorno d'oggi, senza che ne venga fornita una definizione specifica. Infatti con il concetto di mixed reality si intende, in modo generale, un'integrazione della percezione sensoriale dell'utente attraverso degli strumenti digitali.

Per comprendere adeguatamente il significato di questo termine è necessario esplorare la relazione fra mondo reale e mondo virtuale. Queste due entità appaiono molto distanti fra loro, eppure dalla loro intersezione si creano realtà diverse, ognuna con una sua caratteristica distintiva. Ciascuna di queste può essere generalmente indicata come una mixed reality, in quanto contiene sia elementi del mondo reale, sia elementi del mondo virtuale.

La teoria appena illustrata costituisce il principio fondante del reality-virtuality continuum elaborato da Milgram, Takamura, Utsumi e Kishino nel 1994.

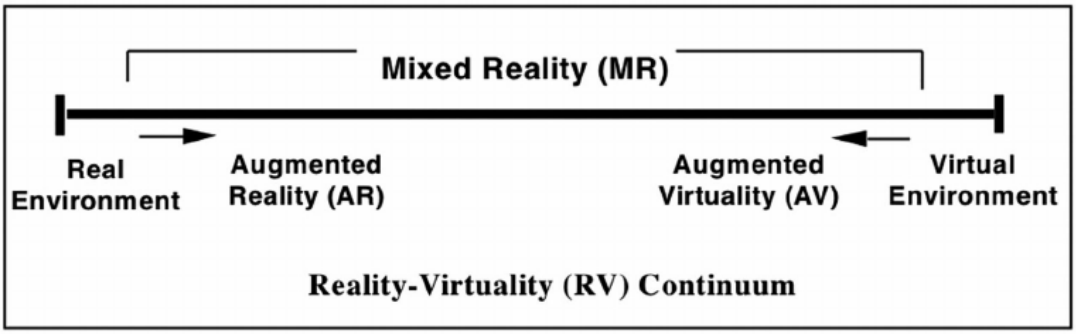

Figura 1.1: Reality-virtuality continuum

In Figura 1.1 si può notare la rappresentazione schematica della linea di mixed reality[1]. Vengono esaminati i concetti che permettono di comprendere le varie tipologie di casi possibili:

- ambiente reale,
- mixed reality, a sua volta suddivisibile in
  - augmented reality,
  - augmented virtuality,
- virtual reality.

Il caso situato all'estrema sinistra del continuum definisce un semplice ambiente nel quale sono presenti oggetti fisici e reali, che possono essere osservati dal vivo da una persona oppure essere visualizzati tramite un display. Dalla parte opposta troviamo invece un ambiente completamente virtuale costruito tramite delle simulazioni grafiche da un computer. Si parla in questo caso di virtual reality.

La concezione di mixed reality si pone nel segmento compreso fra questi due punti estremi e può assumere diverse sfaccettature. Vengono esaminate due tipologie principali di mixed reality, si tratta di augmented reality e augmented virtuality. Queste due categorie si fondano su un principio base condiviso: estendere la percezione visiva dell'utente attraverso l'introduzione di elementi virtuali. Sulle modalità di esecuzione e sugli obiettivi troviamo invece delle discrepanze che ci permettono di effettuare tale classificazione.

Facendo sempre riferimento allo schema in Figura 1.1, è possibile collocare augmented reality come una tecnologia che si fonda su una base di partenza costituita da un ambiente reale, nel quale vengono introdotti degli elementi virtuali che permettono l'aggiunta di alcune informazioni elaborate da un dispositivo digitale.

Se l'augmented reality può essere rappresentata come una freccia, con l'origine in un ambiente reale, che punta verso un ambiente virtuale, è possibile identificare l'augmented virtuality come una freccia che compie il cammino opposto. In questo caso l'ambiente di sfondo è costruito graficamente da un computer, realizzando un contesto esclusivamente virtuale, nel quale sono poste delle componenti reali.

Ogni tecnologia, rappresentabile come un punto situato fra real environment e virtual environment, può essere considerata appartenente alla categoria di mixed reality. Vi è dunque un vasto range nel quale è possibile delineare tecnologie diverse di mixed reaality, che possono essere identificate dall'equilibrio di entità del mondo reale e concetti di un'esperienza virtuale.

Applicando questa direttiva è possibile classificare un altro tipo di mixed reality, che, sulla linea mostrata in Figura 1.1, si può immaginare come un punto rappresentato nello spazio intermedio fra augmented reality e augmented virtuality. Si tratta di un'evoluzione della realtà aumentata che sta acquisendo uno sviluppo sempre maggiore nel mondo informatico e si fonda sulla rappresentazione di ologrammi nel mondo reale.

1.2 Ambiente virtuale

Un virtual environment è un ambiente tridimensionale generato mediante l'utilizzo di un computer, in modo da renderlo esplorabile per una persona. Si introduce quindi un mondo virtuale nel quale l'utente viene completamente immerso e con il quale può interagire, intraprendendo una serie di azioni predefinite.

Se l'ambiente virtuale contiene esclusivamente entità virtuali si parla di virtual reality, se altrimenti sono presenti anche elementi reali ci si riferisce al concetto di augmented virtuality.

1.2.1 Virtual reality

Il termine "virtual reality" è stato coniato nel 1989 da Jaron Lanier, fondatore della VPL Research, e originariamente si limitava ad una "immersive virtual reality", ovvero un'esperienza virtuale nella quale l'utente si trova completamente immerso in un mondo tridimensionale artificiale generato da un computer.

Sistemi di realtà virtuale trovano largo impiego in campo architettonico, nel quale possono essere utili per esplorare edifici in fase di sviluppo, così come in ambito medico, dove possono permettere di simulare un particolare intervento, oltre che per aumentare il realismo nei videogiochi. Il contesto nel

quale acquisiscono un'importanza fondamentale è, però, quello scientifico, nel quale sono presenti processi che possono essere analizzati solo tramite delle simulazioni virtuali[2].

HMD L'utente può essere immerso in un mondo virtuale tramite l'utilizzo di un apposito dispositivo, un head mounted display (HMD), che è incaricato della trasmissione della visuale 3D e dei segnali audio. Gli head mounted display necessitano del supporto di un dispositivo esterno che sia in grado di generare la grafica tridimensionale, spesso di tratta di un PC con elevate prestazioni, mentre in alcuni casi è sufficiente l'utilizzo di uno smartphone. Questa modalità per creare un ambiente virtuale è la più realistica in quanto trasmette all'utente la sensazione di essere completamente immerso in una realtà alternativa a quella reale. Un head mounted display può essere integrato tramite uno strumento che permetta all'utente di interagire con l'ambiente virtuale, come un joystick.

Figura 1.2: Dispositivo Oculus Rift

Un esempio molto diffuso di HMD applicato alla virtual reality è Oculus Rift, dispositivo sviluppato da Oculus VR, mostrato in Figura 1.2. Oculus Rift necessita del collegamento con un computer per compiere il suo funzionamento, svolto con una latenza minima. È caratterizzato da un vasto campo visivo di oltre 90 gradi in orizzontale e da una risoluzione elevata, pari a 2160x1200 nell'ultima versione rilasciata.

CAVE Un'alternativa interessante per creare un ambiente virtuale è fornita dall'utilizzo di un cave automatic virtual environmente, meglio conosciuto con l'acronimo CAVE. Un sistema CAVE consiste in una stanza a forma di cubo provvista di proiettori in ogni muro, compreso il pavimento, che sono in grado di simulare un ambiente virtuale, oltre a vari altoparlanti posti agli angoli della stanza, in modo da creare l'illusione di suoni provenienti da posizioni diverse. L'utente posto all'interno della stanza indossa un visore specifico progettato

per rappresentare delle immagini tridimensionali tramite una tecnica di visualizzazione 3D, nota come stereoscopia. L'esperienza virtuale all'interno della CAVE è regolata da un computer che elabora i dati del sistema di tracciamento dei movimenti dell'utente e produce le immagini in tre dimensioni.

Non immersive VR Il significato di virtual reality con il tempo si è espanso fino ad indicare anche delle applicazioni che non consentono all'utente di essere completamente immerso nel mondo virtuale. In questa categoria possiamo trovare dei sistemi di navigazione relativi ad un ambiente tridimensionale effettuati tramite l'utilizzo del mouse in un semplice PC, oppure dei videogames per varie console di gioco. Nonostante non trasmettano il realismo fornito da un head mounted display, queste applicazioni possono essere inserite nella categoria di virtual reality in quanto consentono all'utente di esplorare un ambiente virtuale, permettendogli anche di interagirvi.

1.2.2 Augmented virtuality

La tecnologia di augmented virtuality condivide tutti i principi della realtà virtuale, considerando però l'introduzione nel mondo artificiale di elementi reali, e può quindi essere definita come una mixed reality. Al contrario della virtual reality non ha raggiunto un grande interesse mediatico, e tuttora gli investimenti delle grandi compagnie informatiche in questo ambito non sono rilevanti. Si preferisce infatti investire su tecnologie più collaudate come la virtual reality e l'augmented reality, che hanno già conseguito risultati importanti.

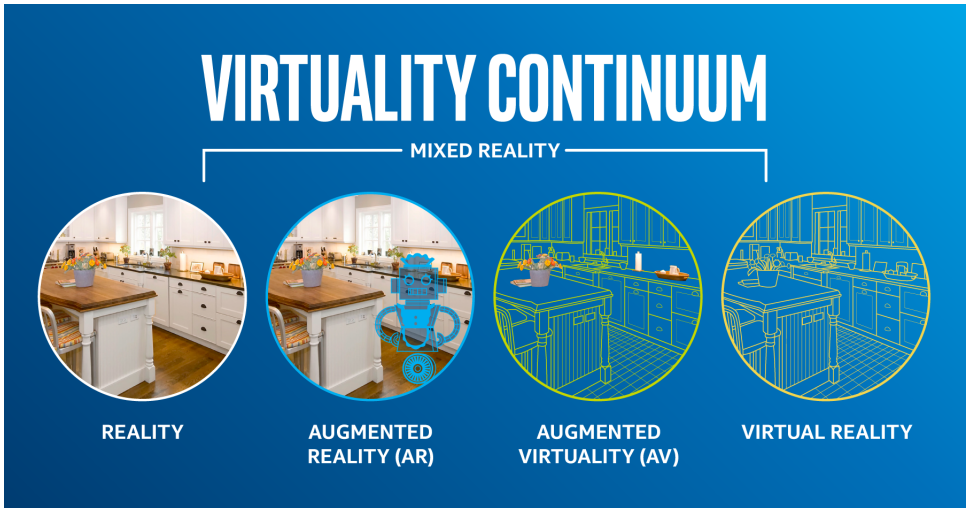

Figura 1.3: Virtuality continuum

In figura 1.3 è mostrato il virtuality continuum, che permette di definire con precisione l'augmented virtuality[3]. Questa tecnologia, che pone le sue basi in un mondo virtuale, si avvale di telecamere e sensori che permettono di identificare degli oggetti del mondo reale e trascinarli nel mondo virtuale generato al computer, permettendo all'utente di interagire con questi oggetti come sarebbe in grado di fare in un ambiente reale.

Una buona percentuale di utenti che utilizzano degli head mounted display per la realtà virtuale hanno espresso alcune problematiche della totale immersione nel mondo virtuale, come ad esempio la necessità di rimuovere temporaneamente il dispositivo per essere in grado di interagire con oggetti reali, quali una semplice tastiera o un bicchiere d'acqua[4].

L'introduzione dei principi di augmented virtuality consentirebbe di includere un numero ristretto di oggetti reali nel mondo virtuale, in modo da permettere all'utente di interagirvi anche quando si trova immerso in un mondo virtuale. Il punto critico dello sviluppo di applicazioni strutturate in questo modo consiste nel determinare il giusto compromesso fra la presenza di oggetti reali e l'ambiente virtuale, in quanto un'introduzione eccessiva di entità reali potrebbe creare un effetto negativo, compromettendo la percezione di immersione totale dell'utente nel mondo artificiale costruito intorno a lui.

Uno degli sviluppi più interessanti di augmented virtuality si pone come obiettivo quello di includere le mani dell'utente nel mondo virtuale in modo da permettergli di visualizzarle e utilizzarle per interagire con gli oggetti virtuali, oltre che con quelli dell'ambiente reale. Nei più diffusi sistemi di virtual reality l'unica possibilità concessa all'utente per interagire con gli elementi dell'ambiente virtuale consiste nell'utilizzo di un apposito dispositivo esterno. L'augmented virtuality si sta muovendo in questa direzione per risolvere tale problema, utilizzando dei sensori di movimento per elaborare gli input effettuati dall'utente tramite delle gesture.

1.3 Ambiente reale, augmented reality

Come si è già discusso in precedenza, il concetto di mixed reality è molto vasto, tanto da comprendere anche una tecnologia come l'augmented virtuality che si fonda su un virtual environment. Il campo dove però trova una maggiore applicazione corrisponde ad un ambiente reale, tramite lo sviluppo di una tecnologia, ampiamente diffusa al giorno d'oggi, quale la realtà aumentata.

Negli ultimi anni si sta verificando un'evoluzione del concetto di augmented reality, con l'introduzione di ologrammi nel mondo reale, dando vita ad una realtà olografica che si avvicina come modalità di utilizzo all'augmented virtuality, mantenendo però la sua applicazione in un ambiente reale.

1.3.1 Descrizione

Augmented reality è una tecnologia di mixed reality che pone le sue basi in un mondo reale, e lo arricchisce attraverso l'integrazione di elementi virtuali. Un sistema per essere considerato appartenente alla categoria di realtà aumentata deve soddisfare tre caratteristiche:

- combinare elementi virtuali nel mondo reale,
- essere definito in un campo tridimensionale,
- essere reattivo in tempo reale.

Questa definizione, coniata da Azuma nel 1997, è sicuramente attuale anche oggigiorno[5]. Dal punto di vista dell'utente, augmented reality, può essere invece definita come un'estensione del mondo reale, uno strumento in grado di aumentare la propria percezione sensoriale attraverso elementi grafici, ma anche, in alcuni casi, uditivi.

Un dispositivo di realtà aumentata, per essere messo in condizione di poter visualizzare degli elementi virtuali nel mondo reale, necessita di un supporto esterno. Il sistema deve essere in grado di comprendere la struttura dell'ambiente circostante e di conoscere la posizione esatta dell'utente in ogni momento. Il tracciamento della posizione dell'utente in uno spazio aperto può essere effettuato mediante l'utilizzo del global positioning system (GPS), coadiuvato da altri sensori quali accelerometro e giroscopio. Mentre se il contesto è quello di uno spazio chiuso è necessario fornire al sistema un modello geometrico 3D dell'ambiente in esame[6].

1.3.2 Campi applicativi

Le interazioni con oggetti virtuali risultano molto meno intuitive rispetto a quelle che interessano oggetti fisici del mondo reale. Uno degli obiettivi principali di augmented reality è limare questa problematica, permettendo di lavorare con entità virtuali in modo estremamente più semplice, facilitando la progettazione di elementi tridimensionali[7].

Oltre a ciò la realtà aumentata permette di effettuare delle simulazioni pratiche tramite visualizzazione di elementi virtuali. Per queste sue caratteristiche, augmented reality trova largo impiego in ambito professionale[5].

Medicina Le implicazioni di augmented reality in campo medico sono varie. Ad esempio tramite l'utilizzo della realtà aumentata è possibile fornire un aiuto notevole ai medici durante un'operazione. Ciò viene effettuato collocando dei sensori non invasivi sul paziente, con i dati raccolti che vengono elaborati e

trasmessi al medico in tempo reale, in modo da fornirgli una sorta di visione a raggi X all'interno del paziente. Inoltre con il supporto della realtà aumentata è possibile indicare delle informazioni o istruzioni, in modo da facilitare il processo di apprendimento di un novizio alle prime armi.

Manutenzione e riparazione Il processo di manutenzione e riparazione di complessi macchinari industriali o informatici può risultare insidioso e necessita dello studio accurato del manuale di funzionamento. Attraverso l'introduzione di dispositivi di realtà aumentata è possibile bypassare questa parte di ricognizione con la visualizzazione direttamente sul display di istruzioni e immagini esplicative.

Progettazione Il processo di progettazione di un oggetto complesso, come ad esempio un robot, può risultare alquanto complicato. Per verificare il corretto comportamento dell'oggetto in esame è possibile realizzarne un prototipo virtuale e simularne le funzionalità, potendo così controllare lo sviluppo tramite l'ausilio della realtà aumentata.

Aviazione militare L'ambito militare è uno dei primi nel quale i sistemi di realtà aumentata sono stati stabilmente impiegati. In particolar modo l'aviazione militare ha introdotto l'utilizzo degli HUD (Head Up Display), dei dispositivi di realtà aumentata che permettono la visualizzazione dei dati di volo, come quota e velocità, direttamente sul display, in modo da evitare al pilota di reperire tali dati in maniera manuale. Oltre agli HUD, sono stati introdotti anche dei HMS (Helmet Mounted Sight), ovvero dei sistemi per il puntamento di bersagli mobili, utilizzati negli aerei militari.

Visualizzazione di informazioni In ambito pubblico l'utilizzo più rilevante di augmented reality consiste nel permettere all'utente di visualizzare delle certe etichette o informazioni poste su determinati oggetti. Ciò può essere effettuato tramite l'ausilio di un database pubblico che contiene tutte le informazioni relative a delle entità reali di interesse. Tramite l'uso di un normale dispositivo mobile o di un head mounted display l'utente può beneficiare di questa grande mole di dati informativi. Un'alternativa consiste nel permettere all'utente di crearsi le proprie informazioni su un determinato elemento, che rimangono, in questo caso, private e a disposizione del solo utente.

1.3.3 L'evoluzione di augmented reality

L'obiettivo originario di augmented reality era quello di fornire delle informazioni in tempo reale all'utente che ne faceva uso. Con lo sviluppo della

tecnologia è possibile riconoscere una branca della realtà aumentata che si è focalizzata sull'obiettivo di disporre degli elementi virtuali nel mondo reale come se gli appartenessero, incrementando in modo rilevante il senso di realismo. Gli elementi virtuali che si approcciano all'utente non sono più considerati come una simulazione di un processo o banalmente un insieme di informazioni, ma possono essere identificati come oggetti appartenenti al mondo reale. L'interazione fra l'utente e tali elementi virtuali riveste un ruolo chiave all'interno di questo progetto.

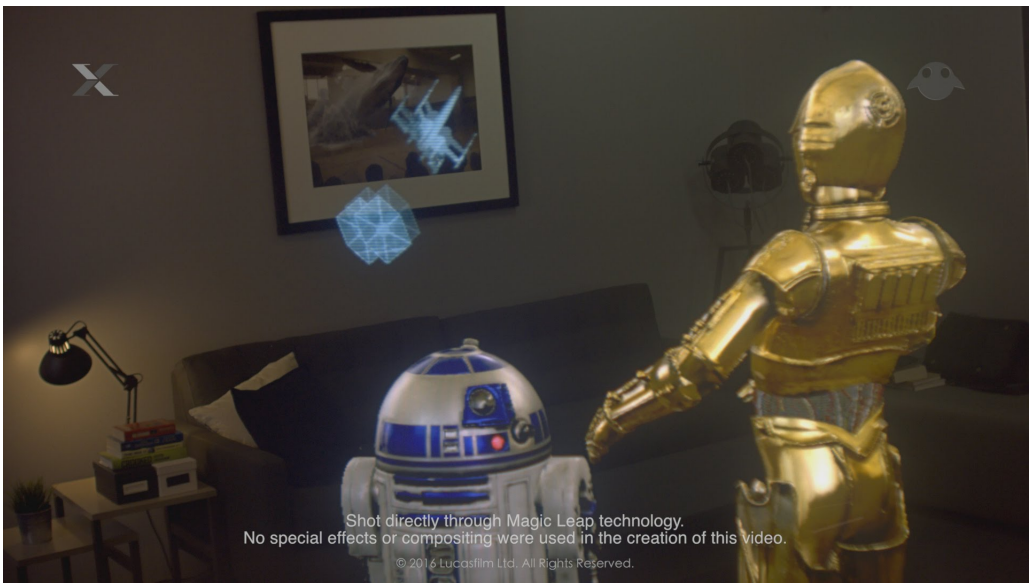

Figura 1.4: Schermata di Magic Leap

In Figura 1.4 viene rappresentata una schermata originale ricavata dal prototipo di un head mounted display prodotto da Magic Leap.

Magic Leap è una startup statunitense, fondata nel 2010 da Rony Abovitz, attiva nel campo della mixed reality. In particolare sta lavorando sulla costruzione di un dispositivo che sia in grado di produrre elementi virtuali ad alta definizione e di sovrapporli al mondo reale. Il team di sviluppo è tuttora impegnato nella creazione del dispositivo, potendo contare sul supporto economico di diversi grandi investitori, tra cui Google. Magic Leap non ha ancora rilasciato alcun prodotto, fornendo solamente dei video dimostrativi registrati direttamente da alcuni prototipi.

Al momento il sistema che si colloca in una posizione di vantaggio nella corsa allo sviluppo di questa tipologia di mixed reality, basata sulla rappresentazione di ologrammi, è Microsoft HoloLens, che nel marzo 2016 ha rilasciato la development edition del suo prodotto[8].

Capitolo 2

Microsoft HoloLens

Microsost HoloLens ha introdotto una tecnologia innovativa che ha il potenziale di rivoluzionare il concetto di mixed reality, e non solo. L'avvento di HoloLens può rivelarsi importante in vari settori, individuando una modalità efficiente per effettuare compiti al momento complessi.

In questo capitolo si andrà a realizzare una prima panoramica sul mondo HoloLens, individuando le caratteristiche più importanti del dispositivo, i principi teorici dello sviluppo olografico e i possibili ambiti di utilizzo della tecnologia lanciata da Microsoft.

2.1 Presentazione

Microsoft HoloLens è il primo dispositivo olografico in grado di sfruttare le potenzialità di Windows Holographic, la piattaforma di realtà aumentata ideata da Microsoft. Si presenta come un visore senza cavi, provvisto del sistema operativo Windows 10, che consente all'utente che lo indossa di poter essere immerso in un sistema di mixed reality, con la possibilità di interagire con delle entità olografiche tramite comandi vocali o determinate gestures.

Lo sviluppo del progetto HoloLens ha avuto inizio nell'anno 2010, ma Microsoft aveva già concepito la realizzazione di un dispositivo olografico durante la produzione del Kinect, un accessorio presentato nel 2009, originariamente pensato per la console Xbox 360, in grado di riconoscere i movimenti del corpo umano senza l'ausilio di alcun dispositivo esterno. Il 30 marzo 2016 Microsoft ha rilasciato la versione Development Edition degli HoloLens, limitandone però l'acquisto ai soli residenti in Stati Uniti e Canada, ad un prezzo di listino pari a $3000.

2.2 Dispositivo HoloLens

Figura 2.1: HoloLens device

Il visore HoloLens appartiene alla categoria degli smartglasses, nonostante il suo funzionamento permetta di caratterizzarlo in maniera unica, innovativa, rispetto agli altri dispositivi dello stesso tipo.

Come si può evincere in Figura 2.1 il dispositivo HoloLens si presenta come un head mounted display progettato per garantire un'esperienza di comfort elevata, infatti il peso degli occhiali olografici viene distribuito lungo la circonferenza della testa, evitando così di creare una pressione fastidiosa su orecchie e naso. L'apparecchio, inoltre, è regolabile in modo da potersi adattare perfettamente ad un vasto insieme di misure.

Il device HoloLens non necessita di alcun dispositivo esterno per compiere il suo funzionamento, garantendo così all'utente libertà di movimento. Nonostante le dimensioni ridotte, può vantare una notevole potenza di calcolo, addirittura superiore a quella di un discreto computer portatile. L'architettura hardware è composta da:

- Un set di sensori comprensivo di accelerometro, giroscopio, magnetometro e sensore di luce ambientale, utilizzati dal dispositivo per creare una ricostruzione dell'ambiente in cui è situato;

- Una telecamera con un angolo di vista pari a 120^ox120^o, che permette di individuare una vasta immagine dell'ambiente circostante;

- Un paio di lenti ad alta definizione, che mediante l'utilizzo di un "light engine" riesce ad offrire una visione completa degli ologrammi con una latenza minima;

- Un sistema audio basato su microfoni, che permettono all'utente di impartire dei comandi vocali, e altoparlanti, che consentono una riproduzione dei suoni emessi dai vari ologrammi come se provenissero dalla locazione nella quale sono stati generati;

- Una holographic processing unit, che si aggiunge a CPU e GPU, ed è un processore realizzato da Microsoft esclusivamente per gli HoloLens, incaricato principalmente della gestione delle informazioni rilevate dai sensori (che si aggirano sull'ordine di terabytes al secondo);

- Un ricevitore IEEE 802.11ax Wi-Fi e un ricevitore Bluetooth 4.1 Low Energy.

2.3 Concetto di ologramma

HoloLens si appoggia sulla piattaforma Windows Holographic, la quale è in grado di sviluppare un'esperienza basata sulla mixed reality. HoloLens si differenzia da tutte le altre tecnologie affini attualmente presenti sul mercato, dettando le basi per il futuro della mixed reality.

La visione di HoloLens è quella di creare ologrammi e disporli nel mondo reale come se gli appartenessero, come se fossero identici ad un qualunque altro oggetto fisico. Ed è proprio grazie a questo aspetto che HoloLens pone le distanze dai principi base dalla tecnologia augmented reality, nella quale si arricchisce la percezione sensoriale umana tramite informazioni aggiuntive riguardo all'ambiente, creando una netta separazione fra gli elementi del mondo fisico e quelli del mondo aumentato. Tale barriera in HoloLens non esiste.

Figura 2.2: Ologrammi nel mondo reale

Come si può notare dalla simulazione della visuale di un dispositivo HoloLens in Figura 2.2, gli ologrammi sono percepiti come parte del mondo fisico, è permesso all'utente di interagire con essi mediante lo sguardo, la voce o i gesti delle mani. In questo ultimo aspetto HoloLens si avvicina ai principi fondamentali di augmented virtuality, senza però creare un mondo virtuale nel quale immergere l'utente, ma effettuando il passaggio inverso, ovvero quello di portare gli elementi digitali nel mondo reale.

Gli ologrammi possono essere bidimensionali, come il calendario presente in Figura 2.2, oppure tridimensionali come i restanti oggetti virtuali disposti nella scena in Figura 2.2. Generalmente si tende a creare ologrammi 3D in quanto trasmettono un senso di realismo maggiore ed è più facile per l'utente considerarli come oggetti del mondo fisico.

2.4 Possibili utilizzi

In contemporanea con il rilascio di HoloLens sono state rese disponibili anche alcune applicazioni sviluppate appositamente per il dispositivo, in modo da sfruttarne le grandi potenzialità. Sono presenti una manciata di giochi che permettono di trasformare un semplice ambiente, come una stanza, in un livello di un videogioco. Inoltre saranno disponibili anche varie applicazioni, conosciute per il loro funzionamento su altre piattaforme, per le quali verrà effettuata una versione in grado di poter esaltare le qualità di HoloLens.

Allo stato attuale, l'unica applicazione universalmente riconosciuta per cui è stata effettuata un'apposita versione per il dispositivo HoloLens è Skype, nota applicazione di messaggistica istantanea e VoIP. Tramite la versione ottimizzata per HoloLens è possibile contattare un qualsiasi altro dispositivo, anche un PC o uno smartphone, che è provvisto della medesima app. Inoltre un utente che sta indossando un device HoloLens può condividere la sua visione del mondo, comprensiva degli ologrammi, con un altra persona. In questo modo tramite HoloNotes via Skype è possibile interagire con gli ologrammi presenti nella scena anche da un dispositivo esterno, con i cambiamenti che verranno percepiti in tempo reale dall'utente che indossa il visore.

Fra le applicazioni attualmente disponibili per HoloLens è presente anche HoloStudio, che più di ogni altra, rappresenta le funzionalità del dispositivo. Come illustrato in Figura 2.3, per mezzo di questa app è possibile creare degli ologrammi che permettano di facilitare la progettazione e modellazione di oggetti 3D, garantendo anche la possibilità di realizzarli fisicamente attraverso l'ausilio di una stampante 3D. Questa applicazione incarna in pieno quelli che sono, al momento, i possibili utilizzi del dispositivo HoloLens.

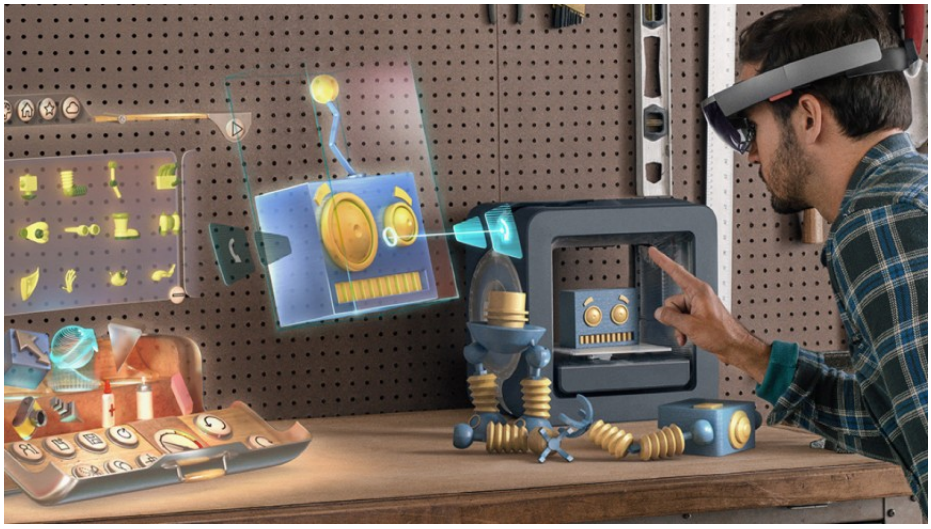

Figura 2.3: App Holo Studio per HoloLens

Il visore può diventare determinante in un ambito professionale, rivoluzionando in maniera significativa il processo di progettazione di elementi in ambito tridimensionale. Poter visualizzare in tempo reale il progetto 3D al quale si sta lavorando, sotto forma di ologramma, consentirebbe ad un ingegnere di ridurre drasticamente i tempi di produzione, permettendogli di effettuare le modifiche di sviluppo dell'oggetto direttamente sull'ologramma. Le potenzialità dell'utilizzo di HoloLens in un contesto professionale sono dunque notevoli con interessanti implicazioni in campo architetturale e medico.

Allo stato attuale HoloLens si presenta come uno strumento sicuramente interessante per le imprese, mentre non ancora pronto per esprimere tutte le sue potenzialità come accessorio di un utente medio.

È necessario, però, ricordare che il dispositivo rappresenta un primo vero approccio della mixed reality applicata al campo wearable, ed è tuttora in fase di sviluppo. Microsoft sta investendo molte risorse sull'espansione di questa tecnologia emergente, raccogliendo i suggerimenti degli utenti e rielaborandoli per costruire un prodotto che possa essere utilizzato in svariati modi diversi.

Il rilascio della consumer version di HoloLens, inizialmente previsto per la prima metà del 2016, è stato posticipato a data da definirsi, segno che Microsoft sta lavorando sulla versione completa del dispositivo, che conterrà degli elementi aggiuntivi rispetto alla development edition resa disponibile nel marzo 2016.

Capitolo 3

Sviluppare per HoloLens

Lo sviluppo di applicazioni HoloLens è ancora bloccato ad una prima fase embrionale, conseguenza del fatto che la devolpment edition è stata rilasciata solamente da qualche mese. È necessario conferire agli sviluppatori il tempo vitale per prendere confidenza con la nuova innovativa tecnologia e i recenti paradigmi di programmazione da essa instaurati.

In questo capitolo si andranno ad analizzare gli elementi essenziali da considerare per la realizzazione di un'applicazione HoloLens, partendo dalla scelta degli strumenti di sviluppo per poi andare ad analizzare i componenti principali della programmazione HoloLens.

3.1 Strumenti di sviluppo

Nel marzo del 2016 Microsoft ha rilasciato, oltre alla development edition di HoloLens, anche la relativa piattaforma di sviluppo, in modo da consentire la progettazione di applicazioni olografiche. HoloLens richiede la versione 2015 di Visual Studio con l'installazione dell'Update 2.

Vengono fornite due possibili alternative per lo sviluppo di applicazioni HoloLens:

- è possibile programmare utilizzando le Windows Holographic API fornite da Microsoft e realizzando una app UWP (Universal Windows Platform) che si appoggia su DirectX per il processo di renderizzazione degli ologrammi;

- alternativamente è possibile sviluppare gli ologrammi in maniera più accurata su Unity e poi compilare il progetto in Visual Studio solo alla fine del processo.

In entrambi i casi le applicazioni HoloLens prodotte possono essere testate direttamente sul dispositivo oppure tramite l'utilizzo di un emulatore che permette di simulare il funzionamento dell'app sul visore.

3.2 HoloLens e Windows Holographic APIs

Un'applicazione HoloLens progettata tramite l'utilizzo delle Windows Holographic APIs è sviluppata tramite Visual Studio e adotta la struttura di un un'app UWP (Universal Windows Platform). Per quanto riguarda gli elementi di resa grafica si appoggia alle funzioni di Direct3D. Sono quindi necessarie delle conoscenze di base di queste tecnologie per poter realizzare un'app HoloLens in modo indipendente da Unity.

3.2.1 Universal Apps

Una universal app è un'applicazione che sfrutta la Windows Universal Platform e garantisce la portabilità fra vari dispositivi che si fondano su un sistema operativo Windows 10.

Figura 3.1: Dispositivi su cui è possibile eseguire un'app UWP

In Figura 3.1 sono mostrati tutti i vari dispositivi che, muniti di Windows 10, sono in grado di supportare un'app UWP. È possibile riscontrare una notevole varietà, partendo dal semplice PC e giungendo fino ai ben più complessi dispositivi embedded, senza tralasciare il mondo mobile.

Nel 2012, con l'avvento di Windows Phone 8.1, Microsoft ha presentato Windows Runtime (WinRT), un'architettura comune per applicazioni destinate a Windows e a Windows Phone. Nel 2015 è stato rilasciato Windows 10

che ha introdotto, invece, la piattaforma Universal Windows Platform, un'evoluzione della precedente WinRT. Questa piattaforma offre un livello di API generale supportato da tutti i dispositivi, ed un set di API specifiche per l'utilizzo su un certo dispositivo. In questo modo è possibile realizzare una base comune dell'app, e poi aggiungere a questa le funzionalità specifiche di ciascun device, permettendo all'app di essere ottimizzata per l'esecuzione su una vasta gamma di dispositivi.

Attraverso l'introduzione della Windows Universal Platform, Microsoft ha posto le basi per un progresso importante per quanto riguarda lo progettazione di applicazioni in ambito Windows, permettendo agli sviluppatori di aumentare notevolmente la portabilità del codice, evitando loro di dover creare un'applicazione diversa per ogni dispositivo.

Fra i dispositivi che supportano la piattaforma UWP vi è anche HoloLens, come si può evincere dal quadro in Figura 3.1. Questo significa che qualsiasi applicazione sviluppata tramite la piattaforma UWP può essere eseguita sul visore olografico. È però evidente che applicazioni pensate per il mondo mobile o per PC non possono riuscire a sfruttare tutte le grandi potenzialità di HoloLens. Si trova quindi la necessità di integrare le API di base della piattaforma comune con delle holographic APIs specifiche per il dispositivo.

3.2.2 Applicazioni UWP olografiche con DirectX

Un'applicazione HoloLens è una Universal Windows Platform app il cui funzionamento è basato su DirectX, una collezione di API utilizzate in maniera prevalente per lo sviluppo di videogiochi sulla piattaforma Microsoft. La gestione degli aspetti grafici di DirectX è affidata a Direct3D, che costituisce un set di librerie per la realizzazione di oggetti 3D.

In maniera analoga rispetto ad una qualsiasi altra applicazione DirectX, la struttura dell'applicazione olografica è composta da un ciclo continuo, il cosiddetto game loop, da una classe DeviceResources, propria di Direct3D e dai costrutti necessari per effettuare il processo di rendering. Oltre a questi elementi, ne sono presenti altri necessari per realizzare la parte olografica dell'applicazione, composta delle Windows Holographic APIs.

L'app si basa sulla creazione di un holographic frame ad ogni iterazione. In ogni frame viene eseguita una funzione di aggiornamento nella quale viene elaborata la scena da visualizzare nel frame successivo. È in questa fase che viene definita la posizione degli ologrammi secondo il sistema di riferimento utilizzato.

Al termine della fase di aggiornamento, ovvero quando tutti gli elementi sono stati esaminati, si passa al processo di rendering, nel quale ogni ologramma viene visualizzato graficamente tramite le APIs di DirectX.

L'applicazione consiste quindi in un ciclo continuo, con l'alternanza della fase di aggiornamento della scena e quella di renderizzazione della stessa.

3.3 HoloLens e Unity

Sviluppare applicazioni HoloLens su Unity è possibile tramite un'edizione specifica del software, attualmente la HoloLens Technical Preview, basata sulla versione 5.4 Beta di Unity. Grazie a questo strumento è possibile curare in modo particolare l'aspetto grafico degli ologrammi presenti nella scena.

3.3.1 Unity

Unity è uno strumento multipiattaforma creato da Unity Technologies utilizzato per lo sviluppo di videogiochi per computer, console, dispositivi mobile e siti web. È stato rilasciato nel giugno 2015 come un'esclusiva Apple per i sistemi OS X, effettuando poi una progressione esponenziale che ha condotto il game engine ad essere uno dei più utilizzati in ambito di grafica tridimensionale, supportando attualmente 21 diverse piattaforme.

Il componente principale della progettazione in Unity è costituito dal game object. Un game object è un oggetto che viene disposto all'interno di una scena e può essere parte integrante del videogioco, rappresentando un qualunque elemento con il quale l'utente può interagire, oppure può identificare dei componenti il cui utilizzo è fondamentale per lo sviuppatore, ma che non sono tangibili all'utente finale, come camera e luci. In quest'ultima categoria sono inseriti anche dei componenti colliders che permettono di individuare una collisione fra due game objects [9].

Tutti i game objects, compresi quelli che non faranno parte del frame renderizzato, devono essere posizionati nella scena utilizzando un sistema di coordinate 3D con gli assi X e Z che si intersecano perpendicolarmente rimanendo entrambi sul piano di lavoro, con l'asse Y che, invece, punta verso l'alto.

3.3.2 Applicazioni olografiche con Unity

Utilizzando questa modalità di sviluppo, il progetto di un'applicazione HoloLens viene realizzato interamente su Unity, abbracciando i principi di funzionamento del game engine.

Un aspetto importante può essere identificato nel ruolo della camera, che in Unity è un componente che ha il compito di inquadrare la scena, in modo che vengano visualizzati solo gli oggetti che rientrano nell'obbiettivo. In HoloLens la camera è l'utente, quindi in un'applicazione realizzata con Unity l'oggetto camera rappresenta ciò che sta guardando l'utente. Saranno oggetto

del processo di render solamente gli ologrammi che si trovano all'interno del campo visivo della camera.

Per quanto riguarda lo sviluppo dell'applicazione, si procede con la creazione di game objects che, nel caso di HoloLens, identificano i vari ologrammi. Questi possono essere creati e modellati a piacimento dallo sviluppatore, altrimenti possono essere utilizzati dei prefab prelevati dall'asset store, il negozio digitale di Unity. È possibile giungere a elevati livelli di dettaglio, realizzando così ologrammi complessi e realistici.

Al termine del processo di creazione di un ologramma è necessario stabilire il suo comportamento, in modo da definire la sua evoluzione nel tempo e la sua reazione ad un'interazione espressa da un utente HoloLens. Ciò viene effettuato associando degli script all'oggetto in esame. Tali script, scritti ad esempio in linguaggio c#, sono delle porzioni di codice che interessano l'oggetto al quale sono assegnati e possono essere eseguiti nel momento in cui il sistema rileva un certo evento. Ad esempio è possibile definire una serie di azioni che un ologramma dovrà eseguire nell'istante in cui dispositivo si accorge di una determinata gesture effettuata dall'utente verso l'ologramma stesso.

Nella fase successiva alla progettazione degli ologrammi si andrà poi a compilare il progetto. Unity, in questo caso, svolge un notevole lavoro di supporto allo sviluppatore, andando in automatico a creare una soluzione Visual Studio contenente tutto il necessario per eseguire l'applicazione.

3.4 Elementi principali di un'app HoloLens

Sia nel caso in cui lo sviluppo di un'applicazione HoloLens venga effettuato con l'ausilio di Unity, sia che si proceda con l'utilizzo delle Windows Holographic API con DirectX, i principi da conoscere e implementare sono gli stessi. Gli elementi principali di HoloLens possono essere individuati come:

- sistema di riferimento,
- gaze,
- gestione degli input,
- spatial mapping,
- spatial sound.

Ognuno di essi deve essere preso in considerazione per realizzare un'applicazione che riesca a sfruttare pienamente le grandi potenzialità di HoloLens.

3.4.1 Sistema di riferimento

HoloLens si basa su di un sistema di coordinate cartesiane costituito da tre assi perpendicolari: X, Y e Z. È necessario però precisare che il sistema di riferimento può essere definito tramite delle coordinate virtuali, e quindi relative al dispositivo, oppure reali, e quindi relative al mondo circostante.

Un sistema di coordinate reali in HoloLens è definito spatial coordinate system. Per facilitare la progettazione degli ologrammi, tali coordinate vengono espresse in metri. Gli assi del sistema di riferimento devono essere orientati in una maniera prestabilita, per permettere di evitare confusione. Lo spatial coordinate system si basa su di un sistema cosiddetto right-handed, definito in questo modo perché è possibile immaginarne una visualizzazione tramite il posizionamento delle dita della mano destra. Viceversa sarebbe stato possibile utilizzare un sistema left-handed. In entrambi i sistemi l'asse X punta verso destra e l'asse Y verso l'alto, l'unica differenza è la posizione dell'asse Z, che nei sistemi right-handed punta all'indietro, mentre in quelli left-handed punta in avanti.

Stationary frame of reference Per poter utilizzare il sistema di riferimento all'interno di un'applicazione HoloLens è necessario stabilire un punto dello spazio che possa rappresentare un riferimento per il dispositivo. Questo punto andrà a costituire l'origine del sistema di coordinate.

In questo modo sarà possibile posizionare gli ologrammi nel mondo reale utilizzando un sistema predefinito e immutabile. La posizione degli ologrammi sarà gestita come un vettore di tre coordinate che andranno a rappresentare la distanza espressa in metri dal punto preso come riferimento. Tale punto è definito come un stationary frame of reference, e su di esso viene costruito il sistema di coordinate dell'applicazione HoloLens.

In questo modo gli ologrammi posizionati secondo questo sistema di coordinate rimangono fissati in un certo punto, indipendentemente dalla posizione dell'utente e dalla direzione del gaze.

Spatial anchors La posizione di tutti gli ologrammi che sono creati utilizzando un sistema di riferimento relativo ad un stationary frame of reference potrebbe essere soggetta a leggere correzioni con il passare del tempo. La posizione stabilita inizialmente per un certo ologramma potrebbe non essere completamente accurata, in quanto al momento della creazione dell'ologramma il dispositivo potrebbe non avere i mezzi necessari per conoscere alla perfezione l'ambiente attorno a sé. La percezione che il dispositivo HoloLens ha dello spazio circostante è costantemente in aggiornamento, quindi la posizione di un certo ologramma potrebbe variare leggermente mano a mano che il dispositivo

acquisisce informazioni sull'ambiente in cui è situato. Questo processo potrebbe causare un effetto di drift, con l'utente che potrebbe avere una visione di ologrammi non perfettamente stabili.

Per evitare questa possibilità sono state introdotte le spatial anchors. Questo meccanismo consente, dopo aver definito un stationary frame of reference, di fissare varie ancore, ognuna delle quali andrà a costituire l'origine di un nuovo sistema di riferimento. Tali ancore, una volta posizionate, non possono più essere spostate. Gli ologrammi costruiti sul sistema di riferimento di una spatial anchor risultano stabili nel tempo, in quanto, essendo soggette ad un sistema di riferimento indipendente da quello stabilito dal stationary frame of reference, non subiscono alcun effetto di drift. L'unico punto critico è il continuo aggiornamento di una spatial anchor rispetto ad altre ancore per garantire che ciascuna di esse rimanga esattamente dove era stata posta rispetto al mondo reale.

Solitamente si ricorre all'utilizzo di una spatial anchor nel momento in cui il compito di stabilire la posizione di un certo ologramma è demandato all'utente. In questo caso è necessario che il sistema di riferimento garantisca una certa stabilità nel tempo, evitando di trasmettere all'utente la percezione che un ologramma disposto nel mondo direttamente da lui non mantenga in maniera precisa la posizione da lui prestabilita.

Le spatial anchors costituiscono un meccanismo fondamentale anche per permettere all'applicazione di tenere traccia di una certa location in maniera persistente, anche a seguito dello spegnimento del dispositivo. Ciò viene realizzato attraverso il salvataggio della spatial anchor in un spatial anchor store presente all'interno dell'applicazione, per permettere il loro utilizzo in un secondo momento semplicemente caricandole dallo store.

Inoltre le spatial anchor possono essere condivise attraverso un processo di sharing. In tal modo due o più dispositivi possono stabilire una condivisione dello spazio, avendo una medesima visione dei vari ologrammi comuni relativi alle ancore condivise.

Attached frame of reference In alcuni casi non serve che gli ologrammi siano fissati in un punto preciso del mondo, ma è invece necessario che si muovano in modo da seguire l'utente nei suoi spostamenti. Per implementare questa idea HoloLens mette a disposizione un attached frame of reference.

Solitamente questi ologrammi sono del tipo body-locked content, ovvero si limitano a spostarsi mantenendo sempre una certa distanza dall'utente. È possibile utilizzare anche ologrammi del tipo head-locked content che, oltre a seguire i movimenti dell'utente, modificano la loro posizione in modo da rimanere sempre visibili all'utente nel display del dispositivo. Questi ultimi

sono poco utilizzati in quanto possono risultare fastidiosi ed è più difficile per l'utente considerarli parte del mondo reale.

3.4.2 Gaze

Il gaze è un elemento fondamentale dell'universo HoloLens. Sta ad indicare la direzione nella quale sta guardando l'utente. Viene rappresentato come un raggio laser che parte in linea retta dal centro del dispositivo e può intersecare vari oggetti sulla sua strada, andando quindi a cercare di interpretare la direzione dello sguardo dell'utente.

Il gaze riveste un'importanza chiave nello sviluppo di applicazioni HoloLens, in quanto da esso dipendono tutte le interazioni che l'utente può avere nei confronti degli ologrammi presenti nella scena. Può essere immaginato come un cursore che permette all'utente di selezionare tramite lo sguardo un certo oggetto, reale oppure virtuale, presente nello spazio.

Quando il dispositivo rileva l'intenzione dell'utente di interagire con un oggetto attraverso una gesture oppure tramite un comando vocale viene utilizzato il gaze per identificare l'oggetto interessato. Per far ciò è necessario cercare un'intersezione tra il gaze, considerato come una retta che parte dal dispositivo, e i vari ologrammi presenti nella scena, andando poi a selezionare l'oggetto più vicino all'utente che rispetta questo vincolo.

Può risultare utile, in alcuni casi, visualizzare un cursore che rappresenta il gaze nel momento in cui esso interseca un oggetto rilevante, in modo da fornire all'utente l'indicazione dell'oggetto sul quale avranno effetto le varie forme di input che andrà ad effettuare.

3.4.3 Gestione degli input

La gestione degli input è un aspetto essenziale per garantire all'utente la possibilità di interagire con gli ologrammi situati nelle sue immediate vicinanze.

Una forma di interazione può essere identificata nel gaze. Come illustrato precedentemente, il Gaze indica la direzione dello sguardo dell'utente, quindi tramite questo elemento è possibile stabilire quale ologramma è sotto osservazione in un certo istante e reagire di conseguenza, ad esempio effettuando una selezione dell'oggetto in esame. L'analisi del gaze non è però sufficiente per effettuare un'interazione completa, ma è necessario che l'utente manifesti la volontà di comunicare con l'oggetto attraverso una gesture o un comando vocale.

Gestures Una delle forme primarie di input è costituita dalle gestures. Una gesture è effettuata tramite un preciso movimento di una mano visibile dal

display del dispositivo. Esistono solamente due posizioni della mano che sono riconoscibili dal dispositivo, ovvero quando la mano è in un ready state (pugno chiuso in avanti con l'indice alzato) oppure quando è in un pressed state (pugno chiuso in avanti con l'indice abbassato).

La gesture più diffusa consiste in una sorta di click e viene effettuata ponendo la mano inizialmente in posizione di ready, per poi passare nello stato pressed e tornare rapidamente allo stato di ready. Questa gesture, definita press and release, viene utilizzata solitamente per selezionare o attivare un certo oggetto, identificato dal gaze. Prolungando il tempo di pressed di una press and release si compone un'altra gesture, detta hold. Questa gesture può essere utilizzata per effettuare sull'oggetto un'azione diversa da quella del press and release.

Può essere necessario in vari casi permettere all'utente di modificare direttamente la struttura dell'ologramma, spostandolo, ruotandolo o ridimensionandolo. Queste azioni possono essere attuate tramite una manipulation gesture, che consiste nel movimento della mano nello spazio dopo essere passati da uno stato di ready ad uno di pressed. L'identificazione dell'oggetto su cui effettuare una manipulation è effettuata, come nei casi precedenti, tramite il gaze; con l'unica differenza che una volta che inizia il movimento della mano, il gaze non è più rilevante per l'oggetto selezionato, a questo punto lo spostamento dell'oggetto dipende interamente dalla posizione della mano.

Una gesture molto simile alla manipulation è quella di navigation, che si effettua in maniera analoga, con l'unica differenza che in questo caso il movimento della mano è limitato e serve, ad esempio, per attuare azioni come selezionare una certa operazione da un menù, oppure scorrere gli elementi di una lista.

È inoltre presente una gesture cosiddetta bloom che consente in qualunque momento di tornare al menù di start principale (è dunque l'equivalente del windows key nella tastiera di un PC). Tale comportamento viene innescato inizialmente ponendo la mano con il palmo rivolto verso l'alto e le dita congiunte, per poi aprire la mano.

Comandi vocali Un'altra potente forma di input è costituita dai comandi vocali. Attraverso questo meccanismo si può evitare di dover effettuare delle gestures, ma si può direttamente interagire con un ologramma guardandolo (in modo che il gaze permetta di identificarlo) e pronunciando un certo comando. Ad esempio tramite la parola chiave "select" è possibile ottenere lo stesso risultato di una press and release gesture. La lista dei comandi vocali utilizzabili dall'utente è molto vasta e tramite la frase "hey Cortana" è possibile impartire delle istruzioni o chiedere informazioni di vario tipo.

3.4.4 Spatial mapping

Lo spatial mapping permette al dispositivo HoloLens di ricavare una rappresentazione delle superfici presenti nell'ambiente circostante rispetto all'utente. In questo modo è possibile controllare la posizione e il comportamento degli ologrammi in modo da portare il realismo ad un livello elevato. Grazie a questa tecnica si può evitare che un ologramma si sovrapponga ad una superficie esistente, oppure gestire le collisioni fra un'entità olografica e un oggetto reale.

Tramite l'utilizzo dei dati elaborati dai sensori presenti nel visore è possibile ricostruire anche solo una parte limitata dell'ambiente, utilizzabile nelle applicazioni HoloLens tramite l'introduzione di Spatial Surface. Un'applicazione HoloLens può beneficiare dell'utilizzo dello spatial mapping in vari modi.

Occlusion Attraverso il processo di occlusione si può aumentare in modo determinante il tasso di realismo che un utente può percepire utilizzando un dispositivo HoloLens. Infatti in questo modo un ologramma può essere identificato come un qualunque altro oggetto del mondo fisico, non venendo visualizzato nel momento in cui sono presenti delle superfici fisiche fra la sua posizione e quella dell'utente.

L'ologramma, in questo caso, può essere nascosto completamente all'utente, oppure può essere visualizzato in modo diverso, ad esempio con un livello di luminosità più alto. Utilizzando quest'ultima alternativa è possibile, allo stesso tempo, sia permettere all'utente di essere in grado di localizzare gli ologrammi presenti nell'ambiente, sia mostrare all'utente quando un ologramma si trova dietro una certa superficie.

Visualization Dopo aver elaborato lo spatial mapping dell'ambiente, è possibile mostrare all'utente la ricostruzione effettuata. Questa azione può sembrare contraria ai principi cardine di HoloLens, in quanto diminuisce il senso di realismo percepito dall'utente, ma in alcuni casi può rivelarsi estremamente utile. In HoloLens spesso viene lasciata la libertà all'utente di poter decidere il punto preciso in cui posizionare un ologramma, in questo caso è consigliabile visualizzare la visione del mondo elaborata dall'applicazione, in modo di guidare l'utente alla scelta di una posizione che sia congeniale sia allo spazio fisico che al sistema di riferimento di HoloLens.

Inoltre tramite la visualization l'applicazione è in grado di mostrare all'utente il suo livello di comprensione dell'ambiente circostante.

Placement Il posizionamento di un ologramma in un punto dello spazio reale indicato dall'utente è un processo molto complesso. Infatti l'applicazione riceve

in input soltanto una direzione sotto forma di un vettore, che può coincidere con il gaze, oppure partire dal dito dell'utente mentre indica il punto in cui piazzare l'ologramma. Non è quindi possibile esaminare il concetto di distanza, che deve essere escogitato dall'applicazione nella maniera più precisa possibile.

Un aiuto importante viene fornito dall'elaborazione dello spatial mapping che consente di evitare di porre l'ologramma in corrispondenza di altre superfici e permette di posizionarlo nel modo più congeniale all'ologramma stesso. Ad esempio se l'ologramma è la rappresentazione di una persona sarà necessario fare in modo che sia visualizzato in modo che i piedi siano a contatto con la superficie che rappresenta il pavimento.

Inoltre tramite l'elaborazione dello spatial mapping, il posizionamento degli ologrammi da parte dell'applicazione, senza l'indicazione dell'utente, può essere effettuato in maniera da non interferire con le superfici reali, oppure direttamente a contatto diretto con queste.

Physics Per conferire agli ologrammi un ulteriore livello di realismo è necessario che questi reagiscano alle leggi fisiche come un qualunque oggetto fisico. Ciò può essere eseguito tramite un'elaborazione approfondita delle superfici dell'ambiente, in modo da identificare con la maggiore accuratezza possibile gli oggetti presenti.

Immaginiamo di avere un ologramma che rappresenta una pallina da tennis posta su un tavolo. Per aumentare il senso di realismo nel momento in cui la pallina viene lasciata cadere è necessario regolare il suo spostamento secondo le leggi della gravità e al contatto con una superficie che rappresenta il pavimento simulare i movimenti di rimbalzo e rotolamento.

Navigation Una parte considerevole della realtà olografica è costituita da ologrammi non stabili, che possono muoversi all'interno dell'ambiente in cui è situato l'utente. Per rendere questi spostamenti il più reale possibile è necessario che l'applicazione sia in grado di identificare lo spazio navigabile, in modo da evitare movimenti che non potrebbero essere effettuati da entità del mondo fisico. Anche in questo caso lo studio approfondito delle superfici rilevate dal processo di spatial mapping diventa fondamentale.

3.4.5 Spatial sound

Il dispositivo HoloLens è dotato di un sofisticato sistema audio che permette di simulare dei suoni 3D. Un suono viene trasmesso all'utente in modo diverso a seconda della direzione e della distanza relative all'ologramma che ha emesso il suono. L'esperienza olografica acquisisce un livello di realismo

notevole, con l'utente che avvicinandosi ad un ologramma è in grado di percepire un aumento dell'intensità del suono prodotto da tale oggetto, viceversa allontanandosi potrà udire il suono diminuire progressivamente. Si verifica lo stesso effetto nel momento in cui è l'ologramma a muoversi, avvicinandosi o allontanandosi dalla posizione dell'utente.

Lo spatial sound può essere utile anche per permettere all'utente di identificare la posizione di un ologramma. Il visore rappresenta soltanto gli ologrammi che rientrano nella sezione dell'ambiente visualizzata dall'utente in un certo istante, non mostrando tutti gli ologrammi situati al di fuori di questa finestra. Tramite l'utilizzo dello spatial sound l'utente può percepire il suono emesso dagli ologrammi che non fanno parte del suo campo visivo e dedurre la loro posizione.

3.5 HoloLens emulator

Figura 3.2: Emulatore HoloLens

L'emulatore, reso disponibile da Microsoft in concomitanza con il rilascio della development edition di HoloLens, costituisce uno strumento importante per lo sviluppo di applicazioni olografiche. Considerando le restrizioni imposte sull'acquisto del visore, gli sviluppatori sono vincolati all'utilizzo dell'emulatore per simularne il comportamento, in modo da valutare la correttezza delle applicazioni progettate.

Il funzionamento dell'emulatore si basa su Hyper-V, un hypervisor che permette di creare delle macchine virtuali in sistemi Windows. Le applicazioni HoloLens testate con questa modalità saranno, quindi, eseguite su una macchina virtuale, con l'emulatore che può essere considerato come un dispositivo esterno, indipendente dalla macchina fisica su cui è situato. Attraverso l'utilizzo della scheda di rete della macchina principale sarà possibile connettere

ad internet l'emulatore, che essendo un'entità distinta dal computer sul quale viene installato, avrà un indirizzo IP differente.

Le applicazioni vengono eseguite sull'emulatore allo stesso modo che sul dispositivo HoloLens autentico. Le interazioni fra utente e app possono essere simulate tramite l'utilizzo di mouse e tastiera. In questo modo è possibile testare il comportamento dell'applicazione al verificarsi dell'esecuzione delle gestures principali, potendo inoltre controllare lo spostamento dell'utente e la direzione del gaze.

La criticità principale dell'emulatore è dettata dal fatto che, al contrario del dispositivo fisico, non permette all'utente di visualizzare l'ambiente circostante. Si può notare dalla schermata in Figura 3.2 come la parte non olografica sia sostituita da uno sfondo nero. La filosofia di HoloLens è costituita da un mondo reale nel quale vengono inseriti degli ologrammi, e non poter essere in grado di visualizzare tale realtà limita notevolmente l'esperienza di utilizzo.

Per cercare di ovviare a questo problema sono state introdotte delle stanze simulate, costruite intorno all'utente in modo da poter applicare i concetti dello spatial mapping. Le stanze però non vengono visualizzate nell'emulatore, ma sono solamente costruite per una questione strutturale. L'unico modo per avere una visione vicina a quella reale consiste nell'effettuare il rendering delle mesh dello spatial mapping in ogni scena.

Capitolo 4

Esplorazione della struttura di un'applicazione olografica

Dopo aver esaminato gli strumenti di sviluppo ed analizzato i principali elementi di un'applicazione HoloLens è giunto il momento di addentrarsi nello studio della struttura implementativa di un app olografica. Si è scelto di focalizzare la propria attenzione sull'utilizzo pratico di Universal Windows Platform e DirectX, escludendo Unity dal processo di sviluppo.

In questo capitolo verrà esaminato un codice di esempio fornito da Microsoft HoloLens e ne verranno messe in luce le principali problematiche e punti critici, prendendo in considerazione le possibili soluzioni adottabili.

4.1 Analisi di un esempio HoloLens

In allegato agli strumenti di sviluppo esaminati precedentemente, Microsoft HoloLens fornisce anche un'applicazione esplicativa che costituisce un vero e proprio template. Il prodotto finale di tale applicazione è molto semplice, e consiste nella rappresentazione di un cubo multicolore che ruota su se stesso ad una certa velocità costante.

Al lancio dell'applicazione viene eseguito il metodo `main` presente all'interno della classe `Program`. Qui viene avviata la classe `AppViewSource` che si preoccupa del comportamento dell'applicazione e la classe `CoreWindow`, che consente alla stessa di gestire le modifiche di stato e integrarsi con diversi framework dell'interfaccia utente.

Nella classe `AppViewSource` le prime operazioni che vengono eseguite sono quelle di creazione delle risorse necessarie per permettere alle API di Direct3D di effettuare il processo di rendering. Viene poi definito un oggetto di tipo `HoloLensExampleMain`, nel quale è incapsulato il funzionamento dell'applicazione. `AppViewSource` si limita a invocare i metodi `SetHolographicSpace`,

`Update` e `Render` sull'oggetto `HoloLensExampleMain`. Il primo metodo viene invocato una sola volta all'avvio dell'applicazione, mentre i restanti sono eseguiti alternativamente ad intervalli prefissati per tutta la durata dell'esecuzione dell'applicazione.

La classe `HoloLensExampleMain` è in assoluto quella principale e ha il compito della definizione di tutti gli elementi olografici dell'applicazione, compresi gli ologrammi e le gesture di input. La struttura dell'applicazione può essere definita attraverso lo studio dei blocchi nei quali può essere suddivisa questa classe. La struttura dell'applicazione può, quindi, essere esaminata attraverso l'individuazione di tre blocchi funzionali principali:

- definizione dello spazio olografico,
- aggiornamento degli elementi della scena,
- renderizzazione della scena.

La definizione dell'holographic space viene effettuata nel momento in cui viene lanciata l'applicazione e non si ripete più durante il funzionamento della stessa. Dopo la prima esecuzione il ciclo di vita dell'applicazione sarà costituito dall'alternanza delle fasi di updating e rendering.

I componenti principali dell'applicazione sono costituiti da `HoloLensExampleMain`, incaricato della gestione dei vari elementi, da un oggetto di tipo `SpinningCubeRenderer`, che rappresenta l'ologramma di un cubo che ruota su se stesso e da un oggetto della classe `SpatialInputHandler`, che si occupa della gestione degli eventi rilevati in input.

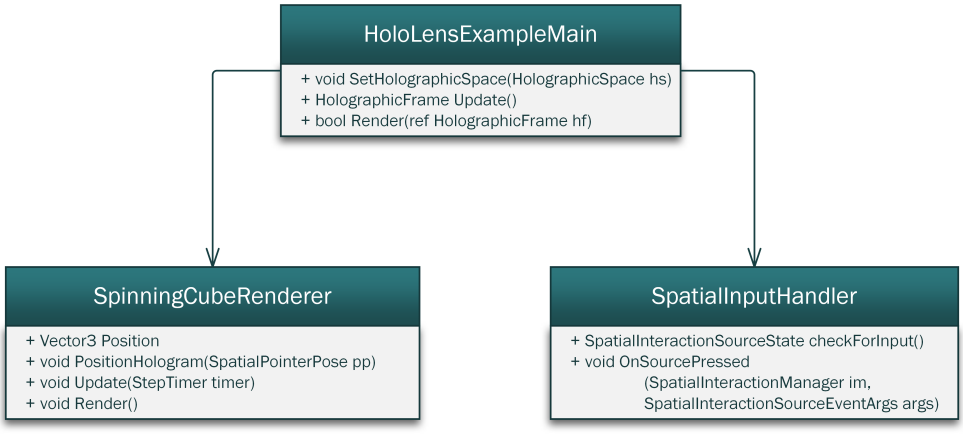

Figura 4.1: Elementi principali dell'applicazione di esempio

Come si può notare in Figura 4.1, la classe `HoloLensExampleMain` contiene al suo interno sia l'ologramma del cubo che il gestore di eventi, e ha il compito

di fare da tramite tra i due oggetti, che non sono relazionati in alcun modo. Sono mostrati anche i metodi pubblici più rilevanti di ogni classe.

HoloLensExampleMain Come già espresso precedentemente il funzionamento di questa classe può essere individuato dai metodi `SetHolographicSpace`, `Update` e `Render`.

Nel metodo `SetHolographicSpace` vengono creati gli elementi olografici dell'applicazione.

```
public void SetHolographicSpace(HolographicSpace holographicSpace)
{
    this.holographicSpace = holographicSpace;

    spinningCubeRenderer = new SpinningCubeRenderer(deviceResources);

    spatialInputHandler = new SpatialInputHandler();

    locator = SpatialLocator.GetDefault();
    locator.LocatabilityChanged += this.OnLocatabilityChanged;

    holographicSpace.CameraAdded += this.OnCameraAdded;
    holographicSpace.CameraRemoved += this.OnCameraRemoved;

    referenceFrame =
        locator.CreateStationaryFrameOfReferenceAtCurrentLocation();
}
```

Vengono assegnati dei metodi adibiti alla gestione degli eventi di aggiunta e rimozione di una camera, nello specifico rispettivamente `OnCameraAdded` e `OnCameraRemoved`. Il sistema di riferimento utilizzato si basa su un stationary frame of reference, collocato in un punto che rappresenta la posizione del dispositivo all'avvio dell'applicazione. Questo punto, segnalato da un oggetto di tipo `SpatialLocator`, andrà a costituire l'origine del sistema di riferimento fisso del mondo. Attraverso l'assegnazione del metodo `OnLocabilityChanged` è possibile reagire agli eventi di cambiamenti nello stato di tracciamento della posizione. In questo contesto viene istanziato l'oggetto rappresentante l'ologramma del cubo, di tipo `SpinningCubeRenderer` e il gestore di eventi, di tipo `SpatialInputHandler`.

L'obiettivo del metodo `Update` è quello di costruire un nuovo holographic frame ad ogni iterazione. Un holographic frame consiste in una serie di informazioni riguardanti gli elementi della scena corrente, come il sistema di riferimento utilizzato e la posizione della camera. L'holographic frame prodotto

sarà utilizzato successivamente per effettuare il rendering del frame successivo. In questa fase viene inoltre effettuato l'aggiornamento degli elementi principali dell'applicazione. Tutte le operazioni collocate in questo punto saranno eseguite una volta per frame.

```
SpatialInteractionSourceState pointerState =
    spatialInputHandler.CheckForInput();
if (null != pointerState)
{
    spinningCubeRenderer.PositionHologram(
        pointerState.TryGetPointerPose(currentCoordinateSystem)
        );
}

timer.Tick(() =>
{
    spinningCubeRenderer.Update(timer);
});
```

Viene richiesto al gestore di input di controllare se è stata rilevata una gesture del tipo specifico di press and release. In tal caso si ottiene un oggetto di tipo `SpatialInteractionSourceState` che contiene gli elementi rilevanti alla gesture rilevata. Fra questi elementi vi è anche un oggetto `SpatialPointerPose` che indica posizione e direzione del gaze nel momento in cui è stata rilevata la gesture. Quest'oggetto viene poi utilizzato come parametro per il metodo `PositionHologram` dell'oggetto `SpinningCubeRenderer`, che si limita ad effettuare uno spostamento del cubo. La classe `HoloLensExampleMain` fa uso di un timer che permette di gestire le operazioni che vanno eseguiti ad intervalli di tempo predefiniti. In questo caso il timer è settato in modo che il metodo `Tick` venga eseguito una volta per frame, come ogni altra operazione presente nel metodo `Update`, non andando ad incidere in nessun modo sullo svolgimento degli operazioni. Nel caso specifico viene eseguito il metodo `Update` del cubo, che consiste nell'attuazione di una sua rotazione parziale.

Il metodo `Render`, incaricato di effettuare il rendering della scena, riceve in input un holographic frame, generato nella fase di updating, e lo elabora costruendo una rappresentazione grafica tridimensionale di tutti gli elementi che rientrano nel campo visivo della camera. Viene dunque effettuato il rendering dei singoli ologrammi, nel caso specifico eseguendo il metodo `Render` dell'oggetto `SpinningCubeRenderer`.

SpinningCubeRenderer Si tratta di una classe che non esprime il concetto di ologramma, ma che è di fatto un costrutto implementato per eseguire il

processo di rendering. Una dimostrazione del concetto appena illustrato è data dal fatto che l'unico parametro del costruttore della classe consiste in un oggetto che incapsula una logica di utilizzo per l'effettuazione del rendering.

L'unico concetto della struttura vera e proprio dell'oggetto è costituito dalla sua posizione. Viene anche concessa la possibilità di modificare questa proprietà, attraverso il metodo `PositionHologram`.

```
public void PositionHologram(SpatialPointerPose pointerPose)
{
    if (null != pointerPose)
    {
        Vector3 headPosition     = pointerPose.Head.Position;
        Vector3 headDirection    = pointerPose.Head.ForwardDirection;
        float  distanceFromUser  = 2.0f;
        Vector3 gazeAtTwoMeters  = headPosition + (distanceFromUser
            * headDirection);

        this.position = gazeAtTwoMeters;
    }
}
```

Il metodo esegue un comportamento molto specifico, infatti permette di modificare la posizione del cubo in modo che esso venga posto esattamente davanti all'utente, a due metri di distanza. Si richiede come parametro il gaze, dal quale poi vengono estrapolati i valori dei vettori di posizione e direzione, utilizzati per stabilire la nuova posizione dell'oggetto.

Il metodo che permette la rotazione del cubo è `Update`, che, chiamato ad ogni frame, modifica la posizione del cubo in modo da fornire un effetto ottico di rotazione.

```
public void Update(StepTimer timer)
{
    float    radiansPerSecond = this.degreesPerSecond *
        ((float)Math.PI / 180.0f);
    double   totalRotation    = timer.TotalSeconds * radiansPerSecond;
    float    radians          =
        (float)System.Math.IEEERemainder(totalRotation, 2 * Math.PI);
    Matrix4x4 modelRotation   = Matrix4x4.CreateFromAxisAngle(new
        Vector3(0, 1, 0), -radians);

    Matrix4x4 modelTranslation =
        Matrix4x4.CreateTranslation(position);
```

```
    Matrix4x4 modelTransform = modelRotation * modelTranslation;

    this.modelConstantBufferData.model =
        Matrix4x4.Transpose(modelTransform);

    if (!loadingComplete)
    {
        return;
    }

    var context = this.deviceResources.D3DDeviceContext;

    context.UpdateSubresource(ref this.modelConstantBufferData,
        this.modelConstantBuffer);
}
```

In questo caso si agisce sugli elementi che saranno utilizzati successivamente nella fase di rendering, attraverso l'utilizzo di codice Direct3D. Viene definita una matrice modelTransform che sarà applicata all'oggetto, come una combinazione di una matrice di traslazione, per la gestione della posizione, e di una matrice di rotazione, per quanto riguarda la rotazione dell'oggetto.

Nel metodo di `Render` viene invece effettuato il processo di rendering dell'oggetto. In questa fase si ha un contatto diretto con del codice Direct3D che attraverso l'utilizzo di `VertexShader`, `GeometryShader` e `PixelShader` permette di effettuare una rappresentazione grafica dell'oggetto.

SpatialInputHandler La classe `SpatialInputHandler` rappresenta un oggetto in grado di rilevare degli eventi di input. In questo caso viene limitata la rilevazione alle sole gesture di press and release, ignorando le altre possibili forme di input.

```
public class SpatialInputHandler
{
    private SpatialInteractionManager interactionManager;
    private SpatialInteractionSourceState sourceState;

    public SpatialInputHandler()
    {
        interactionManager =
            SpatialInteractionManager.GetForCurrentView();
        interactionManager.SourcePressed += this.OnSourcePressed;
    }
```

```
    public SpatialInteractionSourceState CheckForInput()
    {
        SpatialInteractionSourceState sourceState = this.sourceState;
        this.sourceState = null;
        return sourceState;
    }

    public void OnSourcePressed(SpatialInteractionManager sender,
        SpatialInteractionSourceEventArgs args)
    {
        sourceState = args.State;
    }
}
```

Viene definito nel costruttore un oggetto di tipo `SpatialInteractionManager`, in grado di rilevare eventi. Ad esso viene assegnato un metodo `OnSourcePressed` che viene eseguito nel momento in cui viene rilevata una gesture di tipo press and release. In questo caso tale gesture, di tipo `SpatialInteractionSourceState`, viene salvata in una variabile locale, accessibile dall'esterno attraverso l'invocazione del metodo `CheckForInput`.

4.2 Principali problematiche riscontrate

Dall'analisi dell'esempio precedente si può evincere come la struttura di un'applicazione HoloLens sia analoga a quella di un videogame. Infatti sia l'utilizzo degli strumenti grafici, in particolare DirectX, che la definizione degli oggetti sono costruiti in modo da produrre un'applicazione che sia in grado di abbinare elevate prestazioni grafiche ad una latenza minima. Questo approccio applicativo genera, però, varie problematiche.

Assenza di una visione ad oggetti Nel contesto in esame i principi cardine della programmazione ad oggetti passano in secondo piano rispetto agli aspetti grafici. Non esiste una visione di un ologramma caratterizzato da un proprio modello indipendente dalla rappresentazione grafica. La costruzione degli oggetti è orientata solamente ad effettuare il processo di rendering nella maniera più semplice e immediata possibile.

Questo approccio può essere applicabile senza particolari problemi in applicazioni banali, come quella di esempio esaminata in precedenza, ma mostra una serie di svantaggi notevoli nel momento in cui è necessario costruire delle entità complesse. Attraverso una visione ad oggetti la gestione di progetti di grandi dimensioni risulta più facile e immediata, con gli ologrammi che pos-

sono essere definiti attraverso delle proprietà che consentono di identificarli secondo le loro caratteristiche e il loro comportamento. Uno dei vantaggi più consistenti riguarda la possibilità di riuso di codice e l'estendibilità delle classi prodotte, garantite dal rispetto dei principi di incapsulamento, ereditarietà e polimorfismo, alla base della filosofia della programmazione ad oggetti.

Intersezione fra elementi grafici e di modello Nell'esempio esaminato gli elementi riguardanti il modello dell'oggetto e quelli relativi alla sua rappresentazione grafica sono mischiati in un'unica classe generale. In questo modo lo stato di un oggetto e la sua visualizzazione sono collegati e inscindibili. Da questa concezione, completamente opposta rispetto al pattern architetturale model-view-controller, possono derivare alcune limitazioni. La principale riguarda l'impossibilità di fornire rappresentazioni diverse per uno stesso oggetto, in quanto la vista e il modello dell'oggetto sono espresse all'interno della stessa classe e dipendono l'una dall'altra. Inoltre lo sviluppatore non può limitarsi alla concezione dello stato interno dell'oggetto trascendendo la sua rappresentazione, ma è costretto a portare avanti uno sviluppo contemporaneo di modello e vista, che risulta sicuramente più scomodo da gestire. Un'altra problematica da esaminare consiste nel fatto che un oggetto, inteso come entità fisica, potrebbe anche non avere una rappresentazione olografica. Si pensi ad esempio ad un sistema nel quale sono presenti agenti di controllo che non necessitano di una rappresentazione grafica, ma sono comunque entità del mondo aumentato.

Mancanza di una gestione degli ologrammi Esaminando il quadro riassuntivo in Figura 4.1, si può notare come la gestione di ologrammi ed eventi sia affidata esclusivamente al `HoloLensExampleMain`. Nel momento in cui viene rilevata una certa gesture in input dallo `SpatialInputHandler`, la classe `HoloLensExampleMain` deve valutare la tipologia di gesture e modificare di conseguenza l'ologramma corrispondente. Questo concetto può funzionare nel momento in cui è presente un solo ologramma nella scena, ed il numero di gesture è limitato. Immaginiamo però di avere un numero elevato di ologrammi, e che questi ologrammi siano tutti di tipo diverso; in questo caso è ragionevole pensare che ognuno di questi possa reagire ad una stessa gesture in modo diverso, a seconda delle sue caratteristiche. Si crea quindi la necessità di un oggetto che sia in grado di gestire i vari ologrammi, ricevendo le gesture di input da un gestore di eventi e propagandole agli ologrammi interessati.

Capitolo 5

Introduzione di un livello di astrazione per HoloLens

Lo studio della documentazione e degli esempi forniti da Microsoft HoloLens è stato utile per comprendere le problematiche dello sviluppo di applicazioni, soprattutto per quanto riguarda l'ambito Visual Studio con la progettazione di applicazioni UWP e DirectX.

In questo capitolo verranno descritte le fasi di progettazione e realizzazione di funzionalità intermedie in grado di elevare il livello di astrazione dello sviluppo di applicazioni HoloLens, andando poi a realizzare un semplice prototipo di app per testarne il funzionamento e trarne delle preziose considerazioni.

5.1 Requisiti

L'analisi dell'applicazione HoloLens esplicativa ha dimostrato delle criticità evidenti. Le Holographic APIs fornite da Microsoft per lo sviluppo di applicazioni olografiche forniscono un supporto a basso livello, senza però introdurre il concetto di ologramma, e lasciando allo sviluppatore l'unica possibilità di un utilizzo diretto di DirectX per quanto riguarda la realizzazione degli effetti grafici. Prima di poter progettare un'applicazione HoloLens vera e propria è necessario realizzare uno strato intermedio di API che si appoggi sulle Holographic APIs di Microsoft ma che fornisca un livello di contenuti maggiore, in modo da permettere uno sviluppo ad un livello di astrazione più elevato, evitando l'interazione diretta con le API di basso livello.

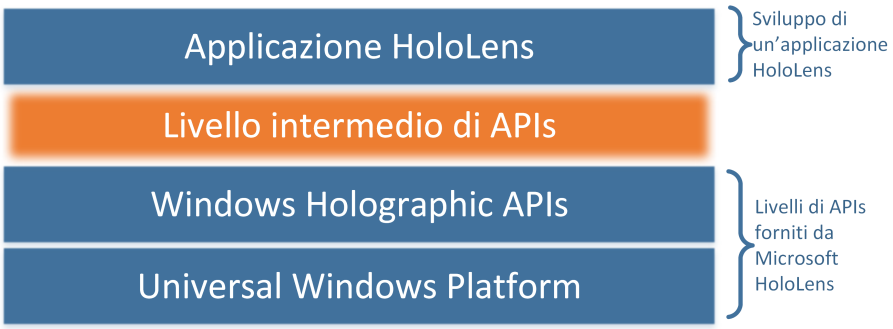

Figura 5.1: Collocazione dello strato intermedio nello stack di API

In Figura 5.1 sono mostrati i livelli di API rilevanti per la progettazione di un'applicazione HoloLens. Microsoft HoloLens pone come base di partenza la piattaforma Universal Windows Platform, comune a tutti i dispositivi che usufruiscono dei servizi del sistema operativo Windows 10, e la integra con le Windows Holographic APIs che forniscono le funzionalità olografiche utilizzabili. L'obiettivo è quello di realizzare uno strato che sia in grado di rielaborare le API di livello inferiore e di fornire delle funzionalità olografiche di base permettendo di sviluppare applicazioni HoloLens ad un livello di astrazione più elevato.

Gli obiettivi del progetto possono essere riassunti nel modo seguente:

- introdurre il concetto di ologramma, operazione sicuramente fondamentale in un sistema olografico;
- effettuare una separazione netta fra modello e rappresentazione;
- delegare al singolo ologramma l'implementazione della reazione agli input, come suggerito anche da HoloLens nella relativa documentazione;
- ottimizzare la gestione degli ologrammi, attraverso la creazione di un'entità incaricata di tenere traccia dei vari oggetti e delle loro interazioni.

5.2 Progettazione

Nella precedente fase di analisi dei requisiti sono stati individuati i concetti chiave del progetto che si desidera realizzare. Prima di procedere con l'implementazione, è necessario soffermarsi su una fase di progettazione nella quale identificare le entità essenziali e le relazioni che intercorrono fra loro.

5.2.1 Individuazione delle entità

Esaminando gli obiettivi predisposti nella fase di analisi sono state individuate sette entità distinte. Ognuna di queste è caratterizzata in modo specifico attraverso uno stato e un comportamento proprio, ed è necessaria per il raggiungimento degli obiettivi prefissati.

Si va ora ad analizzare ognuna delle entità introdotte, descrivendone le caratteristiche principali.

HologramObject Quest'entità rappresenta un semplice oggetto, descritto da caratteristiche generali quali visibilità, posizione e colore. HologramObject individua il componente base del progetto e non contiene alcun riferimento ad elementi olografici. Viene poi fornita la possibilità di creare un ologramma prendendo come base di partenza un oggetto HologramObject e conferendogli delle proprietà aggiuntive. In questo modo è possibile costruire ologrammi diversi che condividono una stessa base comune.

Hologram Quest'entità rappresenta un ologramma. Più precisamente consiste in un oggetto al quale viene applicato il concetto di realtà aumentata. Hologram rappresenta solamente la parte di modello di un ologramma e non considera alcun aspetto della parte di vista, lasciandone la completa gestione all'entità HologramView. In questo modo è possibile effettuare una netta separazione fra i concetti di modello e vista, come imposto nei requisiti del progetto. Un ulteriore vincolo richiesto nella fase di analisi consiste nell'assegnare la gestione degli input rilevati ai singoli ologrammi, quindi Hologram dovrà necessariamente implementare un suo comportamento interno che gli permetta di agire in modo autonomo alle operazioni di gestures rilevate dal sistema.

HologramView Se Hologram rappresenta il modello di un ologramma, HologramView cattura tutti gli aspetti della parte di vista. Questa entità ha il compito di generare la visualizzazione di un ologramma, basandosi sullo stato dell'oggetto, individuato dal modello. Attraverso la realizzazione di due oggetti distinti come Hologram e HologramView è possibile costruire tipologie di viste distinte per uno stesso ologramma, oltre a concepire la possibilità che un ologramma non abbia una corrispondente rappresentazione grafica.

HologramsManager Questa entità ha il compito di gestire gli ologrammi presenti nella scena. Si rivela fondamentale per coordinare le operazioni dei vari componenti del sistema. In particolar modo svolge un ruolo fondamentale nella

gestione degli input rilevati, determinando per ciascuno di essi l'ologramma interessato.

Gesture Questa entità consente di classificare la tipologia di gesture rilevata. Nel contesto del progetto le gesture rappresentano l'unica forma di input considerata, in quanto si è deciso di tralasciare la gestione dei comandi vocali. La gestione di questa entità assume, quindi, un'importanza notevole per permettere un'esperienza interattiva.

InputHandler La gestione delle gesture è resa possibile dall'utilizzo di un InputHandler. Questa entità è in grado di rilevare gli input del sistema e di interpretarli in modo da poter effettuare una classificazione della gesture.

HologramBehaviour Questa entità rappresenta il comportamento tenuto da un ologramma nel momento in cui deve reagire ad una determinata gesture. Si è deciso di esprimere questo concetto in maniera indipendente dall'entità Hologram in modo da poter definire un comportamento diverso per ologrammi dello stesso tipo.

5.2.2 Analisi delle interazioni fra le entità

Nella prima fase di progettazione sono state individuate le entità necessarie per il raggiungimento degli obiettivi posti durante l'analisi dei requisiti. Il passo successivo consiste nella definizione precisa di tutte le relazioni esistenti fra i componenti del sistema. Questo processo è fondamentale per definire lo stato ed il comportamento che dovranno assumere le corrispondenti classi in fase di implementazione.

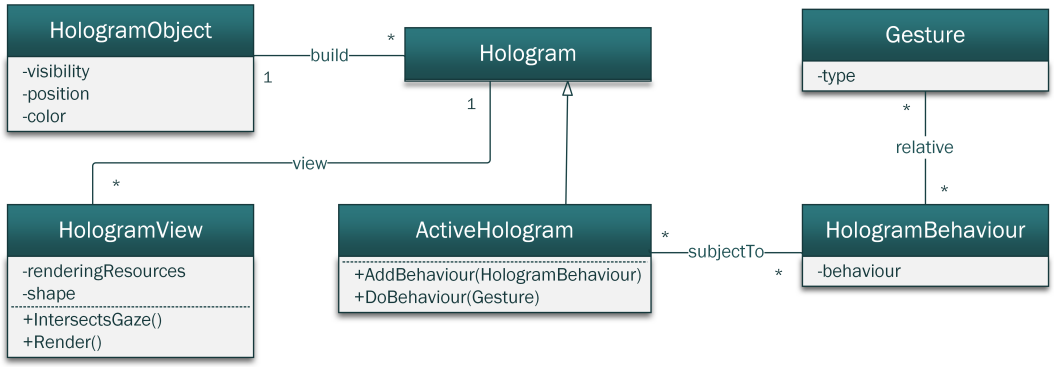

Figura 5.2: Diagramma delle classi relativo a Hologram

La classe Hologram, in Figura 5.2, riveste un ruolo centrale nel contesto in esame e costituisce un punto di partenza importante per esaminare le interazioni tra i vari componenti del sistema.

È possibile notare come la classe Hologram sia costruita mediante l'ausilio obbligato di una classe HologramObject, rappresentante un oggetto basilare. Un oggetto HologramObject non è, però, obbligatoriamente vincolato all'associazione con un Hologram, infatti, come anticipato precedentemente, è possibile definire un oggetto privo di ologramma. Così come è permesso ad un oggetto di avere più ologrammi associati, in modo da essere caratterizzato in modalità diversa a livello olografico.

Un'istanza della classe Hologram rappresenta solamente la parte di modello di un ologramma. È possibile creare una o più viste di un ologramma tramite la classe HologramView, che attraverso l'utilizzo di risorse specifiche di rendering è in grado di fornire una certa rappresentazione dell'oggetto in esame, esaminando lo stato dell'HologramObject associato all'Hologram relativo. Per permettere di identificare gli ologrammi con i quali vuole interagire l'utente sarà necessario che HologramView fornisca un metodo IntersectsGaze in grado di determinare se la vista dell'ologramma viene intersecata dal vettore del gaze. Un istanza della classe Hologram non deve necessariamente avere una propria rappresentazione grafica, mentre un oggetto HologramView corrisponde ad uno e uno solo oggetto Hologram.

La classe Hologram consiste in una descrizione generale del concetto di ologramma, per esaminare la tipologia di ologrammi che implementano una reazione alle gesture dell'utente viene introdotta una sua estensione, ActiveHologram. Questa sottoclasse deve essere posta nella condizione di poter garantire una reazione alle gesture rilevate dal sistema. Per permettere ciò può essere effettuata un'associazione con un oggetto HologramBehaviour. L'associazione non è obbligatoria in quanto è necessario garantire la possibilità di utilizzare anche ologrammi che non necessitano di reagire alle gestures in input. Un istanza della classe HologramBehaviour descrive il comportamento adottato dall'ologramma nel momento in cui viene rilevata una determinata gesture. Un oggetto ActiveHologram potrebbe anche interagire con più oggetti HologramBehaviour, in modo da settare il comportamento dell'ologramma in relazione alla rilevazione di diverse gesture. Il metodo AddBehaviour di ActiveHologram consente di aggiungere un comportamento specifico all'oggetto, mentre DoBehaviour si limita ad eseguire il comportamento associato ad una determinata gesture rilevata.

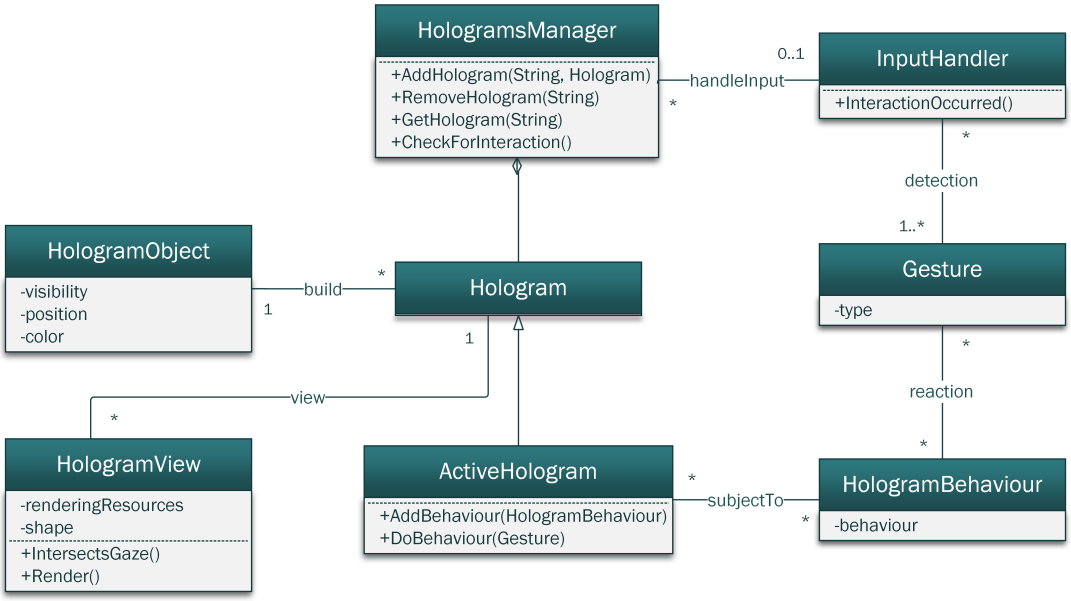

Figura 5.3: Diagramma delle classi completo

Dopo aver esaminato il caso specifico di Hologram, la comprensione dell'intera architettura risulta più semplice e immediata. In Figura 5.3 è mostrato il diagramma delle classi completo, dal quale sono intuibili le relazioni esistenti fra le varie entità presenti nel sistema. Rispetto al diagramma precedente si ha l'introduzione delle classi HologramsManager e InputHandler.

HologramsManager si presenta come un'aggregazione di Hologram. È quindi composto da un'insieme di ologrammi che assumono un significato proprio anche al di fuori del contesto in esame, ponendo le basi per una differenza sostanziale rispetto al concetto di composizione. Un ologramma può essere aggiunto al manager tramite il metodo AddHologram che richiede anche un identificativo univoco da associare all'ologramma. Questo id viene poi utilizzato nelle operazioni GetHologram, che restituisce l'ologramma richiesto, e RemoveHologram, che indica che il manager non dovrà più tenere traccia di tale ologramma. Inoltre è presente un metodo CheckForInteraction per effettuare la funzionalità di gestione degli input.

Un oggetto HologramsManager può, opzionalmente, fare riferimento ad un InputHandler. In questo modo può essere informato sulla rilevazione degli eventi di input. Questa associazione diventa fondamentale nel momento in cui gli ologrammi gestiti da HologramsManager sono progettati per implementare una gestione agli input esterni. Il metodo InteractionOccurred permette di controllare lo stato di rilevazione di input.

5.3 Implementazione

Ognuna delle classi individuate nella fase di progettazione è stata rielaborata e sviluppata in linguaggio c# utilizzando le API di base fornite da Microsoft. L'unica eccezione è costituita dall'entità rappresentante una gesture, per la quale è già presente un'adeguata implementazione nelle Windows Holographic APIs.

HologramObject L'implementazione di `HologramObject` risulta immediata, attraverso una classe con gli attributi di visibilità, posizione e colore, accessibili dall'esterno tramite le corrispondenti proprietà.

```
class HologramObject
{
    private bool visibility;
    private Vector3 position;
    private Vector3 color;

    public HologramObject(bool visibility, Vector3 position, Vector3
        color)
    {
        this.visibility = visibility;
        this.position = position;
        this.color = color;
    }

    public Vector3 Color
    {
        get { return this.color; }
        set { this.color = value; }
    }

    public Vector3 Position
    {
        get { return this.position; }
        set { this.position = value; }
    }

    public bool Visibility
    {
        get { return this.visibility; }
        set { this.visibility = value; }
    }
```

```
}
```

Hologram Per quanto riguarda il modello di ologramma si è scelto di procedere con l'implementazione di una classe astratta molto generale, rappresentante un generico ologramma relativo ad un oggetto `HologramObject` fornito come parametro.

```
abstract class Hologram
{
    private HologramObject hologramObject;

    public Hologram(HologramObject hologramObject)
    {
        this.HologramObject = hologramObject;
    }

    public HologramObject HologramObject
    {
        get { return this.hologramObject; }
        set { this.hologramObject = value; }
    }

}
```

Dopo la definizione della classe generale, è stata realizzata una sua estensione, chiamata `ActiveHologram`, che rappresenta un ologramma con la capacità di reagire autonomamente agli input utente. Questa modalità di gestione degli eventi è stata attuata in quanto è quella consigliata da Microsoft nella documentazione HoloLens ufficiale. `ActiveHologram` eredita tutti gli elementi di `Hologram` e tiene traccia di un oggetto di tipo `Dictionary` che crea una corrispondenza diretta fra una `SpatialGestureSettings` (utilizzata come chiave della lista) e un `HologramBehaviour`. La principale innovazione rispetto alla classe padre `Hologram` risulta essere l'introduzione del metodo `DoBehaviour` che accetta in ingresso una certa gesture ed esegue il corrispondente comportamento.

```
public void DoBehaviour(SpatialGestureSettings gesture)
{
    HologramBehaviour behaviour;
    if (this.gestureBehaviour.TryGetValue(gesture, out behaviour))
    {
```

```
        switch (behaviour)
        {
            case HologramBehaviour.ChangeColor:
                this.HologramObject.Color =
                    RGBColors.GetRandomColorInRGB();
                break;
            case HologramBehaviour.Ghost:
                this.HologramObject.Visibility =
                    !this.HologramObject.Visibility;
                break;
            case HologramBehaviour.Move:
                this.HologramObject.Position =
                    this.HologramObject.Position + new Vector3(0.0f,
                    0.0f, -1.0f);
                break;
            default:
                break;
        }
    }
}
```

In questo caso i comportamenti implementati sono basilari con delle leggere modifiche mirate alla struttura dell'`HologramObject` corrispondente.

HologramView Nel contesto relativo alla vista degli ologrammi è stata realizzata un'interfaccia generale che permetta di identificare un componente in grado di effettuare il processo di rendering del modello presente in `Hologram`.

```
interface IHologramView : IDisposable
{
    Hologram Hologram { get; set; }

    bool IntersectsGaze(SpatialPointerPose pointerPose, out float
        distance);

    void Render();

    void CreateDeviceDependentResourcesAsync();

    void ReleaseDeviceDependentResources();
}
```

L'interfaccia `IHologramView` implementa `IDisposable`, un'interfaccia che of-

fre un meccanismo per il rilascio delle risorse non gestite. `IHologramView` stabilisce che tutti gli oggetti che la implementano debbano fornire una definizione della proprietà in lettura e scrittura dell'ologramma contenuto. Inoltre è necessario implementare il metodo `Render`, nel quale verrà effettuata la rappresentazione grafica dell'oggetto. Siccome si parla della vista di un ologramma, si rivela fondamentale anche l'implementazione del metodo `IntersectsGaze`, che dato come parametro l'oggetto `SpatialPointerPose` che identifica posizione e direzione del gaze, stabilisce se esiste un'intersezione fra lo sguardo dell'utente e la rappresentazione grafica dell'ologramma. Sono presenti altri due metodi utilizzati per la gestione delle risorse di rendering, nello specifico `CreateDeviceDependentResourcesAsync` per la creazione delle risorse, e `ReleaseDeviceDependentResources` per il loro rilascio.

Nel contesto specifico del progetto questa interfaccia è stata implementata da `HologramRenderer`, una classe che rappresenta l'ologramma fornito in input come un cubo. Nella scrittura della classe in esame, per la parte riguardante il processo di rendering, si è utilizzato il codice presente in `SpinningCubeRenderer`, nell'esempio fornito da Microsoft HoloLens e discusso nel capitolo precedente.

HologramsManager `HologramsManager` è l'entità incaricata di gestire gli ologrammi presenti nel sistema. Contiene al suo interno due oggetti di tipo `Dictionary` che permettono di risalire in modo immediato a modello e vista degli ologrammi gestiti. La chiave di entrambi gli oggetti è un id composto da un testo univoco in grado di identificare un certo ologramma. Si è rivelato necessario utilizzare due liste diverse per `Hologram` e `IHologramView` in quanto un oggetto di tipo `Hologram` potrebbe non avere una corrispondente vista. È possibile stabilire facilmente la relazione fra un ologramma e la sua vista in quanto entrambi saranno identificabili nelle rispettive liste tramite la medesima chiave. Relativamente ad entrambe le liste, sono presenti vari metodi attraverso i quali possono essere effettuate dall'esterno operazioni di lettura, aggiunta e cancellazione di un elemento, individuato tramite il suo id.

Uno degli aspetti chiave di `HologramsManager` consiste nella gestione degli input, effettuata tramite il riferimento ad un `SpatialInputHandler`. Nell'implementazione del metodo `CheckForInteraction`, il manager controlla se è stata rilevata una gesture dall'input handler e, in caso affermativo, la ridirige all'ologramma opportuno. Questo metodo è stato implementato seguendo le direttive HoloLens, e dovrà essere invocato ad ogni frame dell'applicazione, costituendo una sorta di polling.

```
public bool CheckForInteraction(SpatialPointerPose pointerPose, out
    string hologramID, out SpatialGestureSettings gesture)
```

```
{
    hologramID = null;
    gesture = SpatialGestureSettings.None;

    if (this.inputHandler.InteractionOccurred(out gesture))
    {
        foreach (string viewID in this.hologramViews.Keys)
        {
            IHologramView view;
            this.hologramViews.TryGetValue(viewID, out view);
            float distance;
            float minimumDistance = float.MaxValue;
            if (view.IntersectsGaze(pointerPose, out distance))
            {
                if (distance < minimumDistance)
                {
                    minimumDistance = distance;
                    hologramID = viewID;
                }
            }
        }
        if (hologramID != null && this.GetHologram(hologramID) is
            ActiveHologram)
            (this.GetHologram(hologramID) as
                ActiveHologram).DoBehaviour(gesture);
    }
    return (hologramID != null);
}
```

`HologramsManager`, nel momento in cui viene rilevata una gesture dal gestore di eventi, deve individuare gli ologrammi oggetto dello sguardo dell'utente e, nel caso in cui siano in numero maggiore di uno, considerare solamente quello più vicino. Per effettuare ciò scorre tutta la lista degli oggetti `IHologramView`, invocando per ciascuno il metodo `IntersectsGaze` che restituisce un valore true se è stata trovata un'intersezione, altrimenti false. Nel caso in cui riceva una risposta affermativa, gli viene fornita anche la distanza fra il gaze e l'ologramma, grazie alla quale è possibile stabilire quale ologramma è interessato dalla gesture rilevata. A questo punto se tale ologramma è un `ActiveHologram` (in `HologramsManager` possono essere presenti `Hologram` di vario tipo) viene effettuato il metodo `DoBehaviour` specifico.

`HologramsManager` mette a disposizione anche il metodo `Render` che si limita a invocare tutte le rispettive funzioni `Render` degli oggetti `IHologramView` presenti nella lista delle viste.

Gesture Per quanto riguarda la classificazione delle varie gesture rilevabili si è scelto di utilizzare l'enumerazione `SpatialGestureSettings` fornita direttamente nelle Windows Holographic APIs.

Member	Value	Description
None | none	0	Disable support for gestures.
Tap | tap	1	Enable support for the tap gesture.
DoubleTap | doubleTap	2	Enable support for the double-tap gesture.
Hold | hold	4	Enable support for the hold gesture.
ManipulationTranslate | manipulationTranslate	8	Enable support for the manipulation gesture, tracking changes to the hand's position.
NavigationX | navigationX	16	Enable support for the navigation gesture, in the horizontal axis.
NavigationY | navigationY	32	Enable support for the navigation gesture, in the vertical axis.
NavigationZ | navigationZ	64	Enable support for the navigation gesture, in the depth axis.
NavigationRailsX | navigationRailsX	128	Enable support for the navigation gesture, in the horizontal axis using rails (guides).
NavigationRailsY | navigationRailsY	256	Enable support for the navigation gesture, in the vertical axis using rails (guides).
NavigationRailsZ | navigationRailsZ	512	Enable support for the navigation gesture, in the depth axis using rails (guides).

Figura 5.4: Valori presenti nell'enum SpatialGestureSettings

InputHandler La corrispondente classe implementata è `SpatialInputHandler`. Come definito in fase di progettazione, questo oggetto deve essere in grado di rilevare le gesture in input, in modo che l'oggetto `HologramsManager` sia messo in condizione di poterle segnalare agli ologrammi interessati. Per la definizione della classe è stato utilizzato il pattern singleton, in quanto è ragionevole pensare che sia possibile ottenere una sola istanza di gestore di eventi. Tale istanza è un campo privato e statico della classe, accessibile dall'esterno attraverso la proprietà statica `InputHandler`.

```
public static SpatialInputHandler InputHandler
{
    get
    {
        if (inputHandler == null)
            inputHandler = new SpatialInputHandler();
        return inputHandler;
    }
}
```

L'oggetto `inputHandler` viene inizializzato solamente durante la prima richiesta dell'istanza tramite l'esecuzione di un costruttore privato, nel quale vengono configurati i componenti necessari per la rilevazione delle gesture. In particolare al suo interno è necessario settare un `SpatialInteractionManager` che è il componente in grado di rilevare le interazioni dell'utente, e uno o più `SpatialGestureRecognizer`, ognuno dei quali è un riconoscitore di input specializzato alla trattazione di una sola tipologia di gesture. Devono essere presenti tanti `SpatialGestureRecognizer` quante sono le tipologie di gesture che si desidera gestire. Nel caso specifico in esame il riconoscimento di gesture è stato limitato alle tipologie principali, quali tap e hold. Viene fornito l'esempio di gestione dell'interaction manager e di un gesture recognizer relativo alla gesture di tap.

```
SpatialGestureRecognizer tapRecognizer = new
    SpatialGestureRecognizer(SpatialGestureSettings.Tap);
tapRecognizer.Tapped += (SpatialGestureRecognizer sender,
    SpatialTappedEventArgs args) =>
{
    interaction = SpatialGestureSettings.Tap;
    interactionOccurred = true;
};
SpatialInteractionManager interactionManager =
    SpatialInteractionManager.GetForCurrentView();
interactionManager.InteractionDetected += (SpatialInteractionManager
    sender, SpatialInteractionDetectedEventArgs args) =>
{
    tapRecognizer.CaptureInteraction(args.Interaction);
};
```

Viene creato un oggetto `SpatialGestureRecognizer` e vengono impostate le operazioni da eseguire nel momento in cui esso riconosce una gesture di tap, in questo caso si setta la tipologia di gesture rilevata. Per permettere al riconoscitore di gesture di effettuare il suo comportamento è necessario definire un oggetto `InteractionManager`, che nel momento in cui rileva una certa interazione in input la propaga al riconoscitore. Una certa gesture identificata è accessibile dall'esterno mediante il metodo `InteractionOccurred`.

HologramBehaviour L'implementazione di `HologramBehaviour` è stata effettuata tramite un'enumerazione contenente tutti i possibili comportamenti per mezzo dei quali un ologramma potrebbe reagire ad una gesture. In questo caso semplificato si è deciso di considerare solo i comportamenti che permettono all'ologramma di agire sull'oggetto base, applicando un aggiornamento delle

caratteristiche di visibilità, posizione e colore. I valori di `HologramBehaviour` corrispondenti implementati sono Ghost, Move e ChangeColor.

5.4 Testing

Terminata la fase di implementazione, si pone il problema di come poter determinare la correttezza degli elementi introdotti nel contesto HoloLens. Per eseguire dei test di funzionamento è necessario costruire un prototipo di un'applicazione che utilizzi le classi sviluppate nelle fasi precedenti. Tale prototipo sarà utilizzato esclusivamente per uno scopo di controllo e verifica.

L'azione più semplice e intuitiva consiste nella costruzione di un oggetto `HologramsManager`, al quale poi andare ad aggiungere vari `ActiveHologram` con le relative view, così da poter verificare l'accuratezza delle varie classi. In modo particolare è necessario prestare attenzione alla gestione delle gesture, per essere certi che la reazione del sistema sia quella prestabilita e avvenga in tempi ristretti.

Lo scheletro del metodo `Main` è stato mantenuto tale e quale a quello del programma di esempio esaminato nel Capitolo 4. La creazione del manager avviene durante il settaggio dello spazio olografico, nel quale viene anche definito il sistema di riferimento tramite un stationary frame of reference. Nel metodo `Update` viene invocato il `CheckForInteraction` dell'oggetto `HologramsManager`, in modo da controllare ad ogni frame se l'utente ha effettuato una gesture. Infine nel metodo `Render` viene invocato il `Render` del manager, così da effettuare una rappresentazione grafica degli ologrammi presenti.

Viene fornito l'esempio di creazione di un `HologramsManager`, indicato come attributo privato della classe `Main` e l'aggiunta allo stesso di un oggetto `Hologram`, definito con l'apporto di un `HologramObject`, e di una vista olografica, individuata tramite un'istanza della classe `HologramRenderer`.

```
this.hologramsManager = new HologramsManager();

HologramObject simpleObject = new HologramObject(true, new
    Vector3(0.0f, 0.0f, -3.0f),
    RGBColors.GetColorInRGB(Color.Green));
ActiveHologram hologramCube = new ActiveHologram(simpleObject);
hologramCube.setGestureBehaviour(SpatialGestureSettings.Tap,
    HologramBehaviour.ChangeColor);
hologramCube.setGestureBehaviour(SpatialGestureSettings.Hold,
    HologramBehaviour.Move);
```

```
this.hologramsManager.AddHologram("activeHologram", hologramCube,
    new HologramRenderer());
```

Per la definizione dell'ologramma si è preso come riferimento un semplice `HologramObject` visibile e situato inizialmente davanti all'utente. Per quanto riguarda il colore è stata utilizzata una classe di utility in grado di abbinare ad ogni colore i corrispondenti valori rgb sotto forma di vettore in tre dimensioni. Dopo aver istanziato l'ologramma è stato definito il suo comportamento in reazione alle gesture rilevate. In questo caso si tratta di una modifica del colore dell'oggetto `HologramObject` nel momento in cui viene rilevata una gesture di tap, e un aggiornamento della sua posizione nel momento in cui viene rilevata una gesture di hold. L'ologramma viene poi aggiunto all'`HologramsManager` che da ora in avanti si occuperà della sua gestione e visualizzazione.

Attraverso la definizione di vari ologrammi diversi, da aggiungere al manager, è possibile determinare il funzionamento del sistema e verificarne il comportamento. Nonostante ciò il test rimane un semplice prototipo e non può assicurare al 100% la correttezza delle classi implementate.

5.5 Considerazioni

La parte di testing ha prodotto un responso positivo con le classi implementate che si sono rivelate in grado di svolgere correttamente il loro funzionamento. L'implementazione del prototipo tramite l'utilizzo dello strato intermedio di API introdotto si è rivelata facile ed intuitiva, sono bastate poche righe di codice per sviluppare un'esperienza olografica completa, seppur semplificata. Anche se il test non può essere considerato completamento accurato, rappresenta comunque un'indicazione importante per il risultato del progetto.

Gli obiettivi esaminati nella fase di analisi dei requisiti sono stati pienamente soddisfatti. Il concetto di ologramma è stato introdotto nel sistema, prestando molta attenzione alla suddivisione fra la relativa parte di modello, contenuta nell'oggetto `Hologram`, e quella di vista, incapsulata nell'oggetto `IHologramView`. Inoltre l'avvento di un `HologramsManager` ha permesso di facilitare la gestione degli ologrammi e le loro interazioni con l'utente.

Una criticità della struttura implementata consiste nel fatto che, per come sono stati sviluppati i componenti, la logica di controllo è incapsulata all'interno di ogni ologramma. Nel caso specifico la logica di controllo è costituita dal comportamento degli ologrammi in reazione alle gesture di input. Il modello, individuato dalla struttura `Hologram` si aggiorna in modo completamente automatico in relazione ad eventi di input, trasgredendo ai principi basi del pattern model-view-controller che pone l'enfasi sull'introduzione di un control-

ler che, notificato della rilevazione di eventi di input, possa agire direttamente sul modello, aggiornandolo in maniera opportuna.

Capitolo 6

Un ulteriore livello di astrazione per HoloLens

Nel capitolo precedente è stato realizzato un primo livello di astrazione per HoloLens, in modo da rendere l'implementazione di ologrammi più semplice. Uno dei concetti esaminati consisteva nella delineazione netta della separazione fra modello e vista degli ologrammi. Una prima divisione è stata implementata nel progetto realizzato, ma non è considerata sufficiente.

In questo capitolo andremo ad esaminare l'importanza di un'ulteriore suddivisione, ad un livello superiore, di modello e vista di un oggetto, andando poi a definire e realizzare le API che consentono questo passaggio. Sarà inoltre effettuata una fase di testing per poter esprimere un giudizio sul lavoro svolto.

6.1 Collocazione di oggetti in un mondo aumentato

Nella concezione del pensiero HoloLens gli ologrammi sono definiti nell'ambiente circostante alla posizione dell'utente e assumono un'elevata importanza nel momento in cui rientrano nel suo campo visivo, o comunque nelle immediate vicinanze. Gli oggetti sono creati specificatamente per permettere ad un utente di beneficiare dei vantaggi di un'esperienza di mixed reality. Non sono concepiti dallo sviluppatore come elementi dotati di una vita propria, indipendentemente dal fatto che siano osservati o meno dall'utente[10].

È possibile pensare agli ologrammi visualizzati dal dispositivo HoloLens come a delle accurate rappresentazioni 3D di oggetti virtuali appartenenti ad un mondo aumentato. Tali oggetti hanno un loro stato interno autonomo rispetto alla visualizzazione che ne viene effettuata. Un oggetto è collocato in una

determinata posizione ed è dotato di certe caratteristiche, indipendentemente dal fatto che possa essere osservato da uno o più utenti.

In questo modo uno stesso oggetto può essere rappresentato anche da dispositivi diversi. Il suo stato ed il suo comportamento non sono soggetti a cambiamenti nel momento in cui viene osservato tramite un dispositivo HoloLens piuttosto che da un altro head mounted display che applica dei concetti di realtà aumentata leggermente differenti. Ogni dispositivo fornirà all'utente una propria rappresentazione di uno stesso oggetto, con un modello prefissato.

L'utente che utilizza un dispositivo HoloLens, dal canto suo, può interagire con i vari oggetti disposti nel mondo aumentato. Queste interazioni agiscono direttamente sull'oggetto e non sulla sua rappresentazione effettuata dal visore. In questo modo vari utenti con dispositivi differenti possono condividere una stessa esperienza olografica.

6.2 Augmented worlds

Attraverso la visione concepita e illustrata nella sezione precedente, si propone di effettuare un approccio alla progettazione e sviluppo di applicazioni olografiche che possa seguire il modello concettuale denominato "augmented worlds" (mondi aumentati), esposto nell'articolo [10].

Il concetto di augmented worlds è quello di porre degli oggetti computazionali in una posizione specifica del mondo reale, astraendo completamente dai paradigmi di programmazione utilizzati per la loro definizione. La posizione espressa deve essere relativa ad un sistema di riferimento locale specificato per il mondo aumentato.

Gli oggetti che fanno parte di un mondo aumentato sono definiti augmented entities (entità aumentate) e, oltre al concetto di posizione, assumono uno stato ed un comportamento ben definiti. Tali entità possono essere gestite da uno o più server che si preoccupano della simulazione degli augmented worlds. Possono essere definite osservabili, ed in questo caso è possibile accedere al loro stato interno, ma possono anche a loro volta essere osservatori di altre entità. Non è immediato il fatto che ad ogni augmented entity debba corrispondere una sua vista, in quanto è possibile concepire lo sviluppo di entità di controllo che permettano di agire su altre entità senza che debbano essere prese in considerazione dal punto di vista grafico.

In una visione semplificata di augmented worlds, le entità, a livello di modello, possono essere gestite tramite una visione ad oggetti, che permette di incapsulare in ciascuna di esse gli elementi relativi al loro stato e al loro comportamento, pensando ad un augmented world come al loro contenitore. Questa modellazione non si dimostra completamente efficiente per catturare aspetti

di concorrenza e di interazioni asincrone, che però non saranno esaminati in questo contesto.

I benefici relativi ad uno sviluppo orientato a all'approccio di augmented worlds sono molteplici. Per quanto riguarda il lavoro in esame, uno dei principali vantaggi consiste nella separazione fra aspetti di model, view e control, per cui si definisce il concetto di entità aumentata, rappresentante il modello, in modo separato dalla definizione della sua vista. In questo modo è possibile definire il modello di augmented entity in maniera completamente indipendente dalla loro vista, che non è presa in considerazione all'interno di un augmented world. Sarà invece compito dei dispositivi che si connettono al mondo aumentato fornire una vista del modello presente all'interno di un augmented world, in modo che l'utente possa essere informato dell'esistenza delle entità aumentate, oltre a potervi interagire.

6.3 Requisiti e funzionalità del sistema

L'idea di base è quella di sfruttare i principi di augmented worlds presentati, realizzando un mondo aumentato nel quale poter disporre vari oggetti caratterizzati da uno stato proprio, rendendo questo mondo accessibile dall'esterno. Gli oggetti devono poter essere percepiti dagli utenti che indossano un dispositivo di realtà aumentata e deve essere fornita loro anche la possibilità di intervenire sul loro stato. Immaginiamo, ad esempio, di porre un contatore, con uno stato molto semplice costituito da un conteggio, in una certa posizione del mondo; nel momento in cui un utente si trova nei paraggi deve essere in grado di visualizzare lo stato del conteggio, e magari incrementarlo attraverso l'esecuzione di una determinata gesture.

Si vuole realizzare un progetto che permetta di creare una suddivisione netta fra il modello e la vista dei vari oggetti. Tale progetto risulta composto da tre blocchi fondamentali:

- un server contenente lo stato del mondo aumentato,
- un'applicazione HoloLens lato client che sia grado di fornire una rappresentazione degli oggetti aumentati,
- un protocollo di comunicazione che permetta al client di interagire con gli oggetti presenti sul server.

Il mondo aumentato sarà implementato nel server e sarà costituito da un insieme di oggetti virtuali diversi, ognuno caratterizzato da un suo particolare stato interno. L'obiettivo del progetto non è indirizzato alla gestione di entità complesse, ma è focalizzato allo sviluppo di un sistema di astrazione. Si è

quindi deciso di limitare le tipologie di entità aumentate a semplici oggetti dotati di una caratteristiche che ne possa descrivere lo stato. Si tratta di un cubo, caratterizzato dal suo colore e di un contatore il cui stato è rappresentato da un conteggio. A livello server non è concepita una rappresentazione di questi oggetti, che sono visti solamente come entità del mondo aumentato con un determinato modello.

L'applicazione HoloLens relativa dovrà implementare solamente una rappresentazione grafica delle entità del mondo aumentato, basandosi sul modello delle stesse strutturato sul server. In questo contesto il dispositivo HoloLens rappresenta solamente una vista del mondo aumentato. Pensiamo ad una normale applicazione dotata di un'interfaccia grafica strutturata tramite un pattern model-view-controller; in quel caso è presente una GUI che osserva il modello dei dati ed agisce di conseguenza effettuando delle rappresentazioni dello stato degli oggetti. In maniera analoga il dispositivo HoloLens osserva il modello presente sul server e ne effettua una propria visualizzazione tramite l'introduzione di ologrammi.

Nel momento in cui un utente, interagendo con le entità olografiche, esprime l'intento di effettuare delle modifiche, la parte oggetto dell'aggiornamento non sarà la vista fornita da HoloLens, ma il modello dell'oggetto aumentato presente sul server. L'applicazione olografica dovrà quindi notificare al server l'interazione rilevata. A questo punto dovrà essere presente un controller lato server in grado di elaborare la richiesta e, applicando una certa logica di controllo, procedere all'aggiornamento dello stato dell'entità aumentata interessata nel modello generale.

Un aspetto fondamentale del progetto è costituito dalla comunicazione tra il server ed il client HoloLens. Può essere implementata una semplice comunicazione client-server, con il server che attende di essere contattato dai vari client e nel momento in cui riceve una richiesta si adopera per fornire una risposta al client nel minor tempo possibile. È necessario stabilire il protocollo di utilizzo a livello di trasporto, e soprattutto la struttura informativa dei messaggi che entrambi i blocchi dovranno essere in grado di comprendere e di elaborare. Inoltre la comunicazione dovrà garantire l'interoperabilità con sistemi diversi, in quanto nel nostro caso il server comunica esclusivamente con un dispositivo HoloLens, ma l'idea è quella di poter accedere al mondo aumentato anche da altri dispositivi.

Gli obiettivi principali del progetto possono essere riassunti nel modo seguente:

- creare un mondo aumentato nel quale sia possibile definire vari oggetti;
- rendere accessibile il mondo aumentato attraverso l'introduzione di un server;

- permettere ad un client HoloLens di visualizzare gli oggetti presenti nel mondo aumentato;
- permettere ad un client HoloLens di interagire con gli oggetti presenti nel mondo aumentato.

6.4 Progettazione

Nella precedente fase di analisi sono stati chiariti ed esaminati i requisiti del progetto. Si rivela necessario soffermarsi su una fase di progettazione nella quale stabilire l'architettura del sistema in modo da poter affrontare il processo di sviluppo avendo già un'idea definita sulle modalità di implementazione.

La progettazione può essere suddivisa in due parti distinte. Le architetture di server e client possono essere sviluppate indipendentemente l'una dall'altra. Sarà poi necessario effettuare un collegamento fra le due parti, stabilendo un protocollo di comunicazione comune che permetta un interfacciamento tra i due componenti. Nel caso specifico verrà implementato un protocollo di comunicazione ottimale per il server, portando poi il client ad adottare il medesimo protocollo per poter effettuare uno scambio di messaggi.

6.4.1 Architettura lato server

Il server ha il compito di realizzare il mondo aumentato nel quale collocare le varie entità.

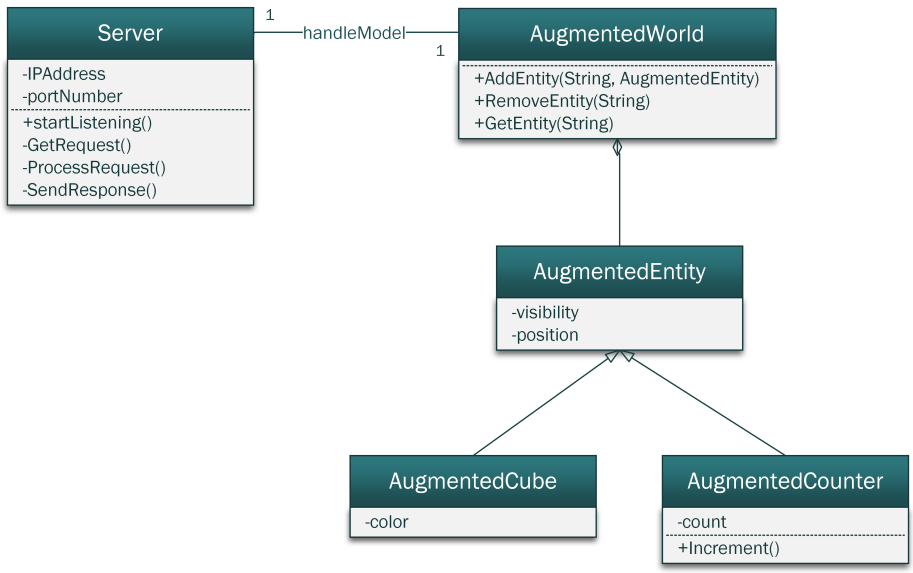

Figura 6.1: Diagramma delle classi relativo al server

In Figura 6.1 è mostrata l'architettura del sistema server. In questa fase server e mondo aumentato sono stati trattati in maniera separata in modo da chiarire i concetti di architettura, è però possibile pensare di poterli unire in fase di implementazione. In questo caso il Server è stato considerato come la classe in grado di rendere accessibile il mondo aumentato dall'esterno. Un Server è ovviamente relativo ad un solo AugmentedWorld e viceversa.

Un Server è identificato dall'indirizzo IP dell'host su cui è situato e dal numero di porta sul quale si registra. Attraverso il metodo startListening è possibile avviare il Server, in modo che si metta in ascolto sulla socket individuata da IPAdress e portNumber. Il funzionamento del server, internamente, si compone di tre funzioni principali, rilevare la richiesta del client attraverso il metodo GetRequest, elaborarla in modo da produrre una risposta tramite il metodo ProcessRequest e inviare tale risposta per mezzo del metodo SendResponse.

AugmentedWorld è costituito da un'aggregazione di AugmentedEntity. Tramite il metodo AddEntity è possibile aggiungere un'entità al mondo aumentato, inserendo anche un identificativo univoco. Attraverso questo id è poi possibile accedere all'entità con il metodo GetEntity, oppure rimuoverla tramite l'utilizzo del metodo RemoveEntity.

La classe AugmentedEntity rappresenta un oggetto generale caratterizzato da una posizione, indicata come un punto tridimensionale, e da una visibilità, che può assumere solamente valori true e false.

È presente una generalizzazione con le classi AugmentedCube e AugmentedCounter che ereditano gli attributi della classe padre, identificata dalla classe AugmentedEntity. Ognuna delle due classi figlie è caratterizzata, inoltre, da un ulteriore attributo che permette di effettuare tale classificazione. Per quanto riguarda la classe AugmentedCube l'attributo in questione è il colore, mentre per AugmentedCounter si ha un valore count che rappresenta lo stato del conteggio. L'unica operazione eseguibile relativamente all'oggetto AugmentedCounter è rappresentata dal metodo Increment che permette di incrementare il conteggio.

6.4.2 Protocollo di comunicazione

Il protocollo di comunicazione è un aspetto importante per permettere a due dispositivi di comunicare. Come in un qualsiasi sistema client-server è il client che ha il compito di iniziare la comunicazione contattando il server, in ascolto su una determinata porta.

Nel caso specifico del progetto il server è in grado di rispondere a tre tipologie di messaggi. Il client per poter garantire all'utente una totale esperienza

nel mondo aumentato dovrà essere in grado di effettuare correttamente ognuna delle tre operazioni descritte.

Join Tramite un messaggio di join il client esprime la volontà di conoscere gli oggetti del mondo aumentato. Il server risponde con un messaggio di join nel quale elenca tutte le entità gestite, indicando per ciascuna di esse le caratteristiche che la descrivono. Tipicamente il client, al momento dell'accensione del dispositivo, invierà un messaggio di join al server, in modo da poter costruire le viste relative agli oggetti del mondo aumentato.

Sync Tramite un messaggio di sync il client comunica al server gli identificativi corrispondenti alle entità di cui vuole ricevere un aggiornamento. Questa operazione verrà eseguita periodicamente dal client che potrà così rimanere aggiornato sullo stato delle entità del mondo aumentato.

In questo caso è necessario introdurre il concetto di versione. Il server assegnerà ad ogni entità aumentata una versione, nella forma di un semplice valore numerico che ne indica lo stato di aggiornamento. Nel momento in cui un'entità viene modificata in qualche modo, la sua versione cambia.

Nella risposta alla richiesta di join il server invierà al client anche la versione di ogni oggetto, permettendo al client di tenere traccia di questo valore. Il client, quando effettuerà una richiesta di sync potrà includere anche la corrispondente versione conosciuta. A questo punto il server controllerà se la versione inviata corrisponde alla versione corrente oppure no; nel primo caso si limiterà a comunicare al client che la sua versione dell'oggetto è quella aggiornata, mentre nel secondo si preoccuperà di inviare al client tutte le informazioni dell'oggetto, compresa la nuova versione. In questo modo il server riesce ad evitare operazioni di sincronizzazioni non necessarie.

Nel caso in cui l'oggetto richiesto non faccia parte del mondo aumentato, il server comunicherà questo aspetto al client, che potrà quindi evitare di continuare a tenerne traccia.

Update Tramite un messaggio di update il client notifica al server l'avvenuta rilevazione di un'azione di input effettuata dall'utente su una determinata vista. Il client dovrà quindi indicare sia l'identificativo corrispondente all'entità interessata, sia il tipo di input effettuato dall'utente; il server si adopererà per elaborare l'input e agire di conseguenza.

6.4.3 Architettura lato client

Nel nostro caso il ruolo del client è rappresentato da un dispositivo HoloLens. Per quanto riguarda la struttura dell'architettura è possibile fare riferi-

mento alla progettazione del livello intermedio di API eseguita nella Sezione 5.2. I concetti riguardanti la definizione di un ologramma sono rimasti immutati. Sono però presenti alcune leggere differenze sulla gestione dell'ologramma, in quanto non è necessario preoccuparsi degli aspetti riguardanti il modello dell'oggetto, dato che essi sono gestiti dal server. In Figura 6.1 è fornita una rappresentazione del diagramma delle classi relativo all'architettura.

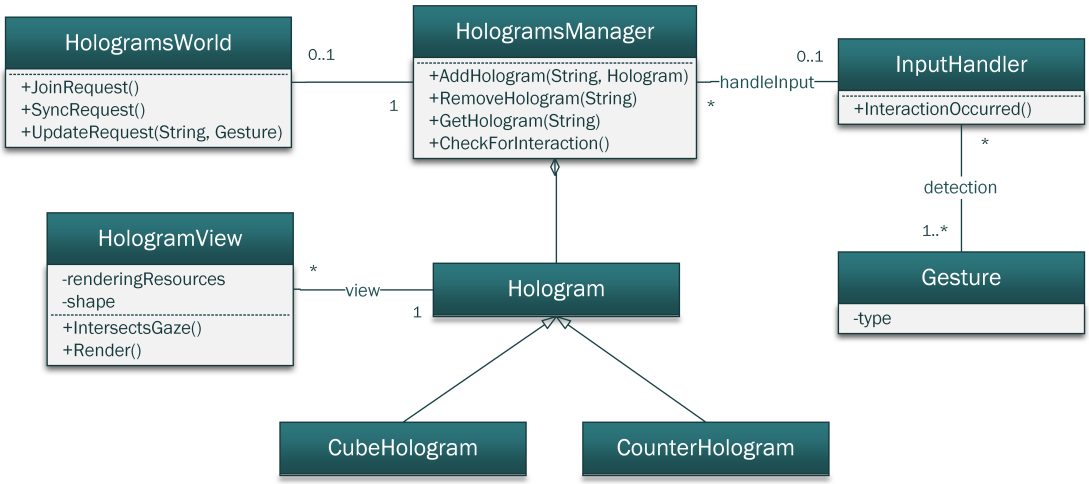

Figura 6.2: Diagramma delle classi relativo al client HoloLens

Per quanto riguarda le classi Hologram, HologramView, HologramsManager, InputHandler e Gesture è possibile fare riferimento a quanto detto in Sezione 5.2. In questo caso, però, la gestione dello stato e del comportamento delle entità aumentate è affidata al server. Non è quindi necessario l'utilizzo della classe HologramObject. Un ologramma sarà, infatti, costruito mediante i dati e le informazioni presenti nel modello del server. Non ha nemmeno senso parlare di comportamento di un ologramma, in quanto non è possibile modificare lato client lo stato dell'oggetto, dato che l'unica entità in grado di agire sul modello è il server.

Sono state introdotte CubeHologram e CounterHologram, due classi figlie di Hologram corrispondenti alle rispettive AugmentedCube e AugmentedCounter definite lato server.

Il ruolo di HologramsManger rimane comunque fondamentale soprattutto per la gestione degli input. Infatti nel momento in cui un oggetto InputHandler rileva una gesture in input è necessario che il manager identifichi l'ologramma oggetto dell'interazione in modo da poter comunicare un quadro completo al server che potrà poi reagire di conseguenza modificando il modello secondo la sua logica di controllo interna.

La classe HologramsWorld riveste un ruolo chiave nel progetto ed è incaricata di gestire l'interfacciamento fra il mondo olografico presente lato client ed il mondo aumentato costruito lato server. Dovrà quindi preoccuparsi di effettuare una comunicazione con il server in modo da essere in grado di ricavarsi il modello degli oggetti del mondo aumentato e di segnalare al server eventuali gesture di input rilevate dalla classe HologramsManager. La classe HologramsWord permette di comunicare con il server eseguendo tutti i tipi di richiesta individuati nella fase di definizione del protocollo di comunicazione. Può effettuare una richiesta di join attraverso il metodo JoinRequest, una richiesta di sync per mezzo del metodo SyncRequest ed infine una richiesta di update tramite UpdateRequest. In quest'ultimo caso è necessario indicare, oltre alla Gesture rilevate, anche l'ologramma sul quale è stata effettuata l'interazione, individuandolo tramite il suo identificativo univoco.

6.4.4 Interazioni fra i componenti

Vengono esaminate nel dettaglio le interazioni fra i componenti principali del sistema attraverso dei diagrammi di sequenza. Tali diagrammi sono stati semplificati, in modo da illustrare le interazioni fra i componenti nella maniera più chiara possibile.

Si esaminano le tre possibili richieste effettuabili dal client: join, sync e update. Le classi HologramsWorld e AugmentedWorld rappresentano i componenti che permettono di interfacciare client e server.

Join In Figura 6.3 viene presa in esame una semplice richiesta di join.

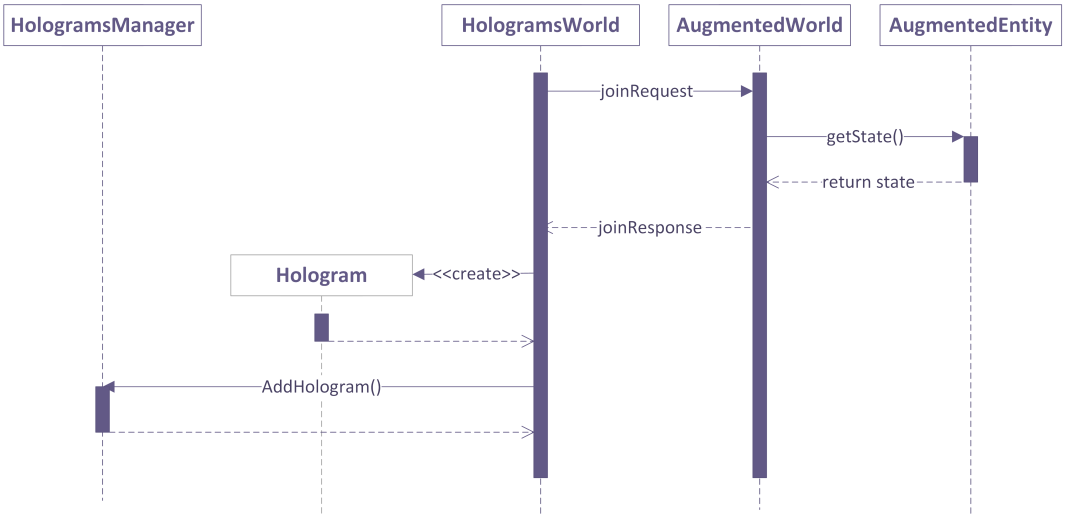

Figura 6.3: Diagramma di sequenza relativo ad una richiesta di join

HologramsWorld effettua una richiesta di join al server, rappresentato da AugmentedWorld, che si adopera per reperire tutte le informazioni di stato delle AugmentedEntity trattate. Terminata la raccolta dei dati risponde al client con tutte le informazioni richieste. HologramsWorld a questo punto crea gli oggetti Hologram relativi alle AugmentedEntity ricevute, andandole poi ad aggiungere a HologramsManager.

Sync In Figura 6.3 viene presa in esame una richiesta di sync semplificata, che non tiene traccia del concetto di versione.

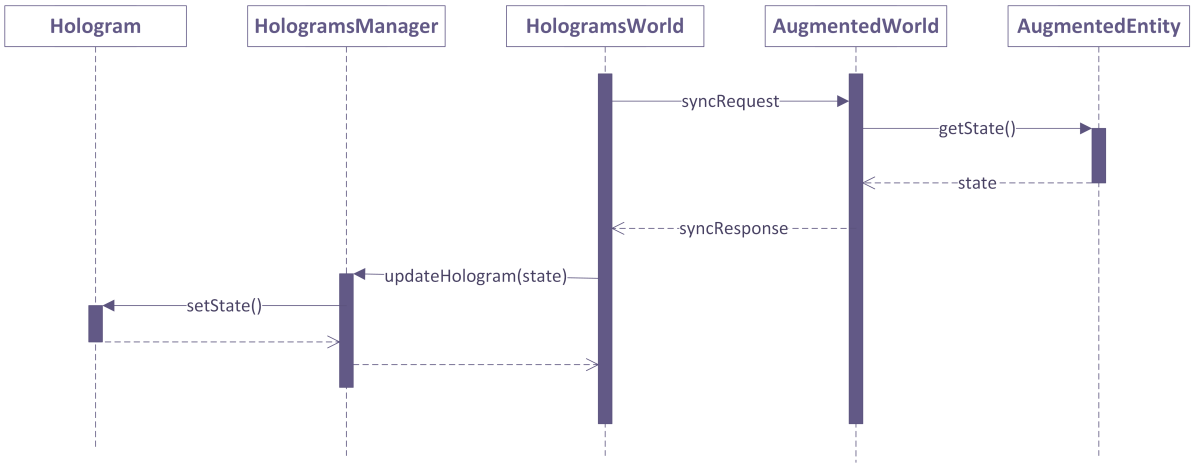

Figura 6.4: Diagramma di sequenza relativo ad una richiesta di sync

HologramsWorld effettua una richiesta di sync al server, rappresentato da AugmentedWorld, che, in maniera analoga ad una richiesta di join, reperisce i dati relativi alle entità aumentate presenti e li trasmette al client. A questo punto, HologramsWorld propaga queste informazioni al manager che procederà all'aggiornamento dei dati di Hologram.

Update In Figura 6.3 viene presa in esame una richiesta di update.

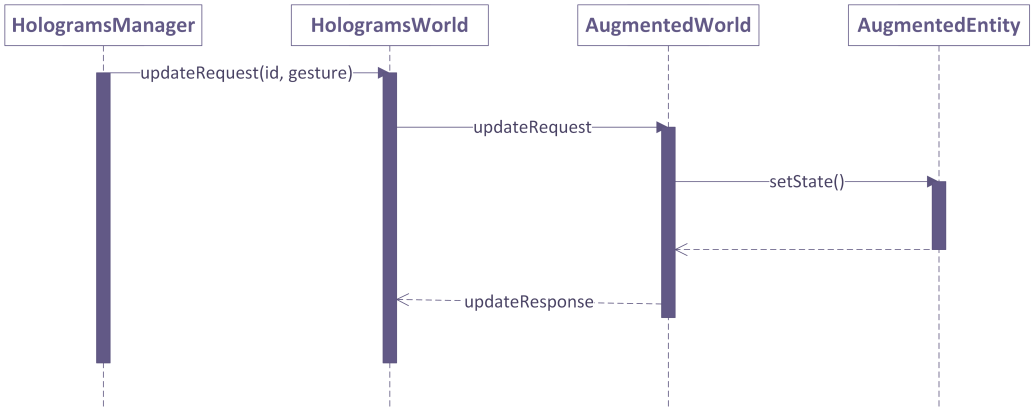

Figura 6.5: Diagramma di sequenza relativo ad una richiesta di update

Nel momento in cui HologramsManager rileva una gesture la notifica a HologramsWorld, fornendogli anche l'identificativo dell'ologramma sulla quale è stata effettuata. A questo punto HologramsWorld effettua una richiesta di update al server, includendo le informazioni acquisite da HologramsManager. AugmentedWorld provvederà a modificare lo stato di AugmentedEntity in modo adeguato, per poi mandare una risposta di conferma al client.

6.5 Implementazione

Dopo la fase di progettazione nella quale sono stati individuati gli elementi principali del progetto, si procede con il processo implementativo nel quale vengono prese anche decisioni sugli strumenti di sviluppo da utilizzare. Si è deciso di sviluppare l'implementazione del server in Java, mentre per la parte client si è utilizzato c# con l'appoggio delle Windows Holographic API.

6.5.1 Sviluppo lato server

Il server, per definizione, deve essere sempre attivo e in ascolto sulla porta dichiarata, in modo da essere raggiungibile in qualunque momento dai client. Deve inoltre essere in grado di gestire connessioni multiple, in quando potrebbe essere contattato contemporaneamente da più client. Il server dovrà quindi essere concorrente e si è deciso di utilizzare TCP come protocollo a livello di trasporto. Inoltre per favorire l'interoperabilità viene richiesta una codifica JSON per i messaggi della comunicazione.

L'implementazione è avvenuta sfruttando l'individuazione dei blocchi principali avvenuta in fase di progettazione. È stato necessario aggiungere un

controller in grado di incapsulare la logica di controllo e agire nella maniera opportuna direttamente sul modello.

AugmentedEntity L'implementazione è stata mirata a favorire l'estendibilità del sistema. In modo particolare è stata definita un'interfaccia generale per un'entità aumentata con le funzionalità minime che dovrà fornire un oggetto aumentato.

```
public interface IAugmentedEntity {

   int getVersion();

   Point3D getPosition();

   boolean getVisibility();

   void setPosition(Point3D position);

   void setVisibility(boolean visibility);

   void gestureDetected(String gesture);

   JSONObject getJSONState() throws JSONException;

}
```

Si tratta per lo più di metodi di get e set relativi alle caratteristiche base di un'entità aumentata. Fanno eccezione i metodi `gestureDetected`, nel quale dovrà essere predisposto il comportamento ad una gesture effettuata sull'oggetto, e `getJSONState` che costruisce un oggetto JSON contenente tutte le informazioni riguardanti il modello dell'oggetto.

L'interfaccia è stata poi implementata da `AbstractAugmentedEntity`, una classe astratta che si limita alla definizione dei metodi previsti dall'interfaccia, senza aggiungere alcun elemento specifico. Nei metodi di set si va a modificare l'oggetto, in questo caso è necessario eseguire un metodo `setUpdate` che provvede ad incrementare la versione dell'entità. Tale metodo è definito protected, in modo che sia accessibile anche dagli oggetti che andranno ad estendere la classe. Nel metodo `getJSONState` viene semplicemento creato un oggetto JSON con i valori degli attributi version, position e visibility.

Per quanto riguarda la gestione della gesture rilevate lato client si è scelto di non implementare il comportamento di ogni entità all'interno della classe, ma

di limitarsi a lanciare un evento che conterrà il riferimento all'entità interessata dalla gesture e alla gesture stessa.

```
@Override
public void gestureDetected(final String gesture) {
   this.notifyViewEvent(new UpdateViewEvent(this, gesture));
}
```

Il comportamento descritto è reso possibile dalla definizione di una classe `AugmentedEntityObservable` che permette a degli oggetti di tipo `Observers` di registrarsi su di essa e venire notificati dal metodo `notifyViewEvent` nel momento in cui viene creato un nuovo evento.

`AbstractAugmentedEntity` estende `AugmentedEntityObservable` e utilizza le funzionalità indicate per notificare agli osservatori, in modo particolare al `Controller`, che lato client è stata effettuata una gesture sulla vista corrispondente all'oggetto corrente. Nello specifico `notifyViewEvent` andrà ad invocare il metodo `handleEvent` di tutti gli observers registrati sull'oggetto. Il metodo `handleEvent` è l'unico richiesto dall'interfaccia `Observer` e richiede come parametro un evento specifico che implementi l'interfaccia generale `Event`.

La classe `AbstractAugmentedEntity` è stata poi estesa da due semplici classi, `AugmentedCube` e `AugmentedCuounter`. Entrambe aggiungono un'informazione di stato al modello generale, nel caso di `AugmentedCube` si parla del colore, mentre per quanto riguarda `AugmentedCounter` viene aggiunto un valore che tiene traccia di un conteggio, inizializzato a 0. L'unica operazione possibile su tale valore è `increment`.

```
public void increment() {
   this.count ++;
   this.setUpdate();
}
```

Il contatore incrementato subisce una modifica, è quindi necessario eseguire il metodo `setUpdate` della classe padre, che andrà poi ad aggiornare la versione dell'oggetto. Il valore di count deve anche essere aggiunto al `JSONObject` che permette di comunicare lo stato dell'oggetto.

```
@Override
public JSONObject getJSONState() throws JSONException {
   final JSONObject message = super.getJSONState();
   message.put("objectClass", "AugmentedCounter");
   message.put("count", this.count);
```

```
   return message;
}
```

AugmentedWorld `AugmentedWorld` è la classe che rappresenta il mondo aumentato nel quale sono posti gli oggetti. Nella sua realizzazione è stato utilizzato un singleton, in quanto è ragionevole pensare che possa essere presente un solo mondo aumentato.

`AugmentedWorld` ha un duplice compito, quello di tenere traccia di tutte le entità aumentate e quello di gestire la comunicazione con i vari client.

Per quanto riguarda il primo aspetto, fornisce metodi di aggiunta, lettura e cancellazione di un'entità `IAugmentedEntity` appartenente al mondo aumentato. Tali entità sono organizzate in una `ConcurrentHashMap`, che permette di evitare problemi nel caso di accessi concorrenti agli oggetti. Ogni entità è posta come valore della mappa, con la rispettiva chiave che è rappresentata da un identificativo univoco.

Per poter stabilire una comunicazione con i client, `AugmentedWorld`, definito come thread, deve fornire una listen socket disponibile in ogni momento ad accettare nuove richieste di connessione dai vari dispositivi esterni su una determinata porta. Il server deve poter garantire connessioni multiple, è quindi necessario l'introduzione di altri thread che possano occuparsi di una singola connessione. Si è deciso quindi di creare una classe privata e innestata chiamata `AugmentedWorldWorker` in grado di gestire in maniera completa una determinata comunicazione con un certo client.

Un `AugmentedWorldWorker` riceve la richiesta del client, la elabora e invia una risposta al mittente. La logica decisionale è stata però implementata nella classe principale `AugmentedWorld`. Il worker, dopo aver ricevuto la richiesta del client sarà quindi portato ad eseguire il metodo `processRequest` di `AugmentedWorld` che dato in input un `JSONObject` è in grado di fornire in output la risposta da trasmettere al client.

```
private JSONObject processRequest(final JSONObject request) throws
    JSONException
{
   switch(request.getString("messageType")) {
   case "join":
      return this.getAllEntities();
   case "sync":
      return this.sync(request);
   case "gestureDetected":
      return this.update(request);
```

```
    default:
        return this.getErrorMessage();
    }
}
```

Si esaminano quattro casi, derivati dal valore del campo "messageType" che rappresenta la tipologia di richiesta effettuata dal client:

- join: viene eseguito il metodo `getAllEntities` che scorre la mappa delle entità e crea un oggetto `JSONObject` formato dalla composizione delle informazioni di stato di ciascuna entità.

- sync: viene eseguito il metodo `sync` che esamina tutti gli oggetti richiesti dal client e per ognuno valuta lo stato della versione invitata, se corrisponde a quella attuale setta il campo di stato a "updated" nel messaggio di risposta, altrimenti lo pone a "notUpdated" e invia tutte le caratteristiche della corrispondente entità aumentata. Se invece l'oggetto non è presente all'interno del mondo aumentato setta lo stato a "missing".

- update: viene eseguito il metodo `update` che ricava dalla richiesta dell'utente l'identificativo dell'entità e la tipologia di gesture effettuata su di essa; a questo punto notifica alla corrispondente entità la presenza di una richiesta di update attraverso il metodo `gestureDetected`; costruisce poi un messaggio di conferma da inviare al client.

- se la richiesta del client non corrisponde a nessuna delle precedenti viene inviato un messaggio di errore.

Controller Il `Controller` è stato definito utilizzando il pattern creazionale singleton, in quanto, per come si è definita la sua funzione, è necessaria una sola istanza di questa classe, che è in grado di occuparsi della gestione di tutte le entità. In questo caso si è potuta effettuare questa scelta a causa del numero molto ridotto di tipologie di entità diverse.

Il Controller, che implementa l'interfaccia `Observer`, definisce l'apposito metodo `handleEvent`, eseguito nel momento in cui viene rilevato un evento di gesture effettuato lato client. Al suo interno viene recuperato l'oggetto `IAugmentedEntity` interessato, oltre alla tipologia di gesture effettuata. Nel caso specifico il comportamento del `Controller` è molto semplice e distingue quattro casi:

- se l'entità è un `AugmentedCounter` e la gesture è di tap, incrementa il contatore relativo;

- se l'entità è un `AugmentedCounter` e la gesture è di hold, cambia la visibilità dell'oggetto;
- se l'entità è un `AugmentedCube` e la gesture è di tap, cambia il colore dell'oggetto;
- se l'entità è un `AugmentedCube` e la gesture è di hold, cambia la posizione dell'oggetto, allontanandolo dall'utente.

Le azioni effettuate dal `Controller` sono basilari e sono utili per esaminare il corretto funzionamento del server. In una fase successiva è possibile ridefinire il comportamento del controller rimpiazzare queste azioni con alcune più efficienti per fornire all'utente un'interazione completa con il mondo aumentato.

EntryPoint La classe `EntryPoint` contiene il metodo `main`, eseguito al lancio dell'applicazione.

```
final AugmentedCube cube = new AugmentedCube(new Point3D(0.2, 0.0,
    -2.0), Color.BLUE);
cube.addViewObserver(Controller.getInstance());
final AugmentedCounter counter = new AugmentedCounter(new
    Point3D(-0.2, 0.0, -3.0));
counter.addViewObserver(Controller.getInstance());

final AugmentedWorld augmentedWorld = AugmentedWorld.getInstance();
augmentedWorld.addAugmentedEntity(CUBE_ID, cube);
augmentedWorld.addAugmentedEntity(COUNTER_ID, counter);
augmentedWorld.start();
```

Al suo interno vengono definite le entità, nel caso specifico le classi `AugmentedCube` e `AugmentedCounter`, con il `Controller` che si registra come osservatore di entrambi gli oggetti. Le entità sono poi aggiunte all'istanza della classe `AugmentedWorld`, così che possano essere considerate parte del mondo aumentato. Il server diventa operativo nel momento in cui viene avviato `AugmentedWorld` attraverso il metodo start.

6.5.2 Sviluppo lato client

Per l'implementazione della parte che descrive il client HoloLens si è fatto ampio utilizzo delle classi prodotte nel progetto realizzato nel Capitolo 5. Grazie all'utilizzo di queste API la stesura del codice si è rivelata molto semplice e immediata.

L'unica leggera variazione rispetto alle classi implementate nel Capitolo 5 è stata effettuata nella classe astratta `Hologram`, nella quale è stato introdotto il concetto di versione tramite l'aggiunta di un attributo privato e della relativa proprietà in lettura e scrittura. Le classi `HologramsManager`, `IHologramView`, `HologramRenderer` e `SpatialInputHandler` sono invece state soggetto ad un utilizzo diretto senza la necessità di apportarvi alcuna modifica. Per la loro implementazione si può far riferimento alla Sezione 5.3.

Hologram Le entità aumentate riconosciute dal server sono `AugmentedCounter` e `AugmentedCube`. Si è rivelato quindi necessario definire lato HoloLens delle corrispondenti tipologie di ologramma. In quest'ottica è stata realizzata l'implementazione delle classi `CounterHologram` e `CubeHologram`, entrambe dichiarate come classi figlie della classe astratta `Hologram`. Le classi risultano basilari, in quanto si limitano ad aggiungere un concetto di stato alla precedente definizione di ologramma. Viene fornita l'implementazione della classe `CounterHologram`, che differisce dalla classe padre `Hologram` solamente per l'aggiunta del campo count che tiene traccia del conteggio e della relativa proprietà in lettura e scrittura.

```
class CounterHologram : Hologram
{
    private int count;
    public CounterHologram(int version, HologramObject
        hologramObject, int count) : base(
        version, hologramObject)
    {
        this.count = count;
    }

    public int Count
    {
        get { return this.count; }
        set { this.count = value; }
    }
}
```

HologramsWorld Si procede con l'implementazione di `HologramsWorld`, la classe nella quale viene effettuata la comunicazione con il server tramite una socket TCP. Proprio a causa di questo motivo nel costruttore vengono richiesti come parametri indirizzo IP e numero di porta del server.

`HologramsWorld` definisce al suo interno un oggetto `HologramsManager` al quale delega la gestione degli ologrammi e degli input dell'utente. L'oggetto in questione è accessibile dall'esterno in sola lettura.

Gli unici metodi pubblici implementati sono `JoinRequest`, `SyncRequest` e `UpdateRequest` che permettono di effettuare uno scambio di messaggi con il server, rispettivamente di tipo join, sync e update. Ciascun metodo è contrassegnato dalla parola chiave `async` che esprime il fatto che la comunicazione avviene in modo asincrono. In ogni metodo viene inviata la relativa richiesta al server e poi viene attesa la risposta dello stesso. Il server utilizza una formattazione di tipo JSON per la codifica dei messaggi, quindi si è rivelato necessario costruire degli oggetti `Newtonsoft.Json.Linq.JObject` per poter effettuare la comunicazione.

Nel metodo `JoinRequest` viene inviata al server una richiesta di join.

```
JObject request = new JObject();
request.Add("messageType", "join");
await streamOut.WriteLineAsync(request.ToString(Formatting.None));
```

Viene creato un oggetto JSON, che in questo caso conterrà solamente l'indicazione del tipo di messaggio. Tale messaggio viene inviato al server tramite `streamOut`, il buffer di output associato alla socket. L'operatore `await` sospende l'esecuzione del metodo finché l'attività di invio del messaggio è completata. La fase successiva consiste nell'attesa della risposta del server.

```
JObject response = JObject.Parse(await streamIn.ReadLineAsync());
JArray entities = response.Value<JArray>("entities");
this.CreateHolograms(entities);
```

Nel momento in cui viene rilevata la risposta del server viene immediatamente codificata come un oggetto `JObject` e ne viene estratto il vettore JSON corrispondente alla chiave "entities". In esso saranno contenuti tutti gli oggetti del mondo aumentato presenti nel server. A questo punto viene chiamato il metodo privato `CreateHolograms` che decodifica il `JArray` e crea gli ologrammi corrispondenti alle entità aumentate.

Per ogni entità aumentata vengono decodificati gli attributi generali, viene esaminato il tipo dell'oggetto e tramite uno switch si assegnano anche i campi specifici. A questo punto si può procedere con la creazione dell'ologramma corrispondente che viene aggiunto all'`HologramsManager` fornendo anche una sua vista, identificata da un oggetto della classe `HologramRenderer`.

Nel metodo `SyncRequest` viene inviata al server una richiesta di sync.

```
JArray entities = new JArray();
foreach (string id in this.HologramsManager.HologramsQueue.Keys)
```

```
{
    JObject entity = new JObject();
    entity.Add("entityID", id);
    entity.Add("version",
        this.hologramsManager.GetHologram(id).Version);
    entities.Add(entity);
}

JObject request = new JObject();
request.Add("messageType", "sync");
request.Add("entities", entities);
await streamOut.WriteLineAsync(request.ToString(Formatting.None));
```

In questo caso, oltre al tipo di messaggio, è necessario inviare al server anche la lista degli oggetti aumentati per i quali si vuole ricevere un aggiornamento. Vengono inclusi tutti gli ologrammi presenti nell'oggetto `HologramsManager` in modo da indicare i campi di id e versione di ciascuno di essi nel messaggio. In modo analogo a quanto fatto per il metodo `JoinRequest` si attende poi una risposta dal server e in questo caso viene eseguito un altro metodo privato della classe, `UpdateHologram`, che accetta in input un vettore JSON di oggetti aumentati.

In questo metodo, per ogni oggetto viene esaminato il valore relativo alla chiave "state". Nel caso in cui il valore sia "missing" significa che l'oggetto non è più presente nel mondo aumentato e quindi si procede con la cancellazione del corrispondente ologramma dall'oggetto `HologramsManager`. Se lo stato è "updated" significa che l'ologramma è già aggiornato all'ultima versione del corrispondente oggetto aumentato, in questo caso non è necessario compiere alcuna azione. Se invece lo stato è "notUpdated" significa che l'ologramma non è aggiornato ed è quindi necessario modificare il suo stato in modo che rispecchi il relativo oggetto aumentato.

Nel metodo `UpdateRequest`, che richiede come parametri la gesture rilevata e l'id dell'ologramma interessato, viene inviata al server una richiesta di update.

```
JObject request = new JObject();
request.Add("messageType", "gestureDetected");
request.Add("entityID", hologramID);
request.Add("gesture", gesture.ToString());
await streamOut.WriteLineAsync(request.ToString(Formatting.None));
```

Oltre al campo che esprime la tipologia del messaggio vengono inseriti anche l'id dell'ologramma e la gesture effettuata, forniti in input al metodo.

Main Lo scheletro del `Main` dell'applicazione è analogo a quello presente nell'esempio HoloLens esaminato nel Capitolo 4. È possibile individuare in esso tre blocchi separati che consistono nel settaggio dello spazio olografico, e nei metodi `Update` e `Render`.

All'avvio dell'applicazione viene eseguito il metodo `SetHolographicSpace`. All'interno di questa funzione viene definito il sistema di riferimento dell'applicazione tramite un stationary frame of reference che pone l'origine del sistema di coordinate nella posizione del dispositivo al momento dell'esecuzione. Successivamente viene creata un'istanza dell'oggetto `HologramsWorld` e su di esso viene invocato il metodo `JoinRequest`, in modo da popolare l'`HologramsManager` presente al suo interno con gli ologrammi corrispondenti alle entità del mondo aumentato.

Il metodo `Update` viene eseguito ad ogni frame. In esso viene invocato il metodo `CheckForInteraction` sull'oggetto `HologramsManager` accessibile in lettura tramite `HologramsWorld`. In questo modo è possibile controllare frequentemente se vengono rilevate gesture utente. Inoltre viene effettuata una `SyncRequest`, in modo da gestire eventuali cambiamenti nel modello del mondo aumentato.

Infine, nel metodo `Render` viene effettuato il rendering della scena. Viene invocato il relativo metodo `Render` sull'oggetto `HologramsManager` relativo ad `HologramsWorld`. In questo modo viene effettuata una rappresentazione grafica degli ologrammi contenuti in esso.

6.6 Testing

Effettuare il testing di un sistema client-server può risultare alquanto complesso, e non è semplice fornire una garanzia della correttezza del sistema costruito. In questo caso si è proceduto con l'introduzione di due meccanismi lato server in modo da simulare un accesso concorrente al mondo aumentato sviluppato.

In situazioni di normale funzionamento il server sarà sempre attivo e accessibile solamente dall'esterno. Gli oggetti presenti nel server saranno modificati solamente a causa di interazioni effettuate dai vari utenti che si collegano al mondo aumentato. In questo caso, esclusivamente per effettuare la fase di testing, si è infranta questa regola, implementando via server due classi in grado di apportare delle modiche dirette al modello, in modo da simulare un accesso concorrente alle entità aumentate.

GUI È stata definita una semplice interfaccia grafica che è utilizzata solamente per la fase di testing. Infatti è necessario sottolineare ancora una volta

come il ruolo di realizzazione dell'interfaccia grafica vera e propria del sistema sia affidato esclusivamente ai dispositivi che si collegano al mondo aumentato.

La `TestGUI` implementata per il testing fornisce un'interfaccia nella quale vengono mostrati i dati di modello degli oggetti presenti, e consente di rilevare delle modifiche effettuate dal dispositivo HoloLens, oltre che permettere di andare direttamente ad aggiornare il modello. La GUI è stata definita come un osservatore del modello, in modo che possa essere notificata nel momento in cui si verifica una modifica sullo stato dello stesso. Inoltre è in grado di generare degli eventi di update, nello specifico `UpdateGUIEvent`, che verranno rilevati dal `Controller` che si occuperà poi dell'aggiornamento del modello. In questo modo, agendo sulla GUI, è possibile simulare il comportamento di un dispositivo connesso al mondo aumentato.

ControllerThread In questo caso il funzionamento è molto semplice. Nel momento in cui vengono definiti gli elementi principali, all'avvio del sistema server, viene creato un semplice thread a cui viene passato il riferimento all'oggetto `AugmentedCounter`. Il comportamento del thread è lineare e definito, ogni 5 secondi va ad incrementare il contatore dell'oggetto. In questo modo tramite il dispositivo HoloLens connesso al mondo aumentato è possibile rilevare le modifiche allo stato dell'ologramma, effettuate da un dispositivo esterno, in modo da percepire degli aggiornamenti che simulano quelli che, in una fase successiva all'installazione del sistema, saranno effettuati dai vari utenti connessi al mondo aumentato.

6.7 Considerazioni

La fase di testing ha prodotto ottimi risultati con il funzionamento del sistema che si è dimostrato efficiente e reattivo, nonostante la presenza di alcune problematiche, soprattutto per quanto riguarda la parte HoloLens.

Il sistema mostra delle buone potenzialità, può essere sicuramente sviluppato e ampliato adeguatamente in futuro.

6.7.1 Principali criticità

Il testing ha permesso di controllare l'adeguatezza del sistema in un modo molto limitato, attraverso la connessione di un solo dispositivo HoloLens al server, per mancanza di una disponibilità maggiore. Nonostante ciò, l'utilizzo concorrente di altri dispositivi è stato simulato mediante l'introduzione della `TestGUI` e del `TestControllerThread`, con esito soddisfacente. Il server ha

dimostrato un impatto positivo sul sistema, le principali criticità riguardano invece la parte relativa all'applicazione HoloLens.

Nel mondo aumentato gli oggetti sono descritti tramite una posizione relativa al mondo fisico. HoloLens è concepito per costruire degli oggetti virtuali intorno all'utente e non essere utilizzato come mezzo di interazione fra l'utente e un mondo aumentato esistente indipendentemente dal dispositivo. Il problema principale che deriva da questo concetto consiste nel sistema di riferimento. Un'applicazione HoloLens, solitamente, pone un stationary frame of reference nella posizione dell'utente al momento dell'avvio dell'applicazione. Questa azione implica che il sistema di coordinate considerato da HoloLens dipende dalla posizione dell'utente al momento dell'avvio dell'applicazione. Si trova quindi difficoltà nella gestione di entità aumentate che hanno una loro posizione fissa e che non deve essere soggetta a cambiamenti.

Nel caso specifico si è assunto che l'utente si posizioni in un punto predefinito, l'origine del sistema di coordinate del mondo aumentato, prima di avviare l'applicazione. Si prenda come esempio una stanza e si immagini che tutte le coordinate degli oggetti del mondo aumentato siano definite rispetto all'origine situata nel centro della stanza; in questo caso l'utente dovrà posizionarsi al centro della stanza prima di poter avviare l'applicazione.

Un aspetto sicuramente migliorabile riguarda la rappresentazione degli ologrammi tramite DirectX. Il supporto fornito da Microsoft su questo fronte è attualmente molto limitato e il recente concepimento delle app UWP ha contribuito in maniera deleteria alla situazione. Infatti la nuova piattaforma universale introdotta con Windows 10 ha rimpiazzato buona parte delle API preesistenti, senza conferire un adeguato livello di documentazione. A fronte di queste difficoltà non è stato possibile realizzare una visualizzazione adeguata per l'oggetto contatore, che è stato semplicemente realizzato come un cubo, su quale in corrispondenza della funzione di incremento viene applicata una rotazione pari ad un angolo di 5°.

6.7.2 Sviluppi futuri

L'architettura del server è stata strutturata in modo da favorire l'estendibilità del software. Risulta estremamente semplice la definizione di nuove tipologie di oggetti aumentati, che si limiteranno ad implementare l'interfaccia generale `IAugmentedEntity` e ad estendere la classe `AbstractAugmentedEntity` che fornisce già la gestione delle informazioni generali che caratterizzano un qualsiasi oggetto aumentato. A questo punto sarà necessario solamente implementare le caratteristiche specifiche della nuova entità e aggiungerla ad `AugmentedWorld`, che poi provvederà a renderla parte del mondo aumentato. In questo modo, tramite l'aggiunta di entità diverse è possibile popolare e ar-

ricchire il mondo aumentato, fornendo all'utente la possibilità di interagire con una vasta gamma di oggetti.

Per quanto riguarda l'applicazione client di HoloLens sarà necessario risolvere le problematiche discusse in Sezione 6.7.1, augurandosi di ricevere un supporto maggiore dal produttore del dispositivo. Inoltre un aggiornamento continuo è inevitabile per mettere in condizioni gli utenti di visualizzare e interagire anche con le ultime tipologie di entità aumentata aggiunte sul server. In questo caso sarà necessario creare una nuova classe di ologrammi, che estenderà la classe generale `Hologram`, in grado di descrivere tutte le caratteristiche dell'entità aumentata introdotta dal server.

Il sistema potrebbe ulteriormente essere esteso tramite la realizzazione di applicazioni client sviluppate appositamente per altri dispositivi. Nel nostro caso l'applicazione client è stata sviluppata esclusivamente per il mondo HoloLens, ma il server è stato sviluppato in modo generale ed è quindi in grado di comunicare con qualsiasi dispositivo.

Conclusioni

La mia valutazione sul percorso che ha portato alla stesura di questa tesi è sicuramente positiva. Il cammino intrapreso è stato lineare, con un prima introduzione dei concetti di mixed reality, per poi affrontare con maggiore sicurezza uno studio approfondito di Microsoft HoloLens, concludendo il tutto con la realizzazione di un sistema software.

L'esplorazione del mondo HoloLens si è rivelata molto interessante, con vari spunti innovativi ed alcune criticità evidenti. HoloLens costituisce una delle tecnologie più attese nel panorama globale ed essere stato tra i primi in grado di affacciarmi su questa piattaforma è sicuramente un motivo di grande soddisfazione personale. D'altro canto le difficoltà iniziali sono state molte, dovute al breve periodo di vita del gioiellino informatico targato Microsoft. Penso che HoloLens possa rappresentare il futuro, ma non è ancora pronto per definire il presente. Nonostante ciò esprimo l'intenzione di ampliare la mia conoscenza in questo contesto, augurandomi che Microsoft possa fornire in tempi brevi un supporto maggiore per gli sviluppatori, o magari addirittura rilasciare la versione completa del dispositivo.

Per quanto riguarda il modello di astrazione realizzato in campo applicativo, posso considerare in modo positivo il lavoro svolto. Infatti è stato possibile definire un livello di astrazione per HoloLens che consente di realizzare un'applicazione olografica in maniera decisamente semplificata. Inoltre la definizione della visione di mondo aumentato è un concetto che sinceramente reputo molto valido e interessante, garantendo svariati sviluppi futuri, come affermato nelle considerazioni del Capitolo 6.

Concludo dicendo che posso considerarmi complessivamente molto soddisfatto del lavoro svolto e dell'esperienza acquisita della realizzazione di questo progetto.

Ringraziamenti

Un ringraziamento particolare va alla mia famiglia, che mi ha sempre supportato nel percorso che ha portato al lavoro di tesi.

Ringrazio il Prof. Alessandro Ricci che mi ha permesso di sviluppare questo progetto e l'Ing. Angelo Croatti per la sua assistenza e disponibilità.

Ci tengo, inoltre, a ringraziare gli amici che mi hanno sostenuto in questo percorso, in particolar modo Alfredo, Gabriele, Dario, Simone e Mattia che hanno condiviso con me questa esperienza universitaria.

Bibliografia

[1] Paul Milgram, Haruo Takemura, Akira Utsumi, Fumio Kishino, *Augmented Reality: A class of displays on the reality-virtuality continuum*, Telemanipulator and Telepresence Technologies, Hari Das, Boston, USA, 1994.

[2] Gilson Giraldi, Rodrigo Silva, Jauvane C. de Oliveira, *Introduction to Virtual Reality*, LNCC, National Laboratory for Scientific Computing Scientific Visualization and Virtual Reality Laboratory, 2003.

[3] Intel Corporation, *The case for Augmented Virtuality*, https://blogs.intel.com/evangelists/2016/04/26/the-case-for-augmented-virtuality, 2016.

[4] Mark McGill, Daniel Boland, Roderick Murray-Smith, Stephen Brewster, *A Dose of Reality: Overcoming Usability Challenges in VR Head-Mounted Displays*, CHI '15 Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, ACM, New York, USA, 2015.

[5] Ronald T. Azuma, *A survey of Augmented Reality*, Presence: Teleoperators and Virtual Environments. vol. 6, no. 4, 1997.

[6] D. W. F. van Krevelen, R. Poelman, *A Survey of Augmented Reality Technologies, Applications and Limitations*, The International Journal of Virtual Reality, 2010.

[7] Jules White, Douglas C. Schmidt, Mani Golparvar-Fard, *Applications of Augmented Reality*, Proceedings of the IEEE, vol. 102, no. 2, 2014.

[8] Microsoft, *Microsoft HoloLens*, https://www.microsoft.com/microsoft-hololens.

[9] Ryan Henson Creighton, *Unity 3D Game Development by Example*, Packt Publishing, 2010.

[10] Angelo Croatti, Alessandro Ricci, *Programming Abstractions for Augmented Worlds*, AGERE! @ SPLASH 2015, International Workshop on Programming based on Actors, Agents, and Decentralized Control, Pittsburgh, USA, 2015.