ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

UN APPROCCIO PER LA MODELLAZIONE DI OGGETTI FISICI IN SISTEMI DI MIXED REALITY BASATO SU TECNOLOGIA META 2

Elaborato in
SISTEMI EMBEDDED E INTERNET-OF-THINGS

Relatore
Prof. ALESSANDRO RICCI

Co-relatore
Dott. Ing. ANGELO CROATTI

Presentata da
RICCARDO SALVATORI

Seconda Sessione di Laurea
Anno Accademico 2017 – 2018

PAROLE CHIAVE

Augmented Reality

Mixed Reality

Augmented World

Meta 2

Unity

Alla mia famiglia

Indice

Introduzione

Negli ultimi anni il continuo sviluppo della tecnologia ha permesso l'inserimento della computazione in maniera pervasiva nella vita quotidiana, prima creando ambienti connessi e intelligenti, poi portandola addirittura nelle proprie tasche. Il prossimo grande passo avanti nell'era della tecnologia è quello della realtà aumentata. Augmented Reality (AR), Mixed Reality (MR) e Virtual Reality (VR) stanno giocando infatti un ruolo fondamentale nella progettazione del nostro futuro, intensificando il modo in cui viviamo il mondo e aumentando le capacità con cui lo percepiamo. L'innovazione portata dall'AR toccherà, e cambierà radicalmente la maggior parte degli aspetti della vita quotidiana, dal lavoro alla società, sviluppando ambienti sempre più intelligenti e interattivi. Riprendendo la frase di Jon Peddie [4]: *"...to present data in the wearer's field of view will be the biggest step forward in how we play, work, learn and communicate since the internet and mobile phones..."*.

É nato e si sta evolvendo un nuovo modo d'interagire con le informazioni, che integra ed estende le enormi possibilità offerte dal mondo dell'Internet of Things e del Web of Things portando nuovi concetti e nuove realtà. AR e MR aumentano l'ambiente in cui viviamo aggiungendo uno strato digitale che è profondamente legato alla realtà percepibile, la estendono permettendoci di utilizzare al meglio l'informazione virtuale senza perdere il contatto con l'ambiente circostante.

É proprio nell'ambito dell'augmented e mixed reality che si inserisce questa tesi, prendendo come punto di riferimento il modello di Augmented World (AW). Nella costruzione di un AW è di fondamentale importanza la rappresentazione digitale degli oggetti fisici in entità aumentate all'interno del mondo. Si deve in questo senso ideare un approccio che permetta di generare e gestire un accoppiamento efficace delle due realtà: fisica e digitale. L'obiettivo principale di questa tesi è quello di offrire uno strumento per la modellazione di oggetti fisici all'interno di un AW, rappresentandoli con entità digitali dotate di posizione, estensione e stato. A questo proposito è stata sviluppata un'applicazione in Unity che sfrutta le funzionalità messe a disposizione dai moderni visori di MR e permette di modellare oggetti fisici estendendoli con un'entità digitale. La tesi verrà suddivisa in cinque capitoli: il primo capitolo servirà

da introduzione al mondo della MR. Prima verranno discusse le caratteristiche fondamentali di AR, MR e VR spiegandone le similitudini e le differenze, poi verranno presentate tre principali tecnologie per la MR: HoloLens, Meta 2 e Magic Leap One. Nel secondo capitolo verrà approfondito il concetto di AW e di Mirror World prestando attenzione alle entità e alle funzionalità che entrano in gioco nella sua modellazione. Al termine del secondo capitolo si descriverà in maniera più specifica l'obiettivo della tesi. Il terzo capitolo tratterà l'analisi dei requisiti dell'applicazione ed esporrà alcuni tra i possibili domini applicativi. Nel quarto capitolo verrà discussa la parte relativa alla progettazione e allo sviluppo dell'applicazione introducendo per la spiegazione diagrammi UML, immagini e parti di codice. Nel quinto ed ultimo capitolo verranno esposti i risultati ottenuti grazie a questo lavoro; verranno inoltre riassunte le funzionalità messe a disposizione dall'applicazione, ed in ultimo descritti i problemi emersi nell'utilizzo.

Capitolo 1

Background

In questo primo capitolo verranno esposti i concetti fondamentali che teorizzano il vasto mondo della realtà aumentata. Verranno sottolineate le diverse combinazioni virtuale/reale esposte nel *reality-virtuality continuum* e discusse le differenze e le similarità . Infine si presenterà una lista delle più importanti ed influenti tecnologie in uso per l'*augmented* e *mixed reality*.

1.1 Mixed Reality vs Augmented Reality

Un prima definizione di realtà aumentata può essere data attraverso una descrizione delle sue caratteristiche [1]:

Realtà aumentata è un qualsiasi sistema che presenta queste tre caratteristiche:

1. Coniuga un ambiente reale e un ambiente virtuale
2. É interattivo in tempo reale
3. Si sviluppa in uno spazio tridimensionale

Prescindendo quindi da tecnologie specifiche possiamo dire che un mondo aumentato vede sovrapposti oggetti digitali, appartenenti al mondo virtuale, ad oggetti fisici, appartenenti al mondo reale. Il modo in cui avviene questa integrazione ed il suo livello di profondità definiscono un vero e proprio *continuum* di realtà possibili che Milgram e Kishino [5] definiscono: *reality-virtuality continuum* (fig 1.1). Ad un estremo di questo insieme di mondi troviamo quello totalmente reale, che comprende oggetti concreti, sottoposti alle leggi fisiche, che può essere percepito attraverso i sensi corporei o visto attraverso un schermo. All'estremo opposto troviamo invece il mondo totalmente digitale, comunemente definito Virtual Reality (VR), dove il mondo fisico viene rimpiazzato da un ambiente non reale nel quale si è totalmente immersi

e nel quale è possibile interagire esclusivamente con oggetti virtuali. Ciò che non rientra nelle due realtà sopracitate viene identificato come Mixed Reality (MR) ed occupa la parte centrale della retta abbracciando tutte le possibilità di fusione tra le due realtà.

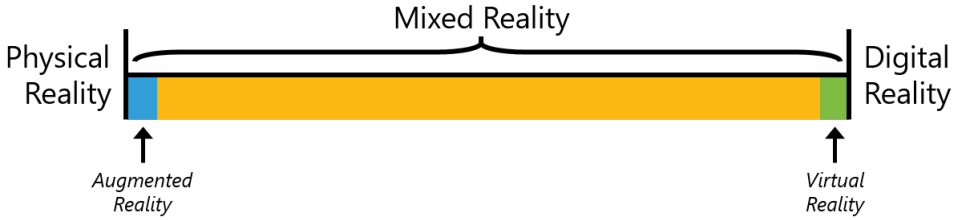


Figura 1.1: Reality-virtuality continuum

1.1.1 Augmented Reality

L'Augmented Reality (AR) è dunque un sottoinsieme del vasto mondo della MR che rimane legato alla realtà fisica ed al contempo la arricchisce sovrapponendovi contenuti digitali. Gli elementi cosiddetti "aumentanti" posso provenire da dispositivi diversi come telefoni, PC o anche elmetti e visori, e hanno la funzione di rendere l'ambiente di interesse più chiaro e utilizzabile per l'utente, aggiungendo e in alcuni casi nascondendo informazioni. L'utente ha inoltre la possibilità di percepire l'informazione digitale guardandola e modificandola in tempo reale, senza mai perdere la possibilità di vivere il mondo fisico. Nei sistemi di AR la componente digitale viene solitamente sovrapposta attraverso uno schermo al mondo fisico, l'interazione virtuale è dunque limitata e inibisce il senso di immersione nel mondo digitale.

1.1.2 Mixed Reality

La MR a differenza dell'AR ha il compito più arduo di far coesistere oggetti fisici e virtuali in modo simbiotico, così da poter interagire tra loro e creare un'unica realtà mista in cui l'utente può muoversi liberamente. Tanto più si vuole creare un'esperienza di MR soddisfacente tanto più la connessione tra mondo fisico e virtuale deve essere profonda. Sorgono a questo proposito alcuni problemi di notevole importanza, fondamentali in un'applicazione di MR:

- conoscere in maniera dettagliata l'ambiente esterno
- determinare la posizione degli oggetti virtuali e dell'utente
- vedere e manipolare ologrammi come oggetti reali e trattare oggetto reali anche come entità digitali

- gestire l'illuminazione degli elementi virtuali in maniera realistica

La soluzione a questi problemi, che deve avvenire in tempo reale, richiede potenza computazionale considerevole che spesso viene raggiunta con l'aiuto di GPU e hardware specifico.

1.2 Tecnologie

In questa sezione verranno presentate brevemente le tecnologie più rilevanti attualmente disponibili per la realtà aumentata.

1.2.1 HoloLens

HoloLens [1.2] è un dispositivo Head Mounted Display (HMD) standalone per la realtà aumentata sviluppato da Microsoft. É dotato un display ottico HD ed un sistema di scansione spaziale che permettono di interagire con il mondo digitale attraverso gesti, mani e voce. Ad oggi si classifica come una tra le tecnologie più all'avanguardia sia a livello hardware che software. Monta al suo interno un sistema operativo Windows 10 e contiene oltre alla combinazione CPU-GPU, un Holografic Processing Unit (HPU); ovvero un coprocessore che integra i dati dai vari sensori, tra cui: giroscopio, accelerometro, sensore di profondità e magnetometro, e gestisce attività come la mappatura dell'ambiente, il riconoscimento dei gesti ed il riconoscimento vocale.

Figura 1.2: Microsoft HoloLens

1.2.2 Meta 2

Meta 2 [1.4] è l'headset per la realtà aumentata sviluppato dal brand Californiano Meta.

É il secondo dispositivo commercializzato dall'azienda, che nel 2013, a seguito di una proficua campagna di croudfounding del valore di 194.000$, rilasciò il primo prototipo chiamato Meta 1 [1.3].

Figura 1.3: Headset Meta 1

L'occhiale Meta 2 è un HMD cablato che viene presentato come sostitutore dei classi schermi PC e come tale richiede di essere collegato ad un computer. La gamma di sensori di cui è dotato comprende: sensore di profondità, telecamera da 720p, tecnologia IMU[1], microfoni e casse. Permette una ricostruzione dell'ambiente, il riconoscimento di gesti e il tracciamento dello sguardo. É caratterizzato da un Field-Of-View (FOV) orizzontale di 90° notevolmente più ampio dei concorrenti HoloLens (30°) e Magic Leap One (50°).

[1]IMU o Inertial Measuring Unit è un sensore elettronico composto da accelerometro, giroscopio e magnetometro. Misura la velocità,l'orientamento e la forza gravitazionale di un dispositivo

Figura 1.4: Meta 2

1.2.3 Magic Leap One

Magic Leap One [1.5], prodotto dall'azienda Magic Leap, è l'ultimo uscito tra i dispositivi per la realtà aumentata attualmente in commercio.

Si differenzia sostanzialmente dai precedenti headset sia dal punto di vista estetico che dal punto di vista funzionale. Il dispositivo si compone infatti di tre componenti principali, il visore , chiamato Lightwear, più piccolo rispetto ai più massicci HoloLens e Meta 2, un computer indossabile, il Lightpack, dotato di CPU e GPU, e un controller manuale a 6 gradi di libertà (6DOF).

Figura 1.5: Magic Leap One

Capitolo 2

Augmented World

2.1 Introduzione

Se l'idea di trovarsi immersi in un ambiente virtuale e relazionarsi con oggetti digitali poteva sembrare bizzarra, oggi e nel prossimo futuro potrebbe diventare una sorprendente normalità.

Il continuo sviluppo della tecnologia ha permesso negli ultimi anni di aumentare le capacità sensoriali in modi finora impensabili, inserendo l'uomo in ambienti di tutti i giorni nei quali reale e digitale coesistono ad un livello sempre più profondo. L'integrazione delle funzionalità offerte dal mondo IoT e WoT con quelle messe a disposizione dai moderni visori per l'AR e MR conduce non solo ad ambienti "più intelligenti", ma anche più interattivi e maggiormente percettibili per l'utente.

La progettazione di mondi aumentati, come estensione dello spazio fisico, è quindi un aspetto fondamentale nella programmazione, che richiede nuovi paradigmi e nuovi modelli.

Molto importante in questo ambito è il concetto di *Augmented World* (AW) [3] correlato strettamente a quello di *Mirror World* (MW) [2]. In un MW lo spazio fisico viene accoppiato ad uno strato digitale che incorpora una moltitudine di agenti software dotati di stato e comportamento, capaci di agire direttamente sulla realtà ed essere percepiti e modificati dall'esterno. In uno scenario di questo tipo, nel quale si uniscono nella maniera più profonda reale e virtuale non solo si parla di estensione della capacità umane ma anche di qualsiasi entità, fisica o digitale, che compone l'ambiente.

2.2 Caratteristiche di un Augmented World

In questa sezione verranno descritte le caratteristiche principali di un AW aiutandosi con un modello concettuale, cercando il più possibile di astrarre da una sua possibile implementazione.

2.2.1 Allineamento spaziale

Ciò che contraddistingue un AW sono le entità aumentate che lo compongono. Possono essere definite come oggetti computazionali situati in una specifica posizione del mondo fisico (locale o relativa) e caratterizzate da uno stato ed un comportamento (fig 2.1). Un'entità aumentata può avere una rappresentazione 3D della sua geometria in modo da essere visibile agli utenti.

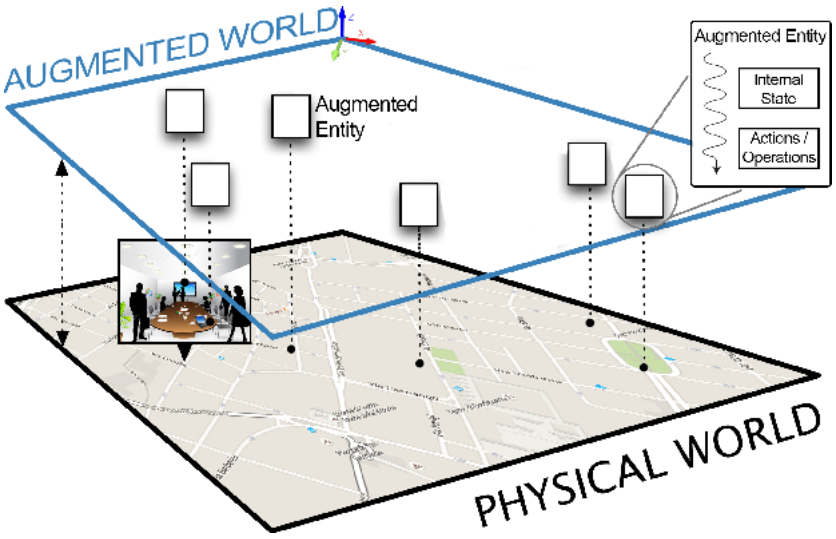


Figura 2.1: Entità aumentate in un Mirror World

2.2.2 Scoperta e osservabilità

La caratteristica spaziale delle entità può essere sfruttata per rendere possibili due tipi di meccanismi per localizzare e interagire con esse: uno di tipo *pull* e l'altro di tipo *push*. Il primo permette di ricevere il riferimento ad un'entità richiedendolo runtime all'AW, mentre il secondo sfrutta il concetto di osservabilità. Con questo si vuole intendere che un'entità viene percepita e osservata nel momento in cui rientra nel "raggio di vicinato" dell'entità osservante: ad ogni oggetto sono infatti associate una lista degli osservatori ed una degli osservanti.

2.2.3 Utenti in un Augmented World

. Generalmente un AW è un ambiente multi-utente, nel quale più utenti utilizzano e esperiscono la stessa realtà aumentata da diverse prospettive. Dal punto di vista digitale è possibile associare ad un utente una controparte virtuale che ne rappresenta lo stato e se necessario la geometria. Questa estensione fa si che anche le entità aumentate possano percepire la presenza di un utente e reagire a suoi eventi.

2.2.4 Estensione degli oggetti fisici

Il concetto di AW si estende anche agli oggetti fisici non computazionali offrendo la possibilità di dotarli di nuove capacità e funzionalità: ad esempio interfacce virtuali o geometrie che specifichino la forma dell'oggetto. Questo tipo di estensione richiede necessariamente un accoppiamento costante tra entità fisica ed entità aumentata che non è facile ottenere in maniera perfetta, non potendo a priori assumere che l'accoppiamento non subisca interruzioni nel tempo. Per perfezionare questo meccanismo è necessario dunque disporre di un meccanismo di ripristino o di controllo di attendibilità dei dati.

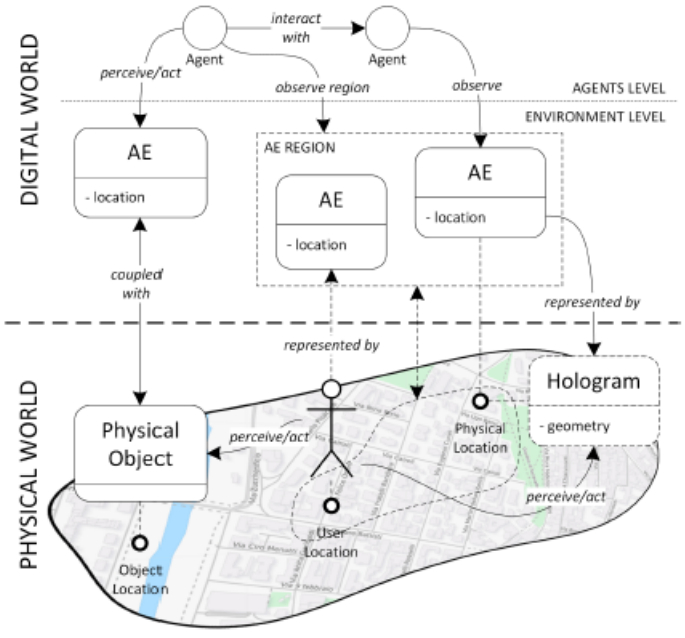


Figura 2.2: Mondo fisico e mondo digitale

2.3 Modello Concettuale

Una formalizzazione dei concetti sopra elencati può essere espressa nello schema UML in figura 2.3.

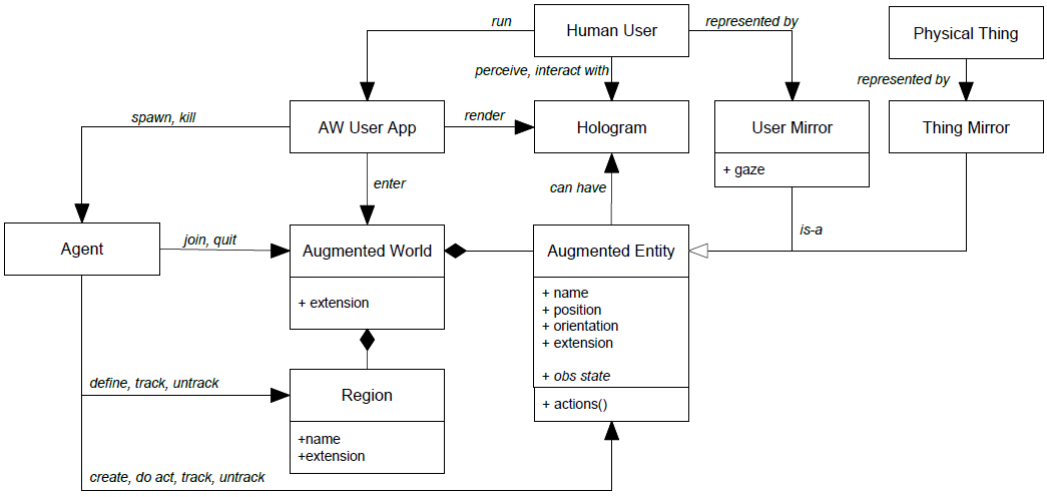


Figura 2.3: Modello concettuale di un Augmented World [3]

Un AW è formato da regioni (*Region*) delle quali sono definiti nome ed estensione.

Le componenti fondamentali di un AW sono le *Augmented Entity* (AE) che lo compongono caratterizzate da nome, posizione, orientamento ed estensione; queste rappresentano gli oggetti computazionali situati "sopra" il fisico. Un AE mostra un proprio stato (*obs state*) ed un interfaccia per le azioni (*actions*). Le AE possono essere dinamicamente create, distrutte e modificate dagli Agents in maniera dinamica.

Un *Agent* per poter compiere azione su un AW deve come prima cosa entrarci attraverso un operazione di *join*. Una volta "iscritto" un Agent può interagire con le AE attraverso le interfacce messe a disposizione e tracciarle (*track*) per conoscerne lo stato. Anche le regioni che mappano porzioni di spazio fisico posso essere tracciate da un Agent così da sapere se una particolare entità è entrata in quella porzione di spazio. Una lista delle azioni specifiche che può effettuare un Agent è riportata nella Tabella 2.1.

Le azioni possono essere suddivise in due categorie. La prima rappresenta le primitive che permettono di entrare ed uscire da AW, per interagire con un AE e per tracciare e definire le regioni. L'altra è rappresenta le funzionalità messe a disposizione dalle singole entità aumentate.

Un AE può avere associato un ologramma (*Hologram*) che la rappresenta e la rende percepibile dall'utente, in maniera indipendente dal dispositivo utilizzato per la rappresentazione. L'ologramma viene aggiornato costantemente, ogni volta che lo stato dell'entità aumentata si modifica.

Un utente (*User*) inizia una sessione all'interno di un AW attraverso una applicazione (*AWUserApp*) che ha il compito di generare l'agente all'interno dall'AW. L'applicazione ha inoltre il compito di rappresentare gli oggetti aumentati e di gestire gli input dell'utente (sguardo, gesti, comandi vocali...). Più utenti possono condividere lo stesso mondo utilizzando anche applicazioni differenti. Alcune delle AE posso rappresentare gli oggetti fisici presenti nel AW, come una sorta di specchio (*Thing Mirror*). La rappresentazione riflessa dell'oggetto definisce un modello digitale dello stato fisico di un oggetto ed evolve in relazione ad esso.

Anche l'utente, considerato come oggetto fisico, può avere una sua rappresentazione (*User Mirror*), ad esempio un augmented body. L'accoppiamento di entità aumentate e oggetti fisici con l'ausilio di specifici attuatori può, in maniera speculare, portare a degli effetti concreti su oggetti fisici scaturiti da eventi digitali.

Tabella 2.1: Azioni eseguibili da un Agent

Primitive	**Descrizione**
joinAW(name,location) : awID	per collegarsi ad un augmented world; ritorna l'id della sessione
quitAW(awID)	per dal corrente augmented world
createAE(awID, name, template, args, config): aeID	per creare una nuova augmented entity in un augmented world, specificando nome, template, parametri (dipendenti dallo specifico template), e una configurazione iniziale
disposeAE(aeID)	per eliminare un augmented entity esistente
trackAE(aeID)	per cominciare a tracciare una augmented entity
stopTrackingAE(aeID)	per smettere di tracciare una augmente entity
moveAE(aeID, pos, orientation)	per modificare la posizione e l'orientamento di una augmented entity (se permesso)
defineRegion(awID, name, region)	per definire una regione specificando nome ed estensione
trackRegion(awID, name)	per tracciare una regione
stopTrackingRegion(awId, name)	per smetter di tracciare una regione

2.4 Obiettivo Tesi

Nell'ottica dell'AW risulta di fondamentale importanza l'estensione degli oggetti fisici all'interno del mondo digitale. Dotare un oggetto di informazione digitale e mantenere coerente l'accoppiamento tra le due realtà non è però un compito semplice. L'obiettivo di questa tesi è quello di sviluppare un approccio che permetta la modellazione di oggetti fisici rappresentandoli come entità digitali di un AW. Si sfrutteranno le funzionalità messe a disposizione dal visore Meta 2 per permettere quindi tale accoppiamento.

Una funzionalità messa a disposizione dalla maggior parte dei visori di realtà aumentata è la ricostruzione dell'ambiente attraverso la generazione automatica di una mesh (fig 4.6). Utilizzando il sensore di profondità vengono individuati diversi punti sulla superficie nel campo visivo dell'utente, poi vengono progressivamente uniti in triangoli che formano la mesh. Anche se questa funzionalità risulta molto utile in diversi contesti, presenta diversi problemi e non riesce a fornire informazioni esaustive sull'ambiente modellato. In particolare la mesh non contiene nessuna informazione in riferimento ai differenti oggetti presenti nell'ambiente e non differenzia specifici oggetti fisici all'interno della ricostruzione. Quella che solitamente viene generata inoltre è una mesh statica, non adatta ad esempio a modellare oggetti che possono cambiare posizione nello spazio o che possono essere spostati e manipolati dagli utenti. Quello che si vorrebbe avere è una modellazione completa del mondo fisico, accoppiando ogni oggetto fisico ad una rappresentazione digitale che sia in ogni istante coerente rispetto all'oggetto e che possa essere manipolata dall'utente. É necessario quindi che l'entità digitale modifichi la sua posizione, e se necessario il suo stato, nel momento in cui l'utente sposta o interagisce con il relativo oggetto fisico.

L'idea è quella di sfruttare l'insieme di punti generati dal processo di ricostruzione per identificare un *bounding box* attorno all'oggetto fisico e trattarlo come un qualsiasi ologramma con cui l'utente può interagire; se viene manipolato l'oggetto fisico, ad esempio ruotato e spostato, anche la sua rappresentazione digitale verrà spostata e ruotata, mantenendo così l'accoppiamento spaziale. Una volta definita la rappresentazione virtuale di un oggetto fisico, questa può essere sfruttata per molteplici utilizzi, ad esempio per visualizzare informazione sull'oggetto fisco quando si interagisce con esso oppure se si vuole permettere l'interazione tra oggetti fisici e digitali.

Capitolo 3

Analisi del progetto

In questo capitolo verrà effettuata l'analisi del progetto. Verranno discussi quindi i requisiti funzionali e non funzionali dell'applicazione, presentando infine alcuni domini applicativi.

3.1 Analisi

Il problema della modellazione di mondi aumentati sta diventando sempre più importante nell'ottica dell'augmented e mixed reality. La creazione di ambienti intelligenti presuppone un legame profondo tra mondo virtuale e mondo fisico, che non si limita alla semplice sovrapposizione digitale-reale ma inserisce allo stesso modo virtuale nel reale e reale nel virtuale. In quest'ottica si vuole sviluppare un'applicazione di supporto allo sviluppo e alla modellazione di elementi fisici all'interno di un Augmented World. L'applicazione permetterà attraverso l'utilizzo di un visore per la realtà aumentata, di modellare la forma di oggetti fisici e dotarli di una rappresentazione digitale. Si potrà associare ad ogni oggetto presente nel mondo reale un *bounding box* creato a partire dai punti individuati nel processo di ricostruzione dell'ambiente. La forma del bounding box potrà essere scelta da un set di diverse forme base, così da avere una maggiore precisione nella creazione del "collider" dell'entità aumentata. Si potrà scegliere la tipologia di bounding box più adatta a seconda delle circostanze o, se necessario, una combinazione di più tipologie. Le principali saranno:

- *Afferrabili* Possono essere presi e spostati dall'utente.
- *Statici*: Non modificano la loro posizione; generano eventi quando l'utente li tocca o si trova vicino.
- *Osservabili*: Generano eventi quando l'utente li guarda.

- *Ruotabili*: Possono essere ruotati.
- *Scalabili*: Possono essere ingranditi o rimpiccioliti.

Per migliorare l'utilizzabilità del bounding box, dal punto di vista dell'utente, sarà possibile avere un feedback visuale dell'entità digitale, come ad esempio la visualizzazione della mesh durante lo spostamento dell'oggetto fisco. Infine si darà la possibilità di salvare un ambiente virtuale, specificandone il nome, ed in seguito caricarlo , scegliendolo dalla lista degli ambienti precedentemente modellati. Nel caso in cui gli oggetti vengano spostati durante una sessione la posizione delle entità digitali viene aggiornata e quindi l'accoppiamento mantenuto anche al caricamento successivo.

3.2 Dominio applicativo

L'applicazione verrà utilizzata prevalentemente per gestire oggetti presenti in ambienti indoor come ad esempio un ufficio o una sala, che si vuole rendere interattivi attraverso la creazione di una realtà aumentata. La modellazione dell'ambiente avviene individuando gli oggetti fisici, le tipologie e le loro proprietà. Gli oggetti più distanti dall'utente, non immediatamente raggiungibili attraverso l'utilizzo delle mani, potrebbero essere accoppiati con entità aumentate attivabili con lo sguardo. Ad esempio un interruttore della luce che, con apposti attuatori, potrebbe essere acceso semplicemente guardandolo. Oggetti fisici potrebbero assumere il ruolo di interruttori digitali e viceversa. La possibilità di avere una rappresentazione digitale degli oggetti fisici permette inoltre di programmare agenti che agiscono nell'ambiente sfruttando queste informazioni per muoversi in maniera intelligente all'interno nell'ambiente, differenziando il proprio comportamento sulla base dell'oggetto fisico.

Capitolo 4

Progettazione e Sviluppo

In questo capitolo verrà discussa la parte relativa alla progettazione e allo sviluppo dell'applicazione. Verrà esposta l'architettura usando il formalismo UML. Nella parte di sviluppo verranno inoltre approfondite le principali tecnologie utilizzate e mostrate alcune le sezioni di codice significative.

4.1 Progettazione

In questa sezione verrà discussa la progettazione dell'applicazione attraverso diagrammi dei casi d'uso e delle classi.

4.1.1 Casi d'uso

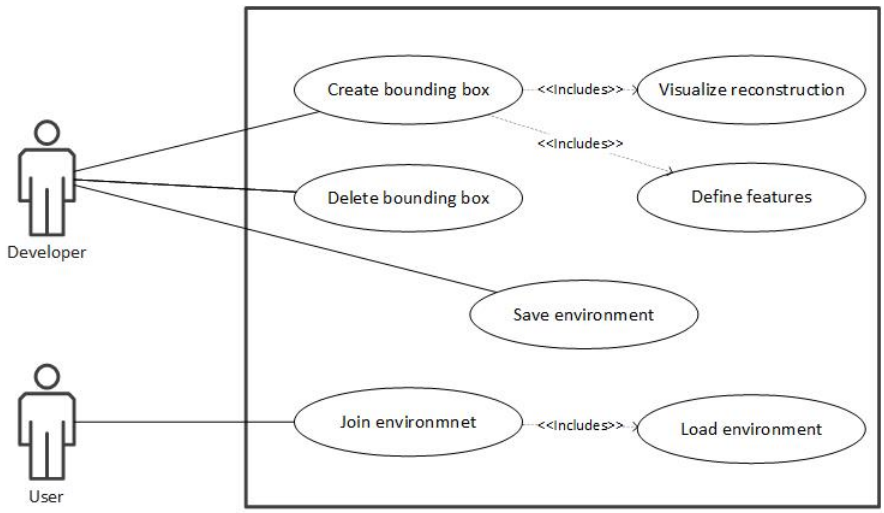


Figura 4.1: Diagramma dei casi d'uso

In fase di progettazione sono stati individuate due tipologie di utilizzatori:

Developer: Utente che utilizza l'applicazione per creare l'ambiente aumentato; modella gli oggetti fisici di interesse e ne decide le proprietà. Creare un bounding box di un oggetto fisico sfruttando la possibilità di visualizzare la mesh generata dal processo di ricostruzione.

User: Utente che, dopo aver selezionato uno tra gli ambienti modellati, interagisce con quell'ambiente.

I ruoli e le interazioni di questi due attori sono descritte nel diagramma dei casi d'uso in figura 4.1;

4.1.2 Modello

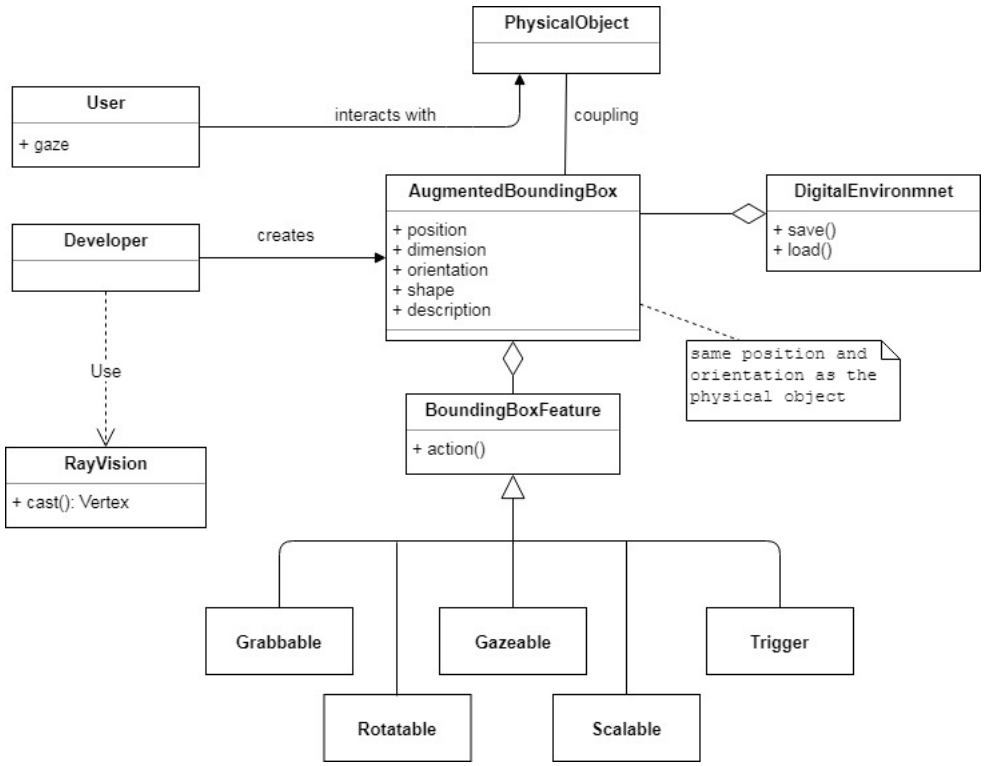


Figura 4.2: Modello concettuale dell'applicazione

L'immagine 4.2 mostra il modello concettuale dell'applicazione. Al centro si trova l'entità *AugmentedBoundingBox* (ABB). Questa rappresenta il bounding associato (*coupling*) all'oggetto fisico (*PhysicalObject*). Le sue caratteristiche

spaziali (*position, dimension, orientation e shape*) coincidono con quelle dell'oggetto fisico a cui si riferisce. L'attributo *description* definisce una stringa che rappresenta la descrizione dell'oggetto fisico accoppiato. Quest'ultima può essere mostrata ad esempio nel momento in cui l'utente interagisce con l'oggetto fisico. Un ABB si compone di *BoundingBoxFeatures*, che ne descrivono le funzionalità. L'insieme degli ABB forma il *DigitalEnvironment*. L'ambiente può essere salvato (ad esempio dallo sviluppatore), e caricato dall'utente. L'entità *Developer* può creare, modificare ed eliminare un ABB ed utilizza il la funzionalità *RayVision*. Una RayVision, attraverso il metodo *cast()* permette di visualizzare punti specifici dell'ambiente circostante. In particolare vertici della mesh derivata dal processo di ricostruzione. Può essere utilizzata nella creazione del bounding box per specificarne le dimensioni di altezza, larghezza e profondità dell'oggetto.

4.2 Sviluppo

L'applicazione è stata sviluppata utilizzando l'headset Meta 2, descritto dettagliatamente nel seguente capitolo. Questo visore mette a disposizione tutte le principali funzionalità richieste per il funzionamento dell'applicazione: ricostruzione dell'ambiente, riconoscimento di gesti e tracciamento dello sguardo; viene messa disposizione inoltre un'ampia libreria per lo sviluppo di applicazione di realtà aumentata da usare in combinazione al software Unity. Seguirà dunque un approfondimento anche su quest'ultima tecnologia, poi una discussione del prototipo dell'applicazione per mostrarne il funzionamento.

4.2.1 Unity

Unity è un motore grafico che permette la creazione rapida di contenuti interattivi multi-piattaforma attraverso l'utilizzo di un ambiente di sviluppo integrato. Viene utilizzato in una vasta tipologia di progetti come videogiochi,animazioni e realtà aumentata, in quanto offre una completa gestione di architetture complesse quali: fisica 2D/3D, grafica , networking.... Qui di seguito verrà spiegata la struttura di un progetto Unity e le sue componenti.

Assests Un *Asset* è la rappresentazione di un qualsiasi oggetto che può essere utilizzato all'interno del progetto Unity. Un Asset può provenire da un file creato al di fuori di Unity, come un modello 3D, un file audio, un'immagine o uno degli altri tipi di file supportati da Unity. Queste risorse vengono raggruppate generalmente in una cartella chiamata "Assests". Attraverso lo

Unity Asset Store è possibile scaricare pacchetti di assets completi in maniera gratuita o a pagamento.

Scenes Una *Scene* di Unity rappresenta un sezione particolare del proprio gioco o progetto, e contiene tutti gli elementi per il suo corretto funzionamento (fig 4.3). In un gioco ad esempio potrebbe essere un livello oppure una schermata di menu. Progetti complessi possono avere anche un vasto numero di scene che solitamente vengono messe in un'apposita cartella all'interno degli Assets chiamata Scenes.

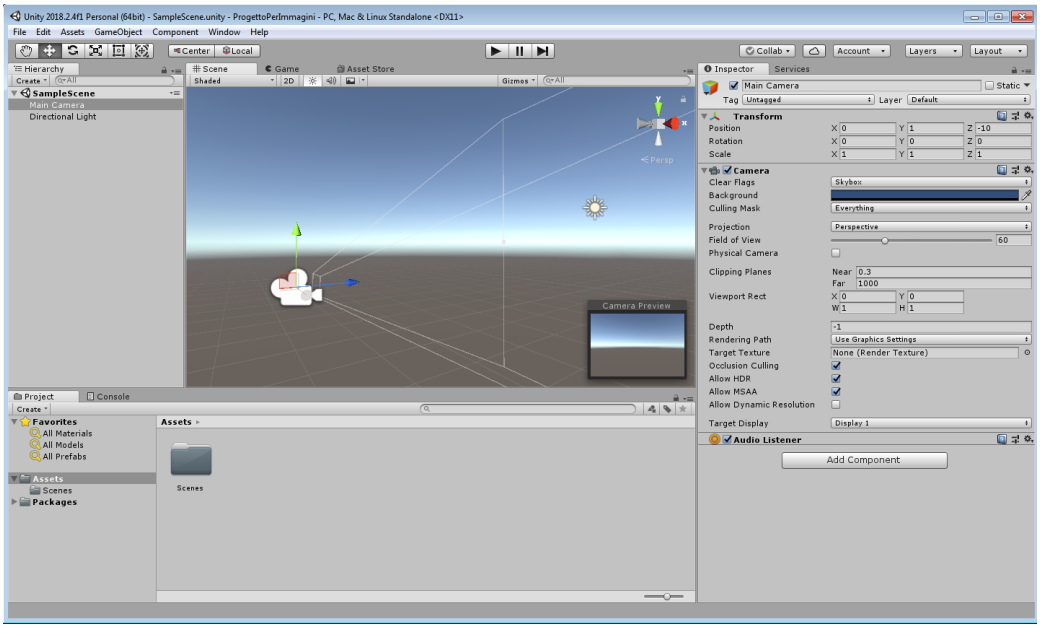


Figura 4.3: Scena base di Unity contenente una camera e un punto luce

GameObject Il *GameObject* è il componente fondamentale di un progetto Unity. Un GameObject può essere qualsiasi oggetto all'interno del progetto, dalla camera per la visualizzazione scena alle luci per illuminarla. Per specializzare un GameObject è necessario dotarlo di componenti che indichino le sue proprietà, e di script che definiscano il comportamento. La proprietà comune a tutti i GameObject è il componente Transform, che specifica posizione, rotazione e scala dell'oggetto e non può essere eliminato.

Components Un *Components* definisce le proprietà e i comportamenti di un GameObject. Ad un GameObject posso essere attaccati più Components così da raggiungere comportamenti sempre più complessi. Alcuni tra i componenti

più importanti sono messi a disposizione da Unity e possono essere utilizzati e inseriti nei propri GameObject (fig 4.4):

Transform Definisce il componente comune a tutti i GameObject contiene le informazioni di posizione, rotazione e scala dell'oggetto.

MeshRenderer Si occupa della visualizzazione della Mesh relativa all'oggetto. Contiene un riferimento ai materiali di cui è composto.

Rigidbody Permette di rendere un GameObject soggetto alle forze fisiche esterne, come ad esempio gravità e urti.

Collider Definisce la regione di spazio attorno ad un oggetto che scatena eventi di collisione.

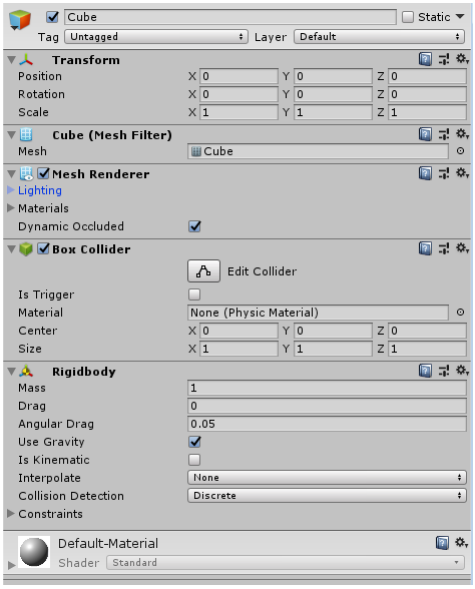


Figura 4.4: Componenti di un cubo

Script Se si vuole creare un comportamento diverso da quelli offerti dai Components di Unity è necessario scrivere uno *Script*. Uno Script è un pezzo di codice che derive dalla classe `MonoBehaviour`. Questa classe specifica il comportamento di un oggetto nei diversi stati del motore di gioco Unity. I due metodi principali sono `Start()` che viene chiamato alla creazione dell'oggetto e `Update()` che viene eseguito ogni frame.

```
using UnityEngine;
public class MonoBehaviour {

    // Use this for initialization
    void Start() {

    }

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

    }
}
```

Listato 4.1: Script vuoto per definire un Monobehaviour

Prefab Nel momento in cui si hanno diversi GameObject che condividono la stessa struttura può essere utili fattorizzare il loro comportamento all'interno di un *Prefab*. Un Prefab è un elemento di Unity che permette di memorizzare la struttura di un oggetto (componenti, script, materiali...) e di istanziarla, quando necessario, in uno specifico oggetto che potrà essere poi particolareggiato a seconda delle esigenze. Una modifica effettuata sul prefab scatenerà le modifiche su tutti gli oggetti che condividono quell'interfaccia così da rendere più semplice la modifica di parti del progetto. L'istanza di un prefab può essere generata a runtime richiamando il metodo `Istanciate()` passando come argomento il riferimento alla prefab.

4.2.2 Dettaglio Meta 2

Questa sezione contiene un approfondimento sulla tecnologia Meta 2, che è quella utilizzata per lo sviluppo del progetto. Vengono mostrati i dettagli hardware e le funzionalità messe a disposizione dal Meta SDK.

Specifiche tecniche

Specifiche riferite al sito ufficiale Meta[1] .

Headset Meta 2

- 90° FOV
- 2.5K resolution, 60Hz refresh rate
- Hand interaction and positional tracking sensors
- 720p front-facing RGB camera
- 9 ft (2.7 m) cable for video, data and power .
- Weighs 1.1 lb / 500g
- Adjustable head straps for comfort
- Memory foam support pads
- 4 surround sound speakers
- 3 microphones

Requisiti hardware

OS: Windows 10 (64-bit)

Processor: Intel Core i7-6700 Processor or AMD AMD FX 9590

Memory: 16 GB RAM DDR4 or more

USB: 2x USB 3.0 ports

3D Engine: Unity 5.6 or higher

Graphics: NVIDIA GeForce GTX 970 or AMD Radeon R9 390X

Video Output: 1x HDMI 1.4b port

Hard Drive Space: 2 GB or more

[1] www.metavision.com

Meta SDK

Il Meta Software Development kit (Meta SDK) è un insieme di funzionalità messe a disposizione da Meta per gli utenti che intendono sviluppare un'applicazione AR con Unity. Oltre a contenere le funzioni basilari del visore è corredato di semplici scene d'esempio che sottolineano le caratteristiche principali di funzionamento ed aiutano l'utente a comprendere le possibilità offerte e come utilizzarle in maniera concreta.

MetaCameraRig MetaCameraRig è un prefab che gestisce l'inizializzazione dell'ambienta e la localizzazione dell'utente. Fornisce le varie videocamere, script e altri GameObject che il visore Meta 2 utilizza per visualizzare e interagire con l'ambiente .Una scena di Unity deve contenere esattamente una singola istanza del MetaCameraRig prefab (fig 4.5).

Figura 4.5: Gerarchia del MetaCameraRig prefab

Meta SLAM localizer All'interno del MetaCameraRig troviamo il componente per il sistema SLAM (Simultaneous Localization And Mapping)[2]. SLAM è un algoritmo che combina i dati delle camere e del sistema IMU per collocare il visore e tenere traccia del suo movimento all'interno dell'ambiente. É possibile scegliere tra la modalità "Limited Tracking" nella quale viene tracciata solamente la rotazione del visore e non la traslazione; gli ologrammi appariranno quindi sempre alla stessa distanza. Questa modalità viene anche definita "rotatin-only" o "3DOF", contrariamente a quella "Full Tracking" che può essere definita a 6DOF.

MetaCompositor MetaCompositor è il componente che si occupa del rendering della scena. Attraverso il compositor è possibile aggiustare i parametri dei sistemi predittivi impiegati per ridurre la latenza di rendering. Permette di attivare o disattivare la Depth Occlusion. Quest'ultima è una funzionalità che si avvale del sensore di profondità e permette agli ologrammi di essere na-

[2]L'algoritmo di SLAM risolve il problema computazionale della costruzione o dell'aggiornamento di una mappa di un ambiente sconosciuto, tenendo contemporaneamente traccia della posizione dell'agente al suo interno.

scosti da oggetti reali che occludono la visuale rendendo così la scena ancora più realistica ed immersiva.

Surface Reconstraction All'interno del MetaCameraRig troviamo anche il prefab "EnvironmentInitialization" che, oltre a dare la possibilità di caricare ambienti ricostruiti in precedenza, permette di attivare la Surface Reconstruction. Quest'ultima, come mostrato in figura 4.6, permette di costruire una mesh 3D dell'ambiente circostante utilizzando i sensori presenti nel visore. L'oggetto risultante dalla ricostruzione può essere utilizzato per integrare in maniera più realistica oggetti virtuali e reali.

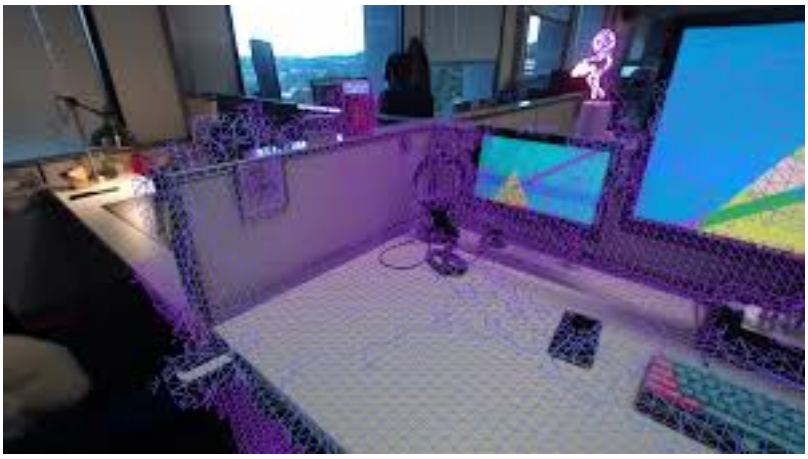


Figura 4.6: Ricostruzione di una ambiente di lavoro tramite mesh 3D

Meta Hands Il prefab MetaHands fornisce informazioni sulle mani dell'utente. Per poter integrare le interazioni delle mani in un progetto è necessario avere un'istanza di questo prefab. Una "Meta Hand" fornisce informazioni sulla mano dell'utente per quanto riguarda la parte centrale (palm), la parte più distante dal polso (top) (solitamente identificata con la punta del dito medio) , e i diversi stati (chiusa, aperta, attiva, non attiva..). Le interazioni e i gesti delle mani riconosciuti dal visore Meta sono diversi, tra i più importanti troviamo:

> ***GrabInteraction***: Se assegnato ad un GameObject questo script permette all'utente di utilizzare le mani per prendere e spostare quell'oggetto nella scena. La chiusura a pungo della mano permette di afferrare l'ologramma, l'apertura di rilasciarlo.
>
> ***TwoHandGrabScale***: Permette di ingrandire o rimpicciolire un ologramma utilizzando due mani.

TwoHandGrabRotation: Permette di ruotare un ologramma utilizzando due mani (fig 4.7).

HandTrigger: Può essere utilizzato per individuare una mano o parti di mano che entrano in nell'area specifica a cui il componente è agganciato.

É possibile combinare o creare nuove interazioni per dare origine a comportamenti più complessi, specifici per la propria applicazione.

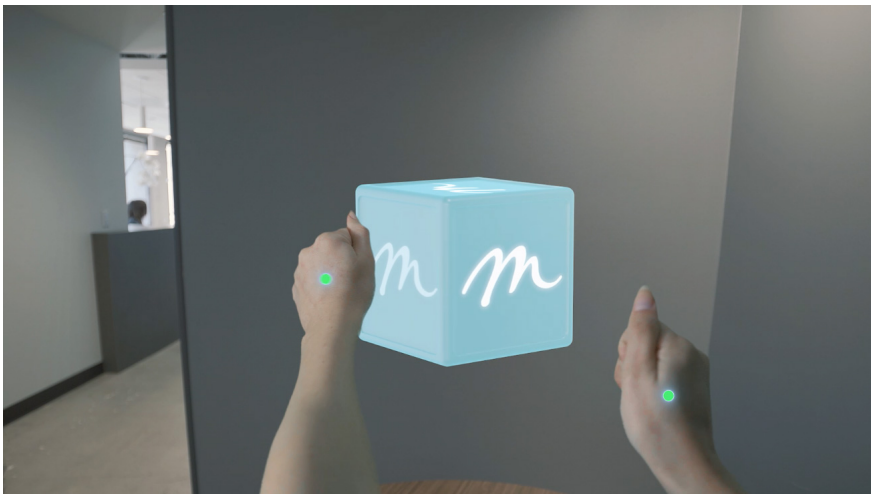


Figura 4.7: Esempio di interazione a duo mani con un ologramma

Meta UI L'SDK di Meta fornisce prefab e script per la creazione di interfacce grafiche nell'ambito AR:

Meta Mouse: Permette di utilizzare il classico mouse da computer come puntatore in un ambiente AR ed interagire con oggetti virtuali. Per utilizzarlo è sufficiente importare nella scena il MetaInputModule prefab e agganciare ai GameObject gli opportuni script: DragTranslate, DragScale e DragRotate.

Meta Canvas: É un prefab che funziona da tool per creare interfacce grafiche 2D che possano interagire con le Meta Hands e il Meta Mouse. Si compone di elementi UI di Unity e di oggetti personalizzati di Meta.

.

Meta Gaze Mete Gaze è un tipo di interazione che utilizza lo sguardo dell'utente per evidenziare oggetti virtuali nella scena. Funziona eseguendo un Raycast partendo dalla camera principale del visore e lanciando l'evento di

gaze appena intercettato un oggetto avente un collider. Per rispondere ad un evento di Gaze un oggetto deve contenere uno script Unity che implementi le interfacce `IGazeStartEvent` e la `IGazeEndEvent`, ovvero che definisca i due metodi `OnGazeStart()` e `OnGazeEnd()`.

4.2.3 Prototipo

Il prototipo dell'applicazione è stato scritto in C# attraverso l'utilizzo dell'IDE di Unity.

Augmented Bounding Box

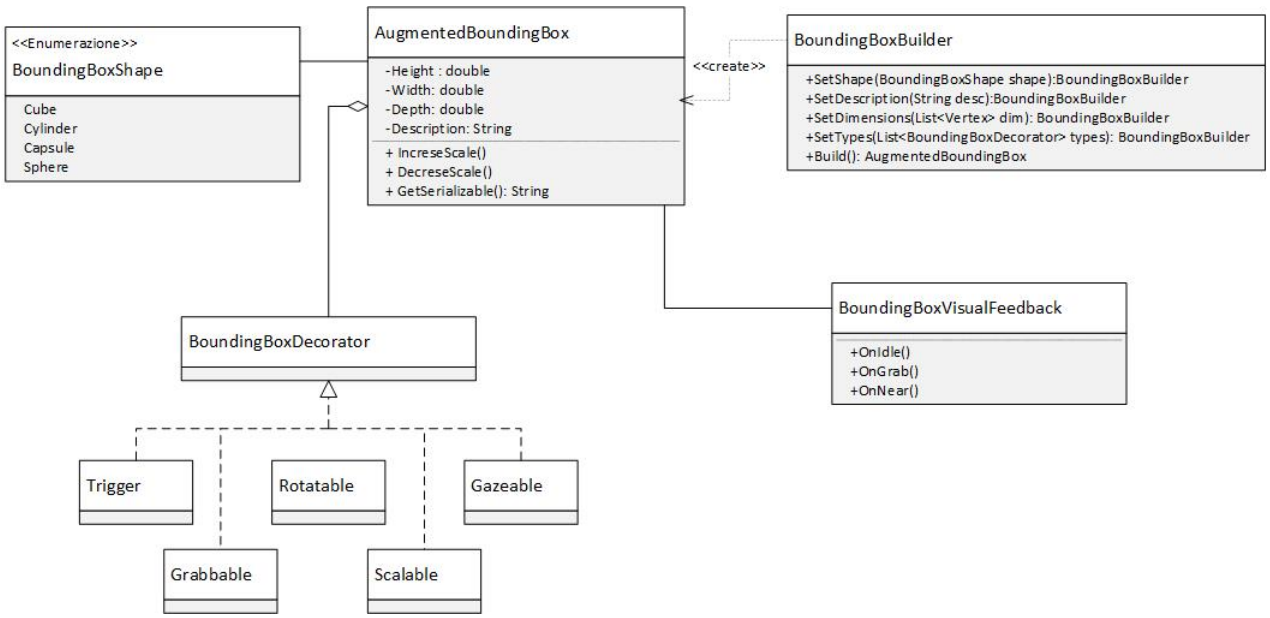


Figura 4.8: Dettaglio su AugmentedBoundingBox

La classe AugmentedBoundingBox (fig. 4.8) incapsula il comportamento del bounding box relativo ad un oggetto fisico. Viene rappresentato nella scena di Unity come un GameObject ed istanziato attraverso un prefab dal *BoundingBoxBuilder* alla creazione. Quest'ultima classe si occupa infatti di generare nella scena il GameObject a partire dalle caratteristiche, dalle dimensioni specificate e dalle tipologie specificate. Il prefab di un augmented bounding box si compone di un collider, di un canvas e dello script *BoundingBoxVisualFeedback*. Il collider può assumere un sottoinsieme delle forme base messe a disposizione da Unity, mostrate in figura 4.9.

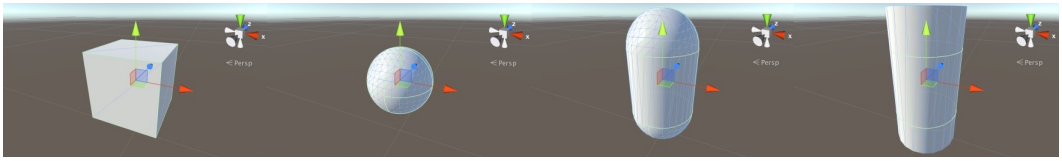


Figura 4.9: Forme di base per la creazione di bounding box

Il canvas viene utilizzato per mostrare la descrizione dell'oggetto nella scena. Lo script definisce invece in che maniera l'utente vede il bounding box nella scena (fig 4.10). Gli eventi principali modellati nell'applicazione sono:

- `OnIdle`: Richiamato quando il bounding box non partecipa a nessuna interazione
- `OnGrab`: Richiamato quando l'utente sta manipolando l'oggetto
- `OnNear`: Richiamato quando le mani dell'utente sono vicine al bounding box.

Figura 4.10: Due oggetti fisici accoppiati a bounding box digitali. In questo caso un cubo e un cilindro. Viene mostrato all'utente un bounding box con un materiale wireframe che enfatizza il contorno dell'oggetto.

Nella creazione del bounding box è possibile definire le interazioni desiderate attraverso la concatenazione di BoundingBoxDecorator:

- **Grabbable**: Aggiunge al GameObject il componente di Meta per il grab *GrabInteraction* per permette all'utente di afferrare e spostare il bounding box.
- **Trigger**: Aggiunge al GameObject il componente HoverInteraction, che genera eventi se la mano dell'utente tocca il bounding box.
- **Rotatable**: Aggiunge al GameObject il componente di Meta per la rotazione *TwoHandGrabInteraction* per permette all'utente di afferrare e spostare il bounding box.
- **Scalable**: Aggiunge al GameObject il componente di Meta per il grab *TwoHandScaleInteraction* per permettere all'utente scalare il bounding box.
- **Gazable**: Aggiunge al GameObject la possibilità di gestire eventi di gaze.

La combinazione di più funzionalità permette di creare entità che permettono interazioni diverse. Se ad esempio vien creato un bounding box Grabbable e Scalable, l'oggetto oltre ad essere afferrabile e spostabile, sarà anche scalabile dall'utente.

Ray Vision

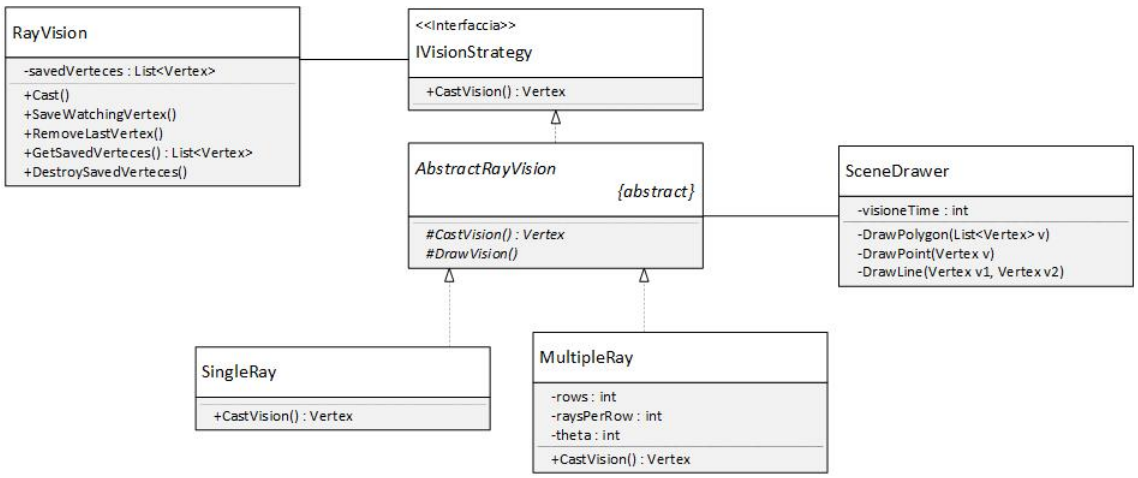


Figura 4.11: Dettaglio su RayVision

La creazione di un AugmentedBoundingBox richiede la conoscenza della forma e delle dimensioni dell'oggetto fisco che si vuole modellare. La classe *RayVision* (fig. 4.11) fornisce i metodi per visualizzare la mesh e salvare i vertici di interesse. Utilizza un'implementazione dell'interfaccia *IVisionStrategy* per trovare il vertice correntemente osservato dal visore. Sono state implementate due strategie nelle classi *SingleRay* e *MultipleRay*. Il listato 4.2 mostra come il SingleRay sfrutti il Raycasting di Unity per generare un raggio che partendo dalla posizione dell'utente intercetti un triangolo della ricostruzione. MultipleRay sfrutta invece più raggi simultaneamente, permettendo una visione più ampia della mesh. Le due strategie estendono la stessa classe astratta *AbstractRayVision* che contiene le funzionalità comuni tra le due. In particolare mette a disposizione un'istanza della classe *SceneDrawer* che sfruttando il componente LineRenderer di Unity, permette di disegnare forme basilari all'interno della scena.

```
 using  UnityEngine;

//Single ray vision strategy.
class SingleRay : AbstractVisionStrategy
{
    public SingleRay(SceneDrawer drawer) : base(drawer) {}

    //Perform a single raycast from the given transfom
    public override Vector3 CastVision(Transform t)
    {
        int n = ReconstructionMeshConstants.RECONS_MESH_LAYER_IDX;
        //Check collisions with world mesh layer
        var layerMask = 1 << n;
        RaycastHit hit;
        var ray = new Ray(t.position, t.forward);
        if (!Physics.Raycast(ray, out hit, Mathf.Infinity, layerMask))
        {
            return Vector3.zero;
        }
        else
        {
            MeshCollider meshCollider = hit.collider as MeshCollider;
            if (ExistsCollisionMesh(meshCollider))
            {
                //Draw hit triangle on scene
                DrawHitTriangle(meshCollider.sharedMesh, hit,
                    GetColor());
                //return the first vertex of the hit triangle
                return GetHitTriangleVerteces(hit)[0];
            }
            else
            {
                return Vector3.zero;
            }
        }
    }
}
```

Listato 4.2: SingleRay strategy

Nella figura 4.12 si può notare come attraverso la visualizzazione della mesh sia possibile individuare le dimensioni di un oggetto fisico.

Figura 4.12: Modellazione di un oggetto fisico attraverso l'utilizzo della ricostruzione di Meta. I punti specificati definisco le dimensioni dell'oggetto

Salvataggio e Caricamento di un Environment

L'applicazione offre allo sviluppatore la possibilità di salvare un ambiente appena modellato. Tramite l'interfaccia è possibile specificare il nome dell'ambiente e premere il pulsante "Save Environment". In questo modo l'utente finale che utilizza l'applicazione troverà l'ambiente nella lista degli ambienti selezionabili Il procedimento descritto sopra viene effettuato dalla classe *EnvironmentManager* (fig 4.13). Questa mette a disposizione i metodi per salvare un AugmentedBoundingBox e caricarlo tramite file. Vengono utilizzate le funzionalità messe a disposizione da Unity per lavorare con il formato Json:

- `JsonUtility.ToJson` - Permette di trasformare una classe marcata come "Serializable" in una stringa formato Json.
- `JsonUtility.FromJson` - Permette di trasformare una stringa in formato Json in un oggetto.

Per ottenere la rappresentazione json di un oggetto AugmentedBoundingBox viene utilizzata la classe *BoundingBoxData* contenente tutte le informazioni serializzabili di un bounding box. Per salvare un intero Environment per

prima cosa viene creata una cartella all'interno della directory principale "Environments" specificando il nome dell'ambiente. All'interno della cartella ogni bounding box viene rappresentato separatamente come un file di testo contente la sua rappresentazione json. Durante il caricamento di un ambiente vengono poi deserializzati tutti i file contenuti in una specifica cartella e successivamente ricreati tutti i bounding box a partire dalla informazioni lette.

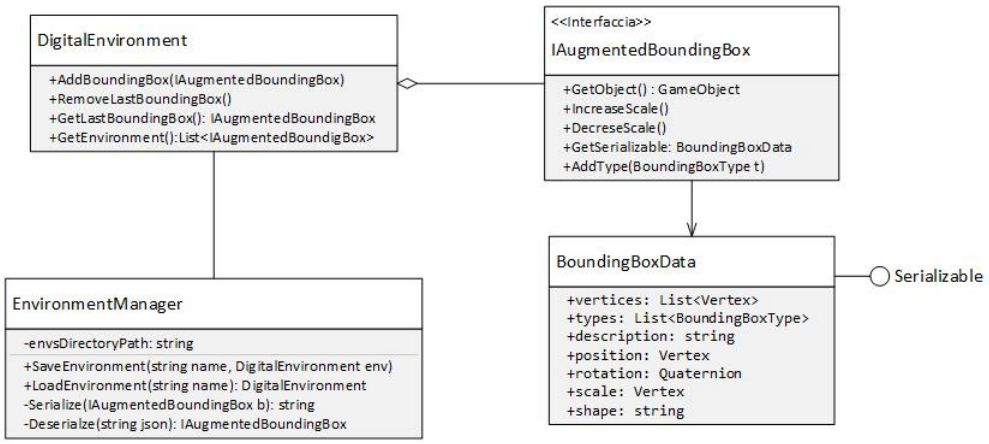
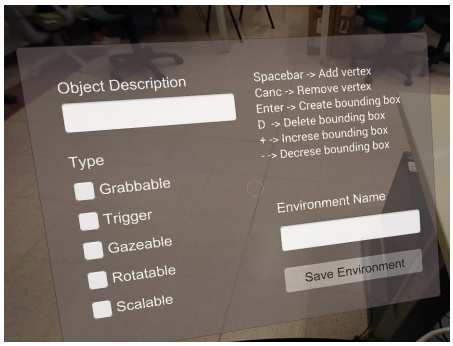


Figura 4.13: Dettaglio su EnvironmentManager

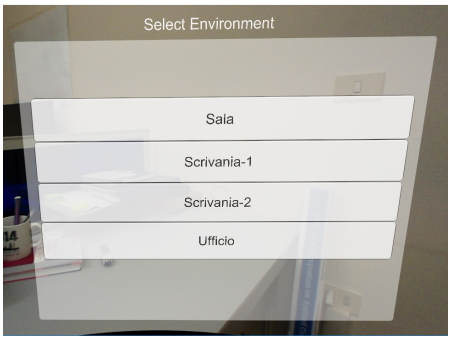
Interfaccia grafica

Per definire le proprietà del bounding box di un oggetto fisico è stato aggiunto nella scena, alla sinistra dello sviluppatore, un pannello (fig 4.14a)che permette di specificare nome (attraverso una textbox) e combinazione di tipologie (attraverso delle checkbox). Sempre dallo stesso panello è possibile, dopo aver selezionato un nome, utilizzare il bottone "Save Environment" per salvare l'ambiente modellato.

L'utente invece, all'avvio dell'applicazione, può visualizzare un pannello (fig 4.14b) che permette di scorrere i diversi ambienti salvati e caricare quello di interesse premendo il relativo bottone.



(a) Interfaccia sviluppatore



(b) Interfaccia utente

Per creare interfacce con caratteristiche spaziali all'interno della scena è stata utilizzata la funzionalità MetaCanvas dell'SDK Meta. Questa permette di creare interfacce grafiche interattive che permettono l'utilizzo combinato di mouse, tastiera e gesti delle mani.

Capitolo 5

Validazione

In questo capitolo verrà presentata la parte di validazione. Nella prima sezione si discuteranno i risultati ottenuti a seguito del lavoro svolto mostrando le funzionalità e le caratteristiche principali dell'approccio presentato. Nella seconda sezione si mostreranno i limiti e le problematiche emerse durante i test, discutendo qualche possibile soluzione.

5.1 Risultati ottenuti

L'obiettivo prefissato per questa tesi era quello di creare uno strumento per la modellazione di oggetti fisici e dotarli di una rappresentazione digitale nel modello Augmented World. A tal proposito è stata sviluppata un'applicazione utilizzando Unity e Meta 2, che permette di sfruttare la funzionalità della ricostruzione dell'ambiente offerta dal visore per identificare bounding box di oggetti fisici partendo proprio dai vertici della mesh individuata. Una volta identificati i bounding box e definite le loro proprietà è possibile sfruttare le informazioni digitali sull'oggetto fisico per definire nuovo interazioni con entità digitali e con utenti. Posso ritenermi soddisfatto del lavoro svolto in quanto l'applicazione funziona correttamente ed implementa tutte le funzionalità emerse in fase di analisi. La possibilità di far interagire oggetti del mondo reale con entità digitali non è banale e i risultati ottenuti sono facilmente sfruttabili per questo scopo. L'utilizzo una tale applicazione è utile in svariati campi: sia per sviluppare un ambiente più usabile ed interattivo per un utilizzatore umano, sia per programmare agenti che si muovano in maniera intelligente all'interno dell'ambiente. Per testare l'applicazione sono stati modellati diversi ambienti con oggetti e forme diverse provando a mischiare l'interazione con oggetti digitali ed oggetti fisici. Si è potuto constatare, come previsto, che i bounding box digitali seguono gli oggetti fisici che l'utente sposta (fig. 5.1),

mantenendo così l'accoppiamento spaziale e premettendo una connessione più profonda tra reale e digitale.

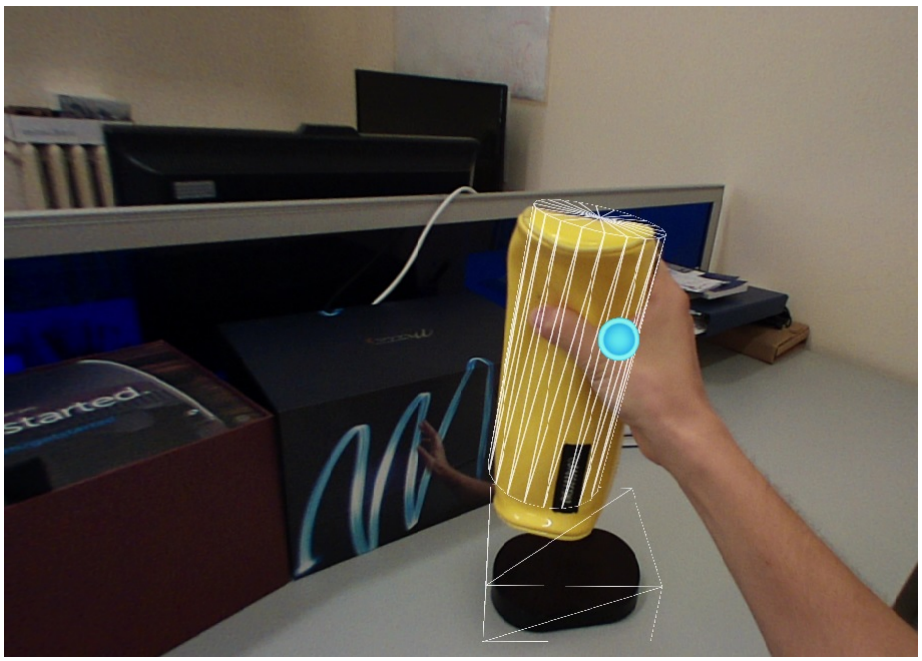


Figura 5.1: Entità digitale in movimento con l'oggetto fisico. L'azione di afferrare l'oggetto fisico viene riconosciuta a livello digitale così da permettere lo spostamento simultaneo.

Effettuando dei test empirici si è riscontrato inoltre che maggiore è la precisione del bounding box rispetto alla forma dell'oggetto fisico migliore è l'usabilità finale dell'oggetto. Le forme disponibili per la modellazione implementate nel prototipo sono cubo, sfera, cilindro e capsula. Non è difficile comunque pensare di estendere la lista aggiungendo mesh più complesse create ad hoc, utilizzando anche strumenti di grafica

Anche se le possibilità offerte da questo approccio sono molteplici bisogna tenere presente che l'applicazione non è esente da difetti. La risoluzione di questi problemi non è banale e apre diverse possibilità per raggiungere una soluzione efficace.

5.2 Problemi rilevati

In questa sezione verranno descritti i principali problemi rilevati durante l'utilizzo dell'applicazione.

5.2.1 Problematiche relative al visore

Una prima difficoltà che si è riscontrata già nelle le prime fasi di sviluppo è quella relativa al funzionamento della tecnologia Meta 2. I problemi principali sono i seguenti:

Ricostruzione dell'ambiente

La ricostruzione dell'ambiente effettuata dal visore non è perfetta; anche dove la superficie fisica è chiara e ben definita la mesh risultante è frammentaria e poco precisa. Una volta creata, inoltre, nel momento in cui l'utente si muove e cambia prospettiva, la mesh si discosta dalla superficie reale perdendo così il legame con il mondo fisico. Queste imprecisione rendono più difficile la modellazione di un oggetto fisico e portano alla creazioni di bounding box con una posizione ed una dimensione non coincidenti con quelle reali.

Riconoscimento dei gesti

Una funzionalità chiave per l'utilizzo di un bounding box aumentato è quella del riconoscimento del evento di grab (ovvero quando la mano sta afferrando un oggetto digitale) e quello di release (quando la mano lascia l'oggetto). Il funzionamento ideale vedrebbe la lettura dell'evento di grab sul bounding box digitale nel esatto momento in cui l'utente afferra l'oggetto fisico e la lettura dell'evento di release quando l'oggetto viene lasciato. Provando il visore Meta 2 risulta evidente però che il riconoscimento dei gesti sia efficace e soddisfacente principalmente in condizioni ottimali; spesso risulta poco preciso e comporta uno sfasamento spaziale tra entità fisica ed entità digitale. Quello che succede è che spostando l'oggetto fisico non viene letto l'evento di grab sull'oggetto digitale rompendo così il legame spaziale. Azioni che evidenziano particolarmente questi difetti sono lo spostamento troppo rapido di oggetti e movimenti poco precisi nell'afferrare gli oggetti. La risoluzione di questi problemi, per il momento, spetta all'utente: utilizzando gesti semplici e riconoscibili, muovendo lentamente le mani e se necessario, ripristinare manualmente l'accoppiamento spaziale.

Queste problematiche sono indipendenti dall'applicazione e difficilmente controllabili. In ogni caso è molto probabile che nuove tecnologie risolvano par-

zialmente o totalmente questi problemi e che in futuro si possa fare affidamento su algoritmi più efficaci e più precisi.

5.2.2 Accoppiamento spaziale

Il problema dell'accoppiamento spaziale tra entità fisica ed entità digitale, non si limita solamente al caso descritto sopra. Gesti che vengono effettuati al di fuori del campo visivo dell'headset infatti non posso essere riconosciuti. Se ad esempio un oggetto fisico viene spostato senza essere visto, l'entità digitale non potrà seguirlo e rimarrà nella stessa posizione in cui era stata lasciata. Al momento, per mantenere la correttezza e la coerenza dell'accoppiamento, è necessario assumere che oggetti fisici accoppiati ad entità digitali non possano essere spostati al di fuori del FOV del visore. L'utilizzo di un algoritmo di visione artificiale potrebbe risolvere questo problema riconoscendo che l'oggetto fisico ha cambiato posizione e riposizionando il bounding box digitale di conseguenza. Un approccio di questo tipo permetterebbe di risolvere anche lo stesso problema causato dall'errato riconoscimento dei gesti di grab e release.

5.2.3 Problema della registrazione

Il problema della registrazione gioca un ruolo fondamentale in un'applicazione di realtà aumentata. In questo caso nel caricamento di un ambiente digitale. La posizione dei bounding box aumentati è gestita in riferimento alla posizione dell'utente che si assume essere (0, 0, 0). La posizione dell'headset viene calcolata all'avvio dell'applicazione, dunque non è certo che il centro del mondo coincida tra due sessioni diverse di utilizzo. Utilizzando l'applicazione si è visto che nel caso in cui il processo di inizializzazione non avvenga precisamente nella stessa posizione si assiste ad uno sfasamento di tutto il mondo digitale rispetto al mondo fisico. Il problema è risolvibile solamente attraverso un processo efficace di registrazione. Ogni ambiente aumentato potrebbe avere ad esempio un oggetto fisico di riferimento sul quale sincronizzarsi se si vuole iniziare una sessione al suo interno, terminata l'inizializzazione l'oggetto fisico rappresenterebbe il centro del mondo trasformando le coordinate rispetto al suo riferimento.

Conclusioni

L'obiettivo della tesi era quello di sviluppare uno strumento che permettesse l'integrazione di oggetti fisici come entità aumentate all'interno di un Augmented World (AW). Sfruttando le funzionalità messe a disposizione dalla tecnologia Meta 2 e utilizzando il software Unity, è stato sviluppato uno strumento che permette di generare un bounding box digitale dell'oggetto fisico a partire dalla ricostruzione dell'ambiente effettuata dal visore e di interagire con l'oggetto fisico trattandolo come un ologramma. L'approccio descritto nella tesi non si limata necessariamente al modello AW preso come riferimento, ma può essere utilizzato in maniera analoga per gestire anche diversi ambienti aumentati, utilizzando visori per la Mixed Reality diversi da Meta 2. Anche considerando i limiti dell'approccio presentato, il risultato finale è notevole e le applicazioni di utilizzo molteplici; in particolare credo che possa essere considerato una novità nell'ambito in cui si inserisce.

5.2.4 Questioni aperte

Per concludere propongo alcune questioni interessanti che non sono state approfondite nello sviluppo del prototipo:

- **Modellazione attraverso forme ad hoc** Il prototipo dell'applicazione permette solamente l'utilizzo di forme base (cubi, sfera, cilindro e capsula), si potrebbe invece permettere la creazione di forme più complesse che modellino meglio l'oggetto fisico e reagiscano meglio alle interazione, addirittura pensando di specificare l'insieme di tutti i punti della mesh appartenenti ad uno specifico oggetto fisico.

- **Automatizzazione del processo di modellazione** Sarebbe utile avere un processo che a partire da una regione della mesh costruisca il bounding box associato in maniera automatica, senza bisogno di specificare manualmente le sue dimensioni.

- **Utilizzo di algoritmi di visione artificiale per il riconoscimento di oggetti** Come detto in precedenza l'utilizzo di algoritmi di visione

artificiale permetterebbe un notevole miglioramento dal punto di vista funzionale ma anche della user experience.

- **Procedure efficaci per il processo di registrazione** É fondamentale nell'ambito di questa applicazione la possibilità di avere un processo efficace di registrazione che permetta una sincronizzazione coerente tra più sessioni. Questo risolverebbe il problema dell'offset che si crea tra il mondo digitale e quello fisico a seguito di un errata inizializzazione.

Ringraziamenti

In primo luogo vorrei ringraziare il professore Alessandro Ricci e il dottore Angelo Croatti per avermi dedicato il loro tempo nello sviluppo di questo progetto e avermi dato l'opportunità di lavorare al loro fianco durante questi ultimi mesi.

Voglio ringraziare anche tutti i miei colleghi che mi hanno accompagnato in questo percorso universitario: è anche grazie al loro importante supporto che tutto questo è stato possibile.

Ringrazio i miei amici perché hanno condiviso con me i momenti più importanti della mia vita rendendoli speciali ogni volta.

Il più grande ringraziamento lo dedico infine alla mia famiglia che mi ha sempre sostenuto ed incoraggiato con grande gioia e amore nello studio come nella vita, permettendomi di raggiungere questo stupendo traguardo.

Bibliografia

[1] Ronald T. Azuma, *A survey of augmented reality*, 1997.

[2] Luca Tummolini, Cristiano Castelfranchi, Alessandro Ricci, Michele Piunti *In The Mirror World: Preparing for Mixed-Reality Living*, , 2015.

[3] Alessandro Ricci, Angelo Croatti, *In A Model and Platform for Building Agent-Based Pervasive Mixed Reality Systems*, DISI, University of Bologna, Via Sacchi 3, Cesena, Italy.

[4] Jon Peddie *In Augmented Reality: Where We Will All Live*, Springer International Publishing, 2017.

[5] F. Kishino, P. Milgram *A taxonomy of mixed reality visual di- splays*, In IEICE Trans. Information Systems, vol. E77-D, ATR Communication Systems Research Laboratories, Japan, 1994.