ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

STUDIO E SVILUPPO PROTOTIPALE DI UN'APPLICAZIONE DI REALTÀ MISTA IN AMBITO HEALTHCARE CON DISPOSITIVO HOLOLENS 2

Elaborato in
SISTEMI EMBEDDED E INTERNET OF THINGS

Relatore
Prof. ALESSANDRO RICCI

Correlatore
Dott. Ing. ANGELO CROATTI

Presentata da
GIULIA NARDICCHIA

Anno Accademico 2020 – 2021

Alla mia famiglia.
Ai miei nonni.
Spero di avervi reso fieri di me.

Indice

Introduzione

Le persone hanno passato tanto tempo ad imparare a comunicare tramite i computer, è diventato sempre più naturale generazione dopo generazione. E' un risultato ottenuto grazie all'avvento delle nuove tecnologie e l'evoluzione dei dispositivi, come ad esempio interfacce utente grafiche, mouse, il touch screen, la voce e i gesti. I sistemi di realtà mista hanno portato l'uomo a sviluppare, sperimentare ed inventare nuovi modi per comunicare differenti rispetto ad uno schermo 2D, ovvero tramite display indossabili [1]. Poiché si tratta di una novità, non si comprendono appieno le opportunità e le potenzialità che può offrire. Però oggigiorno il ruolo della realtà mista è diventato sempre più importante e si è sviluppato nei più svariati settori.

La seguente tesi si pone come obiettivo lo studio e lo sviluppo di applicazioni di realtà mista, descrivendo quali siano le tecnologie di sviluppo più promettenti nell'ambito healthcare.

Nel primo capitolo vengono date le basi teoriche per comprendere la cosiddetta Extended Reality presentando le tecnologie che la compongono come la Virtual Reality, l'Augmented Reality e la Mixed Reality grazie alle definizioni di Azuma, Milgram e altri personaggi di spicco nella storia di questa tecnologia emergente. Inoltre, viene specificata la differenza delle definizioni di AR e MR. Nello stesso capitolo vengono descritte le diverse tipologie di dispositivi in grado di permettere la realtà mista, con le caratteristiche che le contraddistinguono. In particolare vengono presentati due dei dispositivi più ampiamente utilizzati quali: HoloLens 2 e MagicLeapOne. Dopo vengono mostrati tutti gli aspetti da considerare per realizzare un'applicazione di realtà mista, a partire dalle caratteristiche degli ologrammi, come sarebbe meglio posizionarli nello spazio, quali sono i modi per poter interagire con essi, come vengono riconosciute le superfici fino alle esperienze condivise tra più utenti. Successivamente vengono mostrate le applicazioni mediche già presenti in letteratura, create da software house importanti che si pongono gli stessi obiettivi di questo elaborato di tesi avente come scopo primario la diagnosi e il sostegno all'assistenza sanitaria.

Nel secondo capitolo vengono descritti in generale lo scopo dell'applicazione, quali sono gli obiettivi principali del progetto e vengono mostrati gli scenari

d'uso che l'utente può scegliere di effettuare durante l'esecuzione. Successivamente viene introdotto il significato di alcuni termini importanti, si descriverà il concetto di DICOM e modello 3D e perché sono importanti.

Nel terzo capitolo viene data una panoramica delle tecnologie utilizzate nel progetto che servono a sviluppare applicazioni di realtà mista, quali: Unity, MRTK e OpenXR.

Nel quarto capitolo si scende più nel dettaglio della progettazione e implementazione del prototipo. Vengono descritti ad alto livello le funzionalità che svolgono i sottosistemi che compongono l'applicazione e poi viene spiegato più nello specifico come sono stati implementati.

Il quinto capitolo tratta della validazione, presentando un esempio pratico con tutti i passaggi. Viene mostrato cosa avviene all'avvio dell'applicazione, come si può interagire con il menù e come invece con l'ologramma rappresentante il file DICOM e il modello 3D. Poi si evidenziano quali sono le problematiche riscontrate durante la programmazione e vengono, inoltre, presentate delle idee per effettuare dei miglioramenti o nuove funzionalità per sviluppi futuri.

Capitolo 1

Applicazione della Mixed Reality in ambito healthcare

In questo primo capitolo si tratterà in generale degli aspetti fondamentali che servono ad introdurre il mondo della realtà mista applicata nell'ambito della sanità digitale. Però prima di addentrarsi nel merito della trattazione, è bene chiarire alcuni concetti di base che aiuteranno a capire il contesto dentro al quale ci si trova. Si spiegherà il significato di extended reality, virtual reality, augmented reality e mixed reality. Questi termini pian piano sono sempre più entrati a far parte del comune parlato.

1.1 Definizione di Extended Reality

L'Extended Reality, significa letteralmente realtà estesa, vale a dire un'estensione del mondo reale ottenuta mediante tecnologie innovative che permettono di percepire l'ambiente circostante in modo potenziato, combinandola con elementi virtuali. L'Extended Reality è solitamente indicato con il termine XR ed è un termine generico che indica il raggruppamento di tecnologie come la Virtual Reality (VR), l'Augmented Reality (AR) e la Mixed Reality (MR).

1.2 Definizione di Virtual Reality, Augmented Reality e Mixed Reality

Secondo la teoria di Milgram e Kishino, nel famoso "Reality-Virtuality Continuum" [2], il concetto di ambiente reale e ambiente virtuale non sono nettamente separati l'uno con l'altro come ci si potrebbe aspettare ma esistono sfumature di realtà intermedie che collegano i due estremi. Ovvero mostra

come ci sia uno spettro di tecnologie che vanno da un ambiente completamente reale ad un ambiente completamente virtuale.

Adesso si esaminerà la rappresentazione seguente.

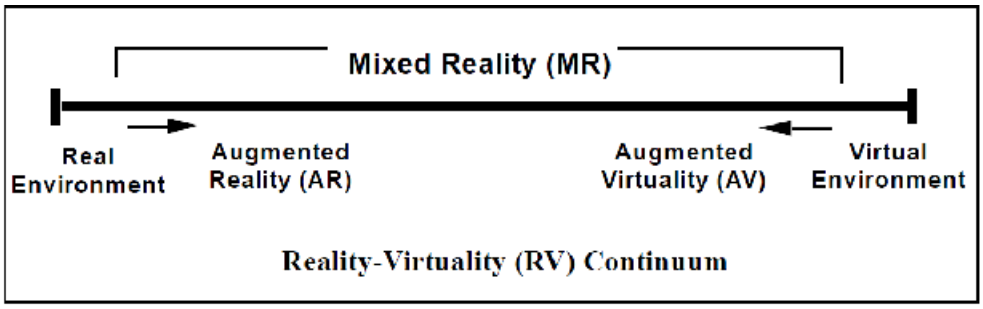


Figura 1.1: Reality-Virtuality Continuum [3]

All'estrema sinistra vi troviamo l'ambiente reale, si tratta dell'esperienza che ciascun uomo sulla terra vive ogni giorno della propria esistenza.

All'estrema destra vi è invece l'ambiente virtuale. La definizione più comune che si può presentare di un ambiente a realtà virtuale è quella in cui l'utente è totalmente immerso in un mondo sintetico, che può imitare o meno le proprietà di un ambiente del mondo reale, esistente o immaginario. L'effetto dell'immersione è amplificata se non si può vedere o sentire lo spazio fisico. Il mondo virtuale può addirittura superare i limiti del fisico creando un mondo in cui si va oltre alle leggi fisiche che governano la gravità, il tempo e le proprietà dei materiali.

Spostandosi da sinistra verso destra nell'immagine, troviamo l'Augmented Reality. Ronald Azuma, considerato da tanti un esperto in AR, nel suo articolo più famoso "A survey of Augmented Reality" [4] pubblicato nel 1997, definisce la realtà aumentata come un'esperienza interattiva di un ambiente reale in cui alcuni elementi sono virtuali. Queste informazioni possono essere additive, che aggiungono qualcosa all'ambiente naturale o distruttive, che mascherano l'ambiente reale. Pertanto l'AR integra la realtà anziché sostituirla completamente, gli oggetti reali e virtuali coesistono nello stesso spazio. (Un esempio concreto di come questi due aspetti possano coesistere è rappresentato dall'effetto raggiunto dal film "Who Framed Roger Rabbit?"). Al contrario della realtà virtuale, in questo caso chiaramente deve essere strettamente vincolato dalle leggi fisiche.

Spostandosi, invece, da destra verso sinistra della realtà virtuale è presente la Virtualità Aumentata. Si tratta di uno spazio prevalentemente virtuale in cui sono integrati degli elementi reali.

In base al pensiero di Milgram e Kishino, tutto ciò che si trova in mezzo alle due estremità rientra nella definizione di Mixed Reality.

La Mixed Reality (MR) combina elementi di AR e VR creando un'esperienza più complessa, generando una nuova realtà composta da elementi reali e virtuali simultaneamente. Per realizzare la realtà mista si ha bisogno di uno specifico device perché si deve avere una consapevolezza del mondo reale.

"Mixed Reality" comparve per la prima volta in un documento del 1994 dei Paul Milgram e Fumio Kishino, "A Taxonomy of mixed reality Visual Displays". Esiste molta confusione riguardo a questo termine, infatti vi è più di una definizione e anche gli esperti dibattono su questo argomento.

1.3 Differenza tra Augmented Reality e Mixed Reality

Molte volte i termini quali Augmented Reality e Mixed Reality vengono usati erroneamente come sinonimi, come parole interscambiabili. Esiste una profonda differenza tra i due, ovvero: l'occlusione. In altre parole ciò che manca all'AR rispetto alla MR è la consapevolezza del mondo reale, anche chiamata *spatial awareness*. Se ad esempio, si ha un ologramma, con l'Augmented Reality esso è libero di muoversi all'interno della stanza ed è in grado di riconoscere le pareti o il pavimento, ma non i mobili, le sedie o altri oggetti reali e quindi passerà attraverso ciascuno di essi e rimarrà sempre visibile. Se si usa, invece, la Mixed Reality, l'ologramma è in grado di interagire con tali elementi reali e può essere usata per creare nell'utente aspettative o l'illusione di una interfaccia naturale basata su interazioni fisiche familiari. Se un oggetto virtuale è occluso da una superficie è perché è solida e l'utente si aspetterà che collida con la superficie e non che l'attraversi. Ad esempio, l'ologramma può sbattere contro la sedia, si può appoggiare sul tavolo e se lo si spinge può cadere per terra. Inoltre se l'ologramma viene posto dietro ad un mobile, risulterà nascosto e non sarà più visibile, per poter individuare nuovamente l'oggetto virtuale l'utente dovrà semplicemente ruotare la testa o spostarsi. Per ottenere l'occlusione è necessario utilizzare specifici device che abbiano gli opportuni sensori per effettuare una mappatura spaziale e riuscire in questo modo a percepire gli elementi circostanti nell'ambiente [5].

1.4 Tassonomia dei dispositivi MR

In questo paragrafo si vuole elencare le varie tipologie di *displays* che vengono utilizzate solitamente in un ambiente MR. Si possono distinguere i dispositivi *Head-mounted display*, *Head-up display* e *Hand-held display* [6].

1.4.1 Head-mounted display

I dispositivi *Head-mounted display* (HMD), tradotto letteralmente in italiano come schermo montato sulla testa, sono la tipologia di dispositivo più comunemente utilizzata in MR. Originariamente sono state sviluppate esclusivamente per VR e quindi permettevano all'utente di percepire solo quello che era possibile vedere attraverso il display del visore, posizionato davanti agli occhi dell'utente e non fornivano alcuna capacità di visualizzazione attraverso. Tuttavia in seguito sono state montate delle videocamere con la quale è possibile integrare il mondo reale mescolando le immagini virtuali alla realtà per creare la MR. Si distinguono diversi tipi di HMD in base alla tecnologia usata per far visualizzare la realtà. Nel caso in cui si utilizza una telecamera per vedere attraverso, si tratta di tecnologia di tipo video see-through HMD ma esiste anche l'optic see-through HMD che utilizza invece degli specchi semi-trasparenti per ottenere una combinazione ottica. Inoltre, gli HMD si differenziano anche in base al campo visivo dell'utente: monoculari (a occhio singolo) e binoculari (a due occhi).
Adesso verranno illustrati due dei dispositivi più ampiamente utilizzati, quali *HoloLens 2* e *Magic Leap One* e verranno descritte in breve le differenze principali.

HoloLens 2

Figura 1.2: Dispositivo HoloLens 2 [7]

Microsoft HoloLens 2 è un visore Head-mounted displays per la realtà mista sviluppata da Microsoft. E' stato il primo dispositivo di tipo video see-through HMD ad eseguire la piattaforma Windows Mixed Reality con il sistema operativo Windows 10. Dal punto di vista del design, HoloLens 2 è piuttosto ergonomico e si adatta alla testa dell'utente, non crea nessun tipo di problemi se si indossano anche gli occhiali anche perché ha un campo di visione molto

ampio. E' totalmente wireless e dispone di una moltitudine di sensori utili a permettere all'utente di lavorare a mani libere. Entrando più nel dettaglio delle caratteristiche tecniche del visore [8], per il tracciamento della testa ha 4 telecamere a luce visibili, per il tracciamento oculare ha 2 telecamere a raggi infrarossi e possiede una fotocamera centrale da 8 megapixel. Possiede i sensori quali l'accelerometro, il giroscopio, e il magnetometro. Per quanto riguarda i sensori per audio e voce ha dei gruppi di microfoni e l'audio spaziale incorporato. Possiede delle tecniche avanzate per la comprensione umana, ovvero riesce ad effettuare il tracciamento della mano, il tracciamento oculare e la comprensione del linguaggio naturale tramite voce. Riesce a fare anche una comprensione dell'ambiente, in cui può effettuare il tracciamento a 6DOF[1] e permette la mappatura spaziale tramite creazione della mesh in tempo reale. Possiede 4GB di RAM e 64GB di storage. E' in grado di connettersi tramite Wi-Fi, l'uso del Bluetooth 5 e mediante cavo USB di tipo C. La batteria dura circa 2-3 ore di utilizzo attivo ed ha il cavo per la ricarica rapida. Il peso è di 566 grammi.

Magic Leap One

Figura 1.3: Dispositivo Magic Leap One [10]

Magic Leap One ha un aspetto molto diverso rispetto a HoloLens 2, non presenta una visiera che l'utente può alzare o abbassare, ma ha un design binoculare. Magic Leap One, non è un dispositivo totalmente wireless e non permette all'utente di avere le mani libere perché si deve utilizzare come comando di input un controller manuale portatile che è collegato al visore tramite cavo. Essendo uscito nel mercato prima di HololLens 2 possiede sofisticati sensori tuttavia meno potenti. In base alle specifiche tecniche [11], il display che

[1]6DOF: è l'acronimo di sei gradi di libertà in inglese e si riferisce al movimento nello spazio tridimensionale, ovvero l'abilità di muoversi liberamente avanti/indietro, su/giù, sinistra/destra combinati con rotazione lungo tre assi perpendicolari [9].

mette a disposizione è da 1280 x 960 pixel RGB per occhio, risoluzione più bassa rispetto a HoloLens 2 che ne garantisce 2000. Possiede i sensori quali l'accelerometro e il magnetometro. Per quanto riguarda l'audio spaziale ha degli altoparlanti stereo integrati. Possiede 8GB di RAM e 128GB di storage. E' in grado di connettersi tramite Wi-Fi, l'uso del Bluetooth 4.2. La batteria dura circa 3.5 ore di utilizzo attivo ed ha il cavo per la ricarica rapida. Il peso è inferiore rispetto al dispositivo precedente e corrisponde a 316 grammi.

1.4.2 Head-up displays

I dispositivi di tipo *Head-up displays* (HUD), che letteralmente significa visore a testa alta, originariamente sono stati creati per l'aeronautica militare per proiettare, nel campo visivo del pilota, le informazioni a lui utili. In seguito è stato sviluppato anche per integrare la strumentazione delle vetture, in modo da proiettare sul parabrezza le informazioni importanti per il conducente per evitare che distolga lo sguardo dalla strada per guardare il cruscotto.

Figura 1.4: Esempio di head-up display [12]

1.4.3 Hand-held display

I dispositivi *Hand-held display* (HHD), sono dispositivi portatili, come gli smartphone e tablet, che non sono stati progettati appositamente per essere utilizzati per esperienze MR. Infatti, per essere sfruttati con questo scopo hanno bisogno di tecnologie aggiuntive come AR Foundation che permettono lo

sviluppo di applicazioni di augmented reality, il quale però non implementa nessuna funzionalità AR e per questo motivo bisogna importare in base alla piattaforma il plugin ARCore per Android e il plugin ARKit su iOS. I dispositivi mobile utilizzano una tecnologia di tipo video see-through: grazie alla videocamera è possibile vedere il mondo reale al quale vengono aggiunte le immagini 3D.

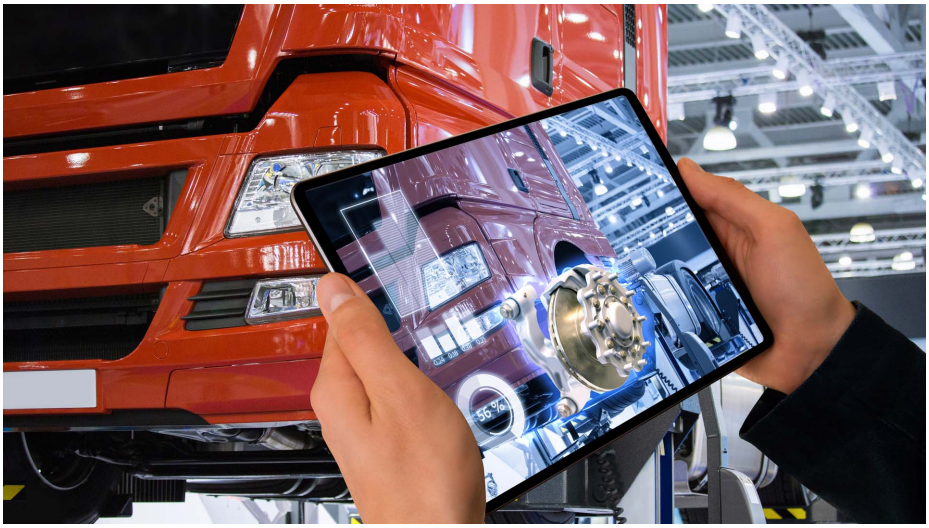


Figura 1.5: Esempio di utilizzo del tablet unitamente alla realtà mista [13]

1.5 Aspetti da considerare per realizzare un'applicazione MR

Le applicazioni di realtà mista sono profondamente diverse da tutte le altre applicazioni esistenti al giorno d'oggi e la progettazione è estremamente complessa. Bisogna tenere conto, non solo delle combinazioni di mondi reali e virtuali che vengono a crearsi ma anche delle nuove esperienze utente che si vogliono offrire. Di seguito verranno descritti i vari aspetti di progettazione da considerare per creare prototipi basati sulla Mixed Reality [14].

1.5.1 Ologrammi

Gli ologrammi sono rappresentazioni digitali di qualsiasi cosa l'essere umano è in grado di immaginare. Sono oggetti fatti di luce e suono che appaiono nel mondo come reali [15]. Non si limitano ad essere oggetti di uso quotidiano, ma possono essere animali, piante, personaggi, possono assumere qualsiasi forma, avere ogni tipo di colore ed emettere suoni. Oltre all'aspetto, possono

avere anche comportamenti diversi. Questi oggetti virtuali sfidano la logica. L'intelletto umano è consapevole del fatto che non siano reali ma comunque si rimane sorpresi di quanto siano convincenti nella fusione con lo spazio fisico. Le persone spesso descrivono la sensazione di interagire con gli ologrammi come ad un qualcosa di magico. Si è momentaneamente ingannati. Gli ologrammi aprono le porte ad un universo di nuove possibilità e fanno sembrare tutto quello che prima era impossibile, possibile. Si può notare come essi siano fondamentalmente una contraddizione: sembrano reali ma sono chiaramente digitali, sono a grandezza naturale ma allo stesso tempo ridimensionabile, reattivi ma non vivi, a portata di mano ma non toccabili. Questo paradosso è il motivo principale per cui si viene così tanto affascinati e per cui gli ologrammi hanno un grande impatto sulle persone. Un altro aspetto interessante degli ologrammi è che possono suscitare una risposta emotiva nella gente che differisce da persona a persona perché sembrano esistere sia nella mente che nello spazio reale contemporaneamente. Un consiglio che viene spesso dato nella letteratura, per i realizzatori di applicazioni di Mixed Reality, è quello di creare una connessione emotiva con la persona impegnata nell'attività [16].

Gli utenti tramite il visore hanno un'estensione del campo visivo che è ridotta rispetto al normale. Possono vedere il mondo della realtà mista attraverso un riquadro di visualizzazione rettangolare chiamato frame olografico. Spesso si tende a sovraccaricare questa area poiché è quella immediatamente visibile all'utente. In verità, viene eseguito un mapping spaziale attorno alla posizione dell'utente per cui tutta quella zona può essere considerata come un'area da disegno potenziale per il contenuto digitale e le interazioni. Una corretta progettazione all'interno di un'esperienza deve incoraggiare l'utente a spostarsi nello spazio e ad esplorarlo, indirizzando l'attenzione sul contenuto principale. Ovviamente questo dipende dallo scopo che si vuole trasmettere all'utente attraverso l'esperienza. Spesso il contenuto olografico che si vuole inserire all'interno del prototipo da realizzare è più grande del frame olografico. La chiave è consentire agli utenti di visualizzare l'ologramma con le dimensioni complete dell'oggetto riducendo la grandezza, in scala più piccola o posizionandolo più distante [17].

Per la progettazione, esistono varie procedure che sono consigliate per quanto riguarda il posizionamento degli ologrammi. Alcuni scenari richiedono che gli elementi virtuali rimangano facilmente individuabili durante tutta l'esperienza. Esistono due approcci ad alto livello per quanto riguarda questo tipo di posizionamento: *display-locked* e *body-locked*. Nel primo, il contenuto è bloccato sul dispositivo di visualizzazione il quale blocca parzialmente la visuale dell'utente che può rimanere infastidito da quella presenza. Nel secondo approccio, invece, l'ologramma è posizionato in base al punto in cui si trova l'utente, ne risulta che l'oggetto virtuale segue la persona, che quindi è libera

di spostarsi senza paura di perderlo e la visuale rimane libera. E' consigliato posizionare gli ologrammi nella zona ottimale tra 1,25 e 5 metri. La distanza di visualizzazione ottimale è di due metri. Se si accorcia la distanza, ovvero ci si avvicina a più di un metro, l'esperienza peggiorerà. Se viene ridotta ancora è possibile che le immagini vengano distorte e quindi si consiglia di dissolvere il contenuto quando si è troppo vicini [15].

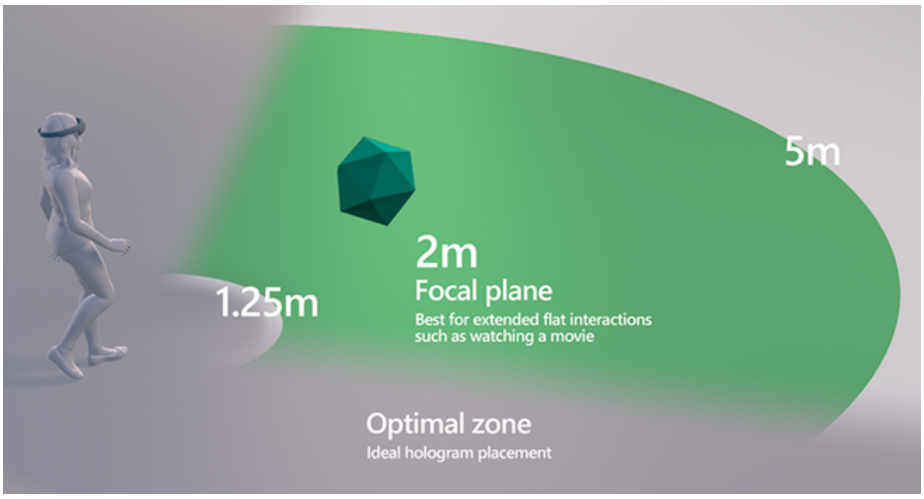


Figura 1.6: Distanza ottimale per posizionare gli ologrammi dall'utente [18]

Il rendering[2] olografico permette all'applicazione di disegnare un ologramma in una posizione precisa attorno all'utente. Il rendering dipende dal tipo di dispositivo usato, ad esempio i dispositivi come HoloLens che sono di tipo see-through, aggiungono luce nel mondo. I pixel bianchi sono traslucidi mentre i pixel neri risultano trasparenti, è per questo motivo che si vede anche il mondo reale. Anche se l'utente ha una percezione del rilievo degli oggetti, dovuta principalmente alla visione binoculare, è possibile aggiungere degli effetti che migliorano l'esperienza utente. Una di queste tecniche consiste nell'aggiungere un bagliore intorno ad un ologramma sulla superficie vicina ed eseguire il rendering di un'ombreggiatura su questo bagliore. In questo modo, si otterrà un effetto che sembrerà che l'ombreggiatura sottrae luce all'ambiente [20]. Un altro importante simbolo di profondità è dato dal suono spaziale. I suoni, in HoloLens provengono da due altoparlanti che si trovano direttamente sopra le orecchie. Anch'essi come i display olografici sono additivi, introducendo nuovi suoni senza bloccare i suoni dell'ambiente reale.

[2]Rendering: nella computer grafica, identifica il processo di resa, ovvero di generazione di un'immagine a partire da una descrizione matematica di una scena tridimensionale, interpretata da algoritmi che definiscono il colore di ogni punto dell'immagine digitale [19]

1.5.2 Sistemi di coordinate

Si è già parlato di come sarebbe meglio collocare gli ologrammi nello spazio, questo però comporta il fatto che devono essere posizionati e orientati in modo preciso sia nell'ambiente virtuale sia reale. Nella documentazione vengono forniti vari sistemi di coordinate chiamati **sistemi di coordinate spaziali**. Tutte le applicazioni di grafica 3D utilizzano sistemi di coordinate cartesiane. I valori delle coordinate sono espresse in metri, questo significa che gli oggetti posizionati a una unità di distanza nell'asse X, Y o Z verranno visualizzati a un metro di distanza l'uno dall'altro quando ne viene eseguito il rendering. Esistono due tipi di sistemi a coordinate cartesiane, *right-handed* oppure *left-handed*. Il programmatore che sviluppa applicazioni per la realtà mista deve tenere conto che i sistemi di coordinate spaziali su Windows sono sempre a destra. Questo significa che l'asse X positivo punta verso destra, l'asse Y positivo punta verso l'alto (allineato alla gravità) e l'asse Z positivo verso l'utente [21]. Un modo semplice di ricordarsi in quali direzioni gli assi sono positivi è utilizzando la regola della mano destra o sinistra, puntando la mano destra o sinistra nella direzione dell'asse X positivo e arricciando le dita nella direzione della Y positiva. La direzione verso cui punta il pollice, verso o fuori dall'utente è la direzione verso cui punta l'asse Z positivo.

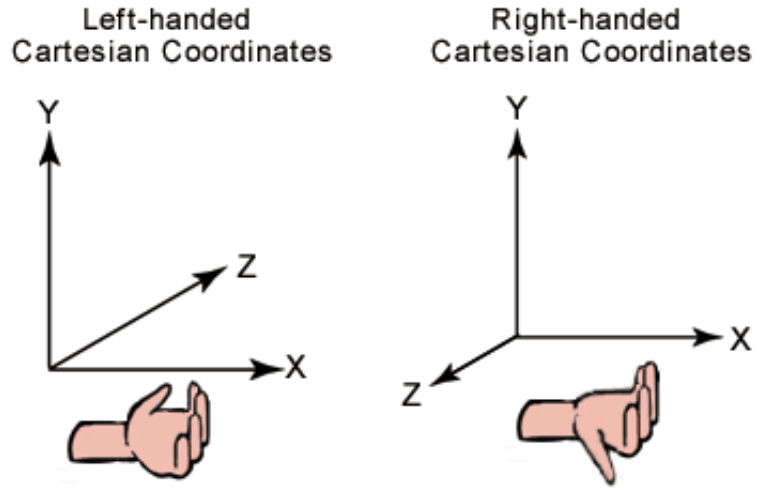


Figura 1.7: Sistemi di coordinate [22]

1.5.3 Mappatura spaziale

Un altro aspetto molto importante da considerare nella progettazione è la mappatura spaziale. Dispositivi come HoloLens tracciano costantemente l'ambiente e costruiscono un modello 3D che rappresenta l'area in cui ci si trova.

Senza *mesh*, così viene chiamato lo *spatial mapping*, non sarebbe possibile avere la consapevolezza spaziale e quindi poter far interagire tra loro gli ologrammi e gli oggetti reali. La mappatura spaziale è rilevante per diverse motivazioni [23]:

- occlusione: di questo argomento se ne è parlato anche nel paragrafo 1.3, permette di appoggiare gli oggetti virtuali su superfici reali e di nasconderli dietro ad altri.

- posizionamento: permette all'utente di interagire con la mappa spaziale per appuntare oggetti alle pareti, permettere ai personaggi di sedersi sul divano (come si può vedere nell'applicazione Fragments di Microsoft) o decorare l'ambiente circostante automaticamente.

- persistenza: significa che gli ologrammi rimarranno nella stessa posizione in cui l'utente li ha lasciati anche dopo lo spegnimento del dispositivo. Sarà in grado di ricordare lo spazio e ripristinare tutti gli elementi che erano presenti.

- fisica: insieme all'occlusione rende l'esperienza utente più realistica. Permette agli oggetti di scontarsi tra di loro o rimbalzare contro il pavimento e i mobili. In pratica agiscono e si comportano come veri e propri oggetti reali sottostanti alle leggi fisiche, come ad esempio la forza di gravità.

- navigazione: consente ai personaggi olografici la capacità di navigare nel mondo reale nello stesso modo in cui lo farebbe una persona reale.

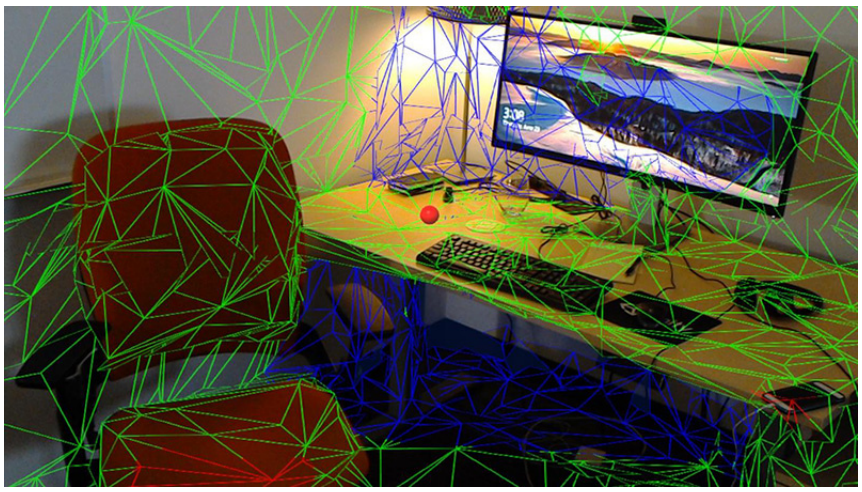


Figura 1.8: Esempio di mappatura spaziale applicata al mondo reale [24]

1.5.4 Interazioni con gli ologrammi

Per una buona progettazione di applicazioni di realtà mista, gli sviluppatori devono cercare di mantenere le interazioni più semplici possibili e intuitive. In generale esistono quattro diversi modi per interagire con gli ologrammi [23]:

- *Gaze*: lo sguardo è il principale metodo che ha l'utente per concentrarsi su ologrammi e oggetti. E' importante e svolge la stessa funzionalità del mouse per il PC. Ovvero, si usa la direzione in cui si guarda per puntare agli oggetti nello spazio 3D così come si usa il mouse per puntare agli oggetti sullo schermo. La posizione del mouse è rappresentata da un cursore, mentre quella dello sguardo è rappresentato da un piccolo punto. In verità, lo sguardo è controllato da movimenti della testa e non dal movimento fisico degli occhi. A questo scopo, prima di iniziare l'esperienza, per il tracciamento oculare, verrà chiesto all'utente di effettuare un processo di calibrazione.

- *Gestures*: i gesti coinvolgono l'uso delle mani per manipolare e controllare gli ologrammi. I gesti possono essere: di air-tap o selezione, home, di presa, manipolazione e navigazione. Gli utenti possono usare la manipolazione diretta con le mani oppure con la modalità di puntamento effettuando in seguito il commit. Quest'ultima modalità permette all'utente di interagire con l'ologramma a distanza.

- *Voice*: l'input vocale è sostanzialmente l'uso di comandi vocali per interagire. Gli utenti possono dire una parola o una serie di parole per selezionare oggetti e/o attivare funzioni. In questo caso, bisogna prestare attenzione ai comandi che si usano e al vocabolario di riferimento che dev'essere semplice, evitare suoni simili, eseguire test con accenti diversi ed evitare di usare i comandi di sistema. La voce può essere usata anche al posto di una tastiera per un veloce inserimento del testo tramite dettatura.

- *Motion controllers*: i controller di movimento estendono le funzionalità fisiche dell'utente, servono ad interagire in modo preciso con gli oggetti virtuali e possiedono delle accurate caratteristiche di posizionamento spaziale. E' la modalità di interazione più diffusa per applicazioni di realtà mista, perché non tutti i dispositivi hanno i sensori che supportano le altre modalità.

Oltre ai metodi di input elencati precedentemente, esiste una ampia gamma di dispositivi hardware che possono essere utilizzati con il visore, come ad esempio la tastiera, il mouse e il gamepad bluetooth [23].

A questo punto ci si potrebbe chiedere qual è il metodo di interazione migliore. Di seguito sono riportate alcune domande che possono aiutare il programmatore a scegliere un modello di interazione [25]:

1. Gli utenti vogliono toccare gli ologrammi ed eseguire manipolazioni olografiche di precisione?
   In tal caso, sarebbe più opportuno utilizzare il modello di interazioni con le mani o con il controller di movimento per la precisione di destinazione e manipolazione.

2. Gli utenti devono avere le mani libere per effettuare le attività reali?
   Allora, si consiglia di esaminare il modello di interazione *hands-free*, che offre un'esperienza senza mani tramite lo sguardo o la voce.

3. Gli utenti hanno tempo per imparare le interazioni per l'applicazione MR oppure hanno bisogno di interazioni che hanno la più bassa curva di apprendimento possibile?
   In questo caso, se si vuole avere un'interazione semplice e intuitiva è raccomandato usare mani o con il controller di movimento perché l'utente è più abituato.

1.5.5 Elementi dell'esperienza utente

Dopo aver visto gli elementi principali, si possono vedere altre caratteristiche che lo sviluppatore deve tenere in considerazione per la progettazione di applicazioni di realtà mista, tra cui la scelta del colore, la luce, i materiali e la tipografia. In base allo scopo possono essere effettuate scelte diverse. Ciascuna decisione deve essere ponderata rispetto ai vincoli del dispositivo e allo scopo di destinazione dell'esperienza.

1.5.6 Esperienze condivise

Una caratteristica di fondamentale importanza è la possibilità di condivisione delle esperienze di realtà mista. Le applicazioni olografiche consentono la condivisione di ologrammi tramite gli ancoraggi spaziali da un dispositivo HoloLens, iOS o Android ad un altro. Permette sessioni condivise in cui gli utenti possono collaborare allo stesso progetto o vedere la stessa esperienza insieme. Per poter progettare esperienze condivise è necessario definire, per prima cosa, quali sono gli scenari di destinazione. Per fare questo Microsoft, durante la creazione di prototipi per HoloLens, ha definito sei domande, che fungono da linee guida, che possono essere utili allo scopo [26].

1. *How are they sharing?*
   Esistono diversi modi per condividere, ma si possono distinguere tre macro categorie principali:

   - Presentazione: si definisce tale quando lo stesso contenuto viene visualizzato da più utenti. Può essere il caso, ad esempio, di un insegnante che mostra un contenuto olografico a diversi studenti.
   - Collaborazione: si determina quando per raggiungere determinati obiettivi comuni le persone lavorano insieme.
   - Indicazione: si ottiene quando c'è una persona che aiuta a risolvere un problema in un'interazione uno a uno.

2. *What is the group size?*
   Si è scoperto che la complessità nel realizzare esperienze condivise di gruppo possono aumentare esponenzialmente quando si passa da gruppi di piccole dimensioni a grandi. Dove per grandi dimensioni ci si riferisce ad un insieme superiore a 7 persone, piccolo si intende superiore a 1 e inferiore a 6 e poi ci sono le esperienze condivise tra sole due persone. La condivisione con gruppi di grandi dimensioni può causare difficoltà sia tecniche, per quanto riguarda la connessione di rete e la trasmissione di dati, sia social a causa dell'impatto di essere in una stanza con diversi avatar.

3. *Where is everyone?*
   Le esperienze condivise possono essere classificate anche in base al luogo in cui si trovano le persone. Se l'esperienza condivisa ha luogo nella stessa posizione e quindi tutti gli utenti si trovano nello stesso spazio fisico, allora viene chiamata *colocated*. Se invece, gli utenti si trovano in spazi fisici separati, l'esperienza condivisa viene detta *remote*. Inoltre ci sono i casi misti in cui vi sono delle persone collegate dallo stesso posto e altre da remoto. Questo punto è importante per riuscire a capire come far comunicare tra loro le persone e se gli ambienti non sono condivisi come dev'essere fatto.

4. *When are they sharing?*
   Quando si pensa ad una esperienza condivisa, immediatamente viene in mente lo scenario in cui lo *sharing* avviene nello stesso momento, cioè in modo sincrono. Tuttavia, si possono avere anche degli scenari in cui si aggiunge ad esempio un elemento virtuale e questo viene visualizzato in seguito da altri utenti. In quest'ultimo caso la condivisione avviene in modo asincrono, l'esperienza olografica è condivisa in momenti diversi. Esistono anche i casi misti in cui a volte gli utenti condividono in

modo sincrono e altre in modo asincrono. Capire quando viene condiviso qualcosa influisce sulla problematica della persistenza dell'oggetto nell'ambiente. Se viene effettuata una modifica, deve essere salvata per poter essere visualizzata in un secondo momento da un altro utente.

5. *How similar are their physical environments?*
   Le esperienze possono differire anche in base alle caratteristiche dell'ambiente. Idealmente, avere ambienti identici migliora la *user experience* ma le probabilità sono molto scarse a meno che non siano state progettate per essere tali. L'ambiente può essere simile, si ha una simile disposizione dei mobili, luce, suoni ambientali e dimensione della stanza. Si può far riferimento ad esempio alle sale conferenza, tipicamente composte da un tavolo centrale circondato da sedie. Oppure può essere dissimile, gli ambienti differiscono in tutto. Bisogna tenere in considerazione come è fatto l'ambiente in quanto potrebbe cambiare la percezione che hanno gli utenti degli oggetti virtuali in relazione agli ambienti reali.

6. *What devices are they using?*
   Al giorno d'oggi è più probabile che le esperienze condivise avvengano tra dispositivi immersivi o tra dispositivi olografici. Tuttavia è possibile effettuare anche la condivisione tra dispositivi 3D e 2D, quali cellulari, tablet o computer. Capire quali dispositivi si utilizzano è utile per gestire in modo differente le cose dato che si hanno vincoli diversi.

1.6 Applicazioni mediche

La Mixed Reality sfrutta i display dei *wearable device*, dei dispositivi *mobile*, dei vetri dei veicoli e altri dispositivi per aggiungere informazioni olografiche all'ambiente reale in cui si vive. Gli ambiti applicativi stanno crescendo sempre di più: ambito medico, industriale, militare, educativo, per i musei, per intrattenimento e molti altri. Uno tra quelli più importanti è però il settore dell'healthcare ed è quello in cui si è sviluppato l'elaborato di tesi.

In letteratura sono già presenti applicazioni di realtà mista sviluppate in ambito sanitario che svolgono determinate funzioni in base allo scopo medico. Fino ad ora queste tecnologie innovative e all'avanguardia sono state utilizzate solo a scopo di ricerca e sperimentazione ma adesso, che si stanno sempre più espandendo, ci si può aspettare che entrino a far parte della pratica medica degli operatori sanitari.

Nella sanità i mondi virtuali, aumentati e misti vengono utilizzati principalmente per quattro scopi:

- riabilitazione o terapia;
- diagnosi o cura;
- gestione del dolore, affrontare fobie o altri tipi di disturbi;
- formazione degli studenti o dei professionisti mediante simulazioni.

Nella riabilitazione, la realtà mista viene integrata a meccanismi robotici per incentivare l'esercizio e l'attività degli arti compromessi. Per quanto riguarda la gestione del dolore, è stato effettuato uno studio pubblicato su *PLOS ONE* [27] in cui i ricercatori, dopo aver effettuato tantissimi esperimenti e prove, si sono resi conto che facendo uso della realtà virtuale come strumento di distrazione si ha una effettiva riduzione del dolore senza uso di farmaci. La formazione tramite mixed reality è un buono strumento per insegnare senza correre rischi, ad esempio simulando una realtà ospedaliera con tutte le difficoltà del caso ma senza conseguenze reali.

Sono molteplici i progetti creati da software house importanti che si pongono gli stessi obiettivi di questo elaborato di tesi avente come scopo primario la diagnosi e sostegno all'assistenza sanitaria. Qui di seguito verranno presentate due importanti applicazioni: *Verima* e *Medivis*.

1.6.1 Verima

La prima applicazione che si vuole presentare è stata creata da un'azienda fiorentina di ingegneria informatica **WITAPP** che si impegna a realizzare prodotti digitali innovativi per migliorare l'imaging sanitario nel campo della diagnostica e il *Clinical Decision Support*. Il software è stato sviluppato da **WITHECA** un gruppo di ricerca appartenente a Witapp, dedicata allo sviluppo di prodotti nel settore healthcare, con la collaborazione e la supervisione scientifica dell'Azienda Ospedaliero-Universitaria di Careggi di Firenze.

Witapp crede che le conoscenze e le competenze informatiche, unite a quelle in campo medico, svolgano un ruolo di essenziale importanza nella tutela della salute delle persone. Per questo motivo si vuole affiancare lo svolgimento del lavoro quotidiano dei medici con tali strumenti tecnologici [28].

Il software chiamato **Verima**, con l'uso della realtà aumentata unitamente alla realtà mista, permette la ricostruzione di una TAC e di una Risonanza Magnetica sotto forma di ologramma interattivo. Il quale è la riproduzione fedele di un organo o uno scheletro, ottenuto grazie al sistema di segmentazione

automatica, elaborato da un modello di deep learning addestrato a riconoscere i diversi tessuti anatomici. Sarà possibile condividere le informazioni tra medici a distanza, in modo da permettere a due o più professionisti di confrontarsi su determinate tematiche per trovare la soluzione migliore. Inoltre, gli ologrammi sono disponibili anche su device di uso comune come smartphone e tablet, usando l'apposita applicazione per mobile "Verima Vewer" che si può trovare nello store iOS e Android.

Questo applicativo è stato creato per mettere a disposizione dei medici uno strumento che migliora la visualizzazione, la pianificazione degli interventi e le attività di supporto alla diagnosi. Tramite questa tecnologia si riducono considerevolmente i tempi per produrre i referti, oltre a ridurre i costi e i tempi per effettuare diagnosi, che tuttavia risultano più precise, accurate e dettagliate [29].

(a) [30]

(b) [31]

Figura 1.9: Esempi di visualizzazione di organi con l'applicazione Verima

1.6.2 Medivis

Medivis è l'applicazione realizzata da Microsoft per fornire una migliore assistenza sanitaria [32]. Fornisce ai chirurghi e ai medici gli strumenti necessari a migliorare la precisione nelle operazioni chirurgiche e offre risultati ottimi per i pazienti. Consente ai team di assistenza di collaborare da remoto e condurre consulenze virtuali con i pazienti in tempo reale in modo da accelerare le diagnosi e ridurre i tempi per il trattamento. In pratica offre un servizio di telemedicina. Usa la realtà mista per sovrapporre le viste 3D delle immagini di una risonanza magnetica e le scansioni tomografiche per assistere gli specialisti prima e dopo l'intervento, migliorando la precisione. Consente agli operatori sanitari di condividere rapidamente i risultati per contestualizzare meglio le conversazioni tra medico e paziente. In questo modo il paziente è più consapevole per poter prendere determinate decisioni. Inoltre, può essere utilizzata per l'apprendimento, i medici e gli infermieri possono formarsi esercitandosi con le simulazioni.

Figura 1.10: Esempio in cui viene usata l'applicazione Medivis [33]

Capitolo 2

Progetto di tesi

Il progetto di tesi che si vuole realizzare è nato a scopo di ricerca con la collaborazione del dipartimento di Ingegneria Biomedica. Il sistema che si vuole realizzare deve fornire un'interfaccia utente che permetta all'operatore sanitario di poter visualizzare e manipolare immagini biomediche con il corrispondente modello 3D. E' possibile decidere quale modello mostrare scegliendolo da un apposito menù che appare all'avvio dell'applicazione. Il prototipo da sviluppare dev'essere basato su Realtà Mista, usando HoloLens 2 come tecnologia di riferimento. I colleghi hanno fornito i modelli 3D e i file DICOM con cui poter verificare il funzionamento del programma. L'obiettivo primario è quello di fornire alle strutture ospedaliere uno strumento di supporto, principalmente a fine diagnostico, da affiancare ai metodi tradizionali utilizzati nelle attività cliniche. Le funzionalità presentate hanno ampie potenzialità di crescita.

2.1 Obiettivi del progetto

Il progetto che si vuole sviluppare, si pone determinati obiettivi da raggiungere. Si vuole realizzare un'applicazione che soddisfi i seguenti requisiti:

- mostrare un menù che visualizzi gli ultimi modelli caricati;
- caricare il modello 3D e il corrispondente file DICOM al click del bottone;
- essere possibile caricare più modelli 3D e file DICOM durante la stessa sessione;
- dare la possibilità di manipolare l'ologramma (spostarlo, girarlo, ingrandirlo o rimpicciolirlo);

- dare la possibilità di scorrere il file DICOM per visualizzare le immagini in corrispondenza del modello 3D;
- dare la possibilità di visualizzare il bordo del modello davanti all'immagine;
- dare la possibilità di caricare i file a *run-time*;
- dare la possibilità di bloccare o sbloccare il menù affinchè segua la posizione dell'utente;
- tempi di caricamento più rapidi;
- funzionare sul dispositivo HoloLens 2;

2.1.1 Casi d'uso

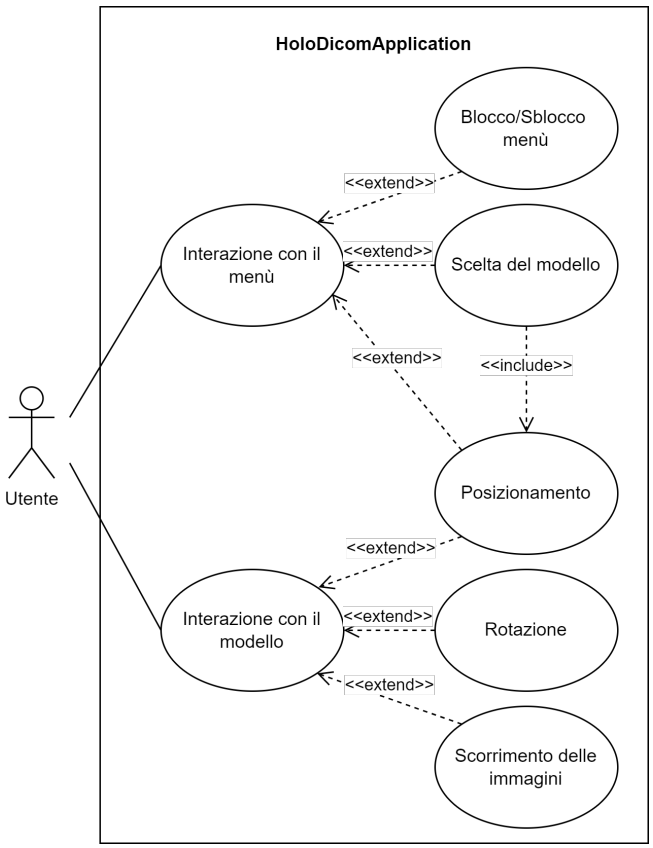


Figura 2.1: Diagramma dei casi d'uso

In questo paragrafo verranno analizzate le possibili operazioni che l'utente può effettuare, attraverso i diagrammi dei casi d'uso. Un diagramma dei casi d'uso è un tipo di diagramma UML (*Unified Modeling Language*) che viene usato spesso per analizzare un sistema. Permettono di visualizzare i ruoli esistenti in un sistema, che possono essere umani, organizzazioni ed enti o altre applicazioni, e come essi interagiscono con il sistema stesso. Come si può notare dal diagramma in Figura 2.1, in questa applicazione l'attore è l'utente ed esistono due casi d'uso principali che può scegliere di effettuare durante l'esecuzione. Il primo è "interazione con il menù", il quale è esteso da altri casi d'uso, che raffigurano le possibili azioni che l'utente può compiere. Queste operazioni sono collegate a "interazione" tramite la relazione $<< extend >>$ e si intende un comportamento opzionale, ovvero il caso d'uso principale è completo anche senza l'estensione. Una volta che è iniziata l'interazione, l'utente è libero di scegliere di eseguire una delle operazioni o nessuna. L'utente può bloccare/sbloccare il menù per fare in modo che lo segua nei suoi spostamenti oppure per fissarlo in un punto specifico e può inoltre voler modificare la posizione dello stesso. L'interazione più importante con il menù è però, la "scelta del modello" tra le varie opzioni disponibili presenti nel menù. Questa operazione viene effettuata subito all'avvio dell'applicazione. In aggiunta, si è inserita la relazione $<< include >>$ tra il caso d'uso "scelta del modello" e "posizionamento". Tale collegamento indica che una volta che viene selezionato un modello, viene usata in automatico dal sistema almeno una volta obbligatoriamente l'operazione per il posizionamento del modello. Il secondo caso d'uso riguarda l'operazione di "interazione" con il modello scelto precedentemente e il corrispondente file DICOM caricato. Anche in questo caso è esteso da altri casi d'uso. L'utente può spostare la posizione del modello, ruotarlo, ridimensionarlo e far scorrere le immagini.

Prima di iniziare la trattazione della parte progettuale e di implementazione del prototipo, è necessario introdurre e spiegare il significato di alcuni termini importanti, ovvero si descriverà il concetto di DICOM e di modello 3D e perché sono utili.

2.2 DICOM

DICOM è l'acronimo di *Digital Imaging and COmmunications in Medicine* e si tratta di uno standard internazionale di integrazione software utilizzato per la gestione delle immagini e comunicazioni digitali in medicina. In particolare, si occupa di stabilire delle regole per la comunicazione, la visualizzazione, l'archiviazione e la stampa di informazioni di tipo biomedico. Si potrebbe pen-

sare ad esso come ad uno degli standard di messaggistica sanitaria più diffusi al mondo. Spesso ci si riferisce col termine DICOM solamente ad un formato d'immagine ma questa definizione è solo parzialmente corretta, in quanto è sia un protocollo di comunicazione sia un formato di file. La motivazione principale per cui si aderisce ad uno standard è per consentire l'interoperabilità.

Il progetto originario fu sviluppato da due organizzazioni statunitensi quali ACR (The American College of Radiology), NEMA (National Electrical Manufacturers Association), da un comitato europeo e dall'associazione giapponese JIRA. Nel 1985 venne ufficializzata la prima versione, chiamata ACR-NEMA a cui seguì nel 1988 la seconda versione con una prima definizione del formato dei file contenenti le immagini e un primo approccio di protocollo per l'interconnessione punto punto tra le apparecchiature. Nel 1993 venne pubblicata una nuova versione: pur mantenendo invariate le specifiche inerenti al formato delle immagini, furono implementati i protocolli di rete TCP/IP e ISO/OSI e aggiunti nuovi servizi. Il nuovo standard venne identificato come DICOM e rivoluzionò la pratica della radiologia, rendendo possibile la sostituzione della pellicola radiografica con un flusso di lavoro completamente digitale. Di conseguenza, si è ottenuta una forma accurata e completa di informazioni dal punto di vista diagnostico in pochi minuti. Lo standard DICOM fornisce un'assistenza sanitaria più rapida e migliore in quanto il radiologo non ha più bisogno di conservare le pellicole, con il rischio di perderle o danneggiarle.

Lo standard DICOM ha perciò garantito migliori prestazioni alla medicina contemporanea, fornendo allo stesso tempo le seguenti funzionalità [34]:

- **Standard universale per la medicina digitale.** Tutti gli attuali dispositivi di acquisizione di immagini mediche digitali creano immagini DICOM e comunicano attraverso reti DICOM.

- **Immagini ad alta qualità.** DICOM supporta fino a 65.536 (16 bit) sfumature di grigio per la visualizzazione di immagini monocromatiche, catturando anche le più piccole sfumature nell'imaging medico. Si avvale delle tecniche più avanzate per la rappresentazione delle immagini digitali per fornire la massima qualità dell'immagine diagnostica.

- **Supporto completo per numerosi parametri di acquisizione dell'immagine e diversi tipi di dato.** DICOM non solo memorizza le immagini, ma registra anche una grande quantità di altri parametri relativi ad essa come ad esempio la posizione del paziente, le dimensioni, l'orientamento, lo spessore della *slice*, le modalità di acquisizione, i filtri di elaborazione dell'immagine e così via. Questi dati arricchiscono il con-

tenuto informativo delle immagini DICOM. Per fare un esempio pratico, sono informazioni essenziali alla creazione di immagini 3D.

- **Codifica completa dei dati medici.** I file DICOM usano un numero altissimo di attributi standardizzati, definiti nel DICOM *Data Dictionary*, per trasmettere i dati medici come le informazioni del paziente, i dati relativi alla diagnosi corrente, i dati che si riferiscono allo studio medico e il dispositivo utilizzato.

- **Chiarezza nella descrizione dei dispositivi e nelle funzionalità dell'imaging digitale.** DICOM mette a disposizione delle interfacce e definisce delle funzionalità per poter lavorare con dispositivi medici rendendo il processo più semplice.

Il DICOM è diffuso in tantissimi settori della medicina, ovunque ci sia bisogno di imaging medico (alcuni esempi: radiologia, cardiologia, oncologia, radioterapia, nefrologia, neurologia, ortopedia, dermatologia, chirurgia, odontoiatria e molte altre). E' difficile immaginare la moderna medicina digitale senza DICOM.

E' implementato in quasi tutti i dispositivi di radiologia, tra cui la **tomografia comuterizzata (CT)**, che combina l'imaging a raggi X e la potenza dell'elaborazione del computer; la tecnica diagnostica della **risonanza magnetica (MRI)**, un sistema che utilizza campi magnetici naturali per acquisire le immagini degli organi del corpo; l'**ecografia** che è una modalità d'indagine che si basa sugli ultrasuoni sfruttando il principio dell'emissione di eco; la radiografia ottenuta mediante **raggi X**, costituiti da radiazioni elettromagnetiche.

2.2.1 Struttura dei file

Un file DICOM è composto da un'intestazione (header) costituita da un insieme di attributi contenenti informazioni in grado di identificarlo univocamente e da un corpo avente una o più immagini. I dati che formano l'intestazione sono organizzati in quattro livelli di gerarchia:

- "*Patient*": corrisponde alla persona a cui vengono effettuati gli esami

- "*Study*": ssi tratta dello studio medico che effettua gli esami, può essere privato o anche pubblico come l'ospedale

- "*Series*": un esame può essere svolto più volte allo stesso paziente da parte dello stesso studio e quindi bisogna mantenere traccia di quale serie appartiene

- "*Instance*" (*Image*): corrisponde all'immagine del DICOM.

DICOM adotta una forma di astrazione ispirata a quella del paradigma object-oriented. La programmazione orientata agli oggetti è un tipo di paradigma di programmazione, caratterizzato da un insieme di concetti di astrazione, in cui un programma viene visto come un insieme di oggetti che interagiscono tra di loro e ognuno ha uno stato, un comportamento e un'identità. Per questo motivo nella documentazione spesso vengono descritti mediante il Modello ER (*Entity Relationship*).

Un ***Information Object Definition*** **(IOD)** è un oggetto informativo utilizzato per rappresentare i dati del mondo reale. Un IOD non simboleggia una specifica istanza di un oggetto reale, ma piuttosto una classe di oggetti raggruppati in base alle relazioni che esistono tra di loro, convividono le stesse proprietà. Un IOD è una collezione di informazioni correlate, raggruppate in entità informative (***Information Entity*** **(IE)**). Ciascuna entità contiene informazioni su un singolo oggetto, ad esempio su un paziente, su un'immagine, sulla modalità di acquisizione. Le entità informative contengono gli attributi, ciascuno dei quali descrive una informazione. Ad esempio l'entità informativa Paziente possiede come attributi: il nome del paziente, la data di nascita, il sesso, il peso e così via tanti attributi quanti sono necessari a catturare tutte le informazioni clinicamente rilevanti sul paziente. Le IE corrispondono alle informazioni salvate nell'header: **Paziente**, **Studio**, **Serie**, **Istanza** (Immagine) ma ne esistono tante altre come: Visit, Equipment, Clinical Trial, Procedure.

Gli algoritmi di sicurezza moderni permettono una protezione delle informazioni dei file DICOM anche se questi non sono accessibili pubblicamente. Tramite l'*encryption*, un processo in cui l'intero file è riscritto in forma crittografata, è impossibile da leggere se non si ha la chiave. La sicurezza dei file DICOM fornisce le seguenti proprietà [34]:

- Confidenzialità dei dati: se i file DICOM vengono rubati non è possibile leggere i dati perchè non sono pubblici.
- Autenticazione dei dati di origine: grazie alla firma digitale si può sapere esattamente a chi appartengono i dati.
- Integrità dei dati: nessuno può modificare i dati.

I colleghi di biomedica forniscono i file DICOM segmentati in cui è stata apportata una protezione in quanto non è possibile vedere le informazioni sensibili riguardanti lo studio medico che le ha fornite ma soprattutto quelle relative al paziente.

Solitamente i file DICOM vengono salvati con l'estensione ".dcm", ma sono possibili anche altre estensioni quali ".dic", ".dc3", oppure nessuna.

2.2.2 Orientazione dell'immagine nello spazio

Nel mondo medico esistono tre tipologie di sistemi di coordinate che vengono comunemente usati nelle applicazioni di imaging medico: il mondo, anatomico e dell'imaging medica [35]. Il sistema di coordinate del mondo si tratta di quello basato sulle coordinate cartesiane.

Il sistema di coordinate anatomiche è costituito da tre piani:

- *Axial*: si tratta di un taglio trasversale che divide il paziente a metà, dalla parte superiore alla inferiore. Questo tipo di vista viene chiamata dall'alto e separa la testa dai piedi del paziente.
- *Sagittal*: in questo caso si ha una vista laterale e il paziente è diviso in lato destro e sinistro e perciò il taglio è verticale.
- *Coronal*: questo piano separa la parte anteriore dalla posteriore del paziente.

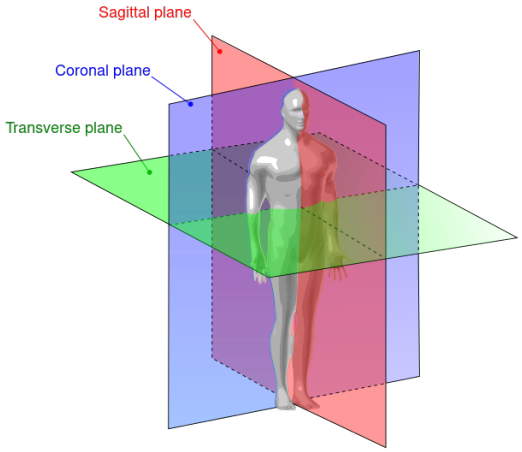


Figura 2.2: Piani anatomici [36]

Per quanto riguarda il sistema di coordinate dell'imaging medica non esiste un solo modo per definire la posizione e l'orientamento. I due modi più diffusi sono:

- LPS, abbreviazione di *Left, Posterior, Superior*. Indica che la direzione dell'asse X va da destra a sinistra, l'asse Y va dal lato anteriore al posteriore e l'asse Z va dalla parte inferiore a quella superiore.
- RAS, abbreviazione di *Right, Anterior, Superior*. Indica che la direzione dell'asse X va da sinistra a destra, l'asse Y va dal lato posteriore a quello anteriore e l'asse Z va dalla parte inferiore a quella superiore.

E' importante sapere che DICOM utilizza il sistema di coordinate LPS, mentre RAS viene utilizzato da software medici quale ad esempio 3D Slicer.

2.3 Modello 3D

Un modello 3D è una rappresentazione matematica di un oggetto tridimensionale. Vengono utilizzati in tantissimi settori, tra cui:

- il campo medico che usa i modelli 3D per rappresentare gli organi provenienti da MRI o TC;
- l'industria cinematografica li usa per rappresentare personaggi e oggetti nei film d'animazione;
- l'industria videoludica come contenuti nei videogiochi;
- in architettura vengono usati per simulare gli edifici e i paesaggi.

Di seguito si vogliono presentare due tipologie di formato di file dei modelli 3D, anche se ne esiste una grande varietà:

- STL (*STereoLithography*, anche chiamato *Standard Triangulation Language*) è un formato di file, binario o ASCII nato per i software di stereolitografia[1] CAD ed è utilizzato spesso nella prototipazione rapida. Risulta semplice in quanto facile da processare e proprio per questo motivo però non è preciso. Descrive la superficie di un oggetto tramite la costruzione di una mesh triangolare, ovvero una rappresentazione di una superficie tridimensionale con faccette triangolari. Il formato .stl standard non supporta la definizione di colore [37].
- OBJ è un formato di file sviluppato da Wavefront Technologies per la creazione di geometrie 3D. All'interno del file vengono definite le posizioni di ogni vertice, la posizione di ogni coordinata UV[2] per la texture, le normali e le facce che compongono il modello.

Il formato di file .stl è quello maggiormente utilizzato dai colleghi di Ingegneria Biomedica. Processano le immagini DICOM acquisite dai pazienti mediante algoritmi di segmentazione per introdurle in dei software per la modellazione 3D ed ottenere, quindi il modello. La segmentazione è un processo di partizione di un'immagine: vengono segmentate le regioni più significative e serve ad ottenere una forma più compatta del modello. A partire da questo vengono poi effettuate delle modifiche e degli aggiustamenti e viene generato il formato di file .obj che serve per l'applicazione che si andrà a realizzare. Si

[1] Stereolitografia: tecnica che permette di realizzare singoli oggetti tridimensionali a partire direttamente da dati digitali elaborati da un software CAD/CAM.

[2] Coordinate UV: una texture viene immaginata giacere su un piano cartesiano e i due assi vengono chiamati u e v.

è richiesto specificatamente questo formato di file perché oltre ad essere il più diffuso, possiede anche delle caratteristiche migliori.

Questa tesi si pone come obiettivo quello di realizzare un'applicazione per la Mixed Reality utilizzando un certo insieme di tecnologie di sviluppo. Prima di entrare nel merito della trattazione della progettazione e implementazione del prototipo verranno mostrate appunto, quali sono queste tecnologie e qual è il contributo che apportano al progetto. I framework di sviluppo utilizzati e che si descriveranno sono: Unity, MRTK e OpenXR.

Capitolo 3

Tecnologie di sviluppo per MR

Microsoft mette a disposizione delle risorse web in cui vi sono tutte le informazioni necessarie allo sviluppo di applicazioni di realtà mista, oltre a fornire esempi di applicazioni già esistenti. Tra le varie informazioni troviamo la pagina per installare alcuni software: Visual Studio 2019, l'emulatore di HoloLens 2 nel caso non si abbia il dispositivo fisico e il motore grafico Unity [38].

3.1 Unity

Unity è un motore grafico multipiattaforma, realizzato da Unity Technologies, che permette ai programmatori e non solo, di creare videogiochi 2D e 3D e altri contenuti interattivi. Inizialmente veniva sfruttato per lo più nello sviluppo di giochi, successivamente grazie alle varie versioni migliorative, è stato adoperato anche per la creazione di applicazioni di Extended Reality. Permette di controllare l'illuminazione, la telecamera (punto di vista), l'audio, importare e manipolare oggetti e modelli 3D, gestire eventi e i comportamenti per ciascuno di esso. Il motivo principale per cui viene utilizzato dai programmatori per lo sviluppo di applicazioni è perché molte delle funzionalità vengono gestite direttamente da Unity, senza che gli sviluppatori intervengano a basso livello. E' un grande vantaggio, ma anche uno svantaggio se si vogliono implementare funzionalità specifiche che l'interfaccia non presenta.

Essendo un ambiente di sviluppo, la maggior parte delle persone che lo utilizzano sono programmatori, ma non esclusivamente perché esso possiede un'interfaccia grafica intuitiva grazie al quale è possibile realizzare funzionalità semplici. È possibile aumentare la complessità dell'applicazione aggiungendo degli script in C#.

Unity supporta tre diversi scripting backend a seconda della piattaforma di destinazione: Mono, .NET e IL2CPP. Secondo la documentazione di Unity, lo scripting backend viene definito come un framework che alimenta lo scripting

in Unity. Con il rilascio della versione 2018.2 di Unity, .NET è stato deprecato, quindi non verrà preso in considerazione in questo elaborato. IL2CPP (Intermediate Language To C++) è un'alternativa al backend Mono, utilizzata di default, fornisce un migliore supporto per le applicazioni su più piattaforme. In pratica converte il MSIL (Microsoft Intermediate Language), ovvero il linguaggio C# negli script, in codice C++, che poi usa per creare un file binario nativo (come per esempio .exe, .apk o .xap) per la piattaforma scelta. Questo tipo di compilazione è chiamato AOT (Ahead-Of-Time), in cui Unity compila il codice in modo specifico per una piattaforma di destinazione, comporta però dei tempi di esecuzione più lunghi. Il backend Mono, invece, compila il codice in fase di esecuzione, con una tecnica chiamata JIT (Just-In-Time compilation) ed è più rapido rispetto a IL2CPP. Ogni scripting backend presenta dei vantaggi e degli svantaggi che il programmatore deve tenere in considerazione.

Unity, ha fornito delle informazioni riguardanti il migior scripting backend da utlizzare in base alla piattaforma in uso. Nel progetto si farà uso della piattaforma ***Universal Windows Platform*** la quale richiede lo scripting backend IL2CPP.

Tra le caratteristiche che hanno reso Unity un ambiente di sviluppo così diffuso è l'**Asset Store**, lo store esterno di Unity, tramite il quale è possibile acquistare o scaricare gratuitamente personaggi, oggetti, ambienti e altre funzionalità sviluppate da terze parti da integrare con la propria applicazione.

Per iniziare ad approcciarsi a questa tecnologia si è può far uso della pagina web "Unity Learn" in cui vi sono mini tutorial, con video corsi ed esempi [39].

Ad ogni modo, Unity da solo non basta per creare un'applicazione di realtà mista. Il motivo risiede nel fatto che non è stato creato a quello scopo. Per agevolare lo sviluppo, sono stati creati diversi toolkit che mettono a disposizione alcuni elementi di base per la MR che altrimenti dovrebbero essere impostati manualmente. MRTK è uno strumento che configura automaticamente l'ambiente per la realtà mista.

3.2 MRTK

Mixed Reality Toolkit (MRTK) è un progetto open source multipiattaforma sviluppato da Micorsoft, che fornisce una raccolta di strumenti e funzionalità che consentono di realizzare applicazioni di realtà mista in Unity. Sono presenti anche alcune

Figura 3.1: Logo [40]

scene d'esempio che illustrano le funzionalità di MRTK. Fornisce i *building blocks* cruciali per le esperienze di Mixed Reality come ad esempio il sistema di input, la consapevolezza spaziale, le interazioni e l'interfaccia utente.

MRTK può essere scaricato da GitHub e mette a disposizione i seguenti pacchetti: *Foundation*, *Extensions*, *Tools*, *Test utilities* e *Examples*. Nella documentazione di MRTK si può vedere la descrizione di ciascuno di essi e adesso verranno illustrati brevemente [41]. Il pacchetto Foundation è il più importante e contiene il codice che consente alle applicazioni di sfruttare funzionalità comuni tra piattaforme di realtà mista. La seguente immagine mostra in generale come è strutturato il pacchetto.

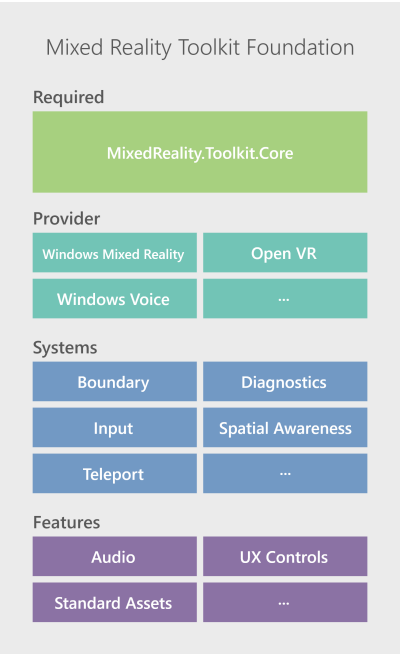


Figura 3.2: Mixed Reality Toolkit Foundation [42]

Il Mixed Reality Toolkit Foundation è quindi composto da:

- MRTK/Core: in cui vi sono le definizioni di interfacce, classi base e standard shader;
- MRTK/Core/Providers: fornitori di dati indipendenti dalla piattaforma in uso e comprende il supporto per il rilevamento delle mani, per la registrazione del movimento della testa, simulazione nell'editor dell'input della mano e dell'occhio, consapevolezza spaziale, dispositivi di input come joystick o mouse;

- MRTK/Providers: fornitori di dati specifici per la piattaforma in uso e comprende il supporto a LeapMotion, OpenVR, Oculus, UnityAR, WindowsMixedReality che include Microsoft HoloLens e visori immersivi;

- MRTK/SDK: include funzionalità sperimentali come shader, controlli dell'interfaccia utente e gestori di sistema. Comprende i profili predefiniti per i sistemi e i servizi Microsoft Mixed Reality Toolkit ed anche alcune risorse standard;

- MRTK/SceneSystemResources: comprende le risorse utilizzate dello Scene system;

- MRTK/Services: comprende i seguenti servizi:

  - BoudarySystem: fornisce il supporto per la visualizzazione dei confini della VR che definiscono l'area in cui l'utente può muoversi liberamente mentre indossa un visore;
  - CameraSystem: fornisce il supporto per la gestione e configurazione della telecamera di sistema;
  - DiagnosticSystem: supporta la visualizzazione della diagnostica di sistema dell'applicazione in uso;
  - InputSystem: supporta la gestione e l'accesso dell'input dell'utente;
  - SceneSystem: fornisce il supporto per applicazioni multi-scena;
  - SpatialAwarenessSystem: supporta la consapevolezza spaziale dell'ambiente in cui si trova l'utente;
  - TeleportSystem: fornisce il supporto per il teletrasporto in cui ci si può spostare tra esperienze diverse.

- MRTK/StandardAssets: comprende modelli, standard shader, materiali di base, texture e altre risorse standard per esperienza di realtà mista.

Il Mixed Reality Toolkit Extensions contiene invece dei servizi aggiuntivi che si vanno ad incorporare a quelli offerti da Foundation. Fornisce un servizio che aggiunge le articolazioni alle mani, semplifica la gestione della perdita di rilevamento sui dispositivi Microsoft HoloLens e semplifica la gestione dell'aggiunta di transizioni di scena.

Il Mixed Reality Toolkit Tools comprende gli strumenti che aiutano a facilitare il processo di creazione e distribuzione di applicazioni Universal Windows Platform, aiutano a gestire le dipendenze degli asset in un progetto e aiutano

nell'aggiornamento del codice che utilizza componenti MRTK deprecati.

Il Mixed Reality Toolkit Test utilities possiede dei metodi per facilitare la creazione di test in modalità play e permette di simulare l'input della mano.

Nel Mixed Reality Toolkit Examples vi sono le demo, ovvero delle semplici scene già create, delle demo sperimentali in cui vengono mostrate alcune caratteristiche sperimentali e le risorse comuni condivise in più scene.

MRTK possiede tre importanti proprietà: modularità, multipiattaforma e ottime prestazioni.

Si tratta di un framework di sviluppo creato per essere modulare, ovvero, non è obbligatorio importare tutto il toolkit nel progetto. Offre agli sviluppatori la possibilità di includere nell'applicazione solo le funzionalità necessarie. Tramite questo approccio le dimensioni del progetto si riescono a mantenere contenute e ciò ne semplifica la gestione.

Il toolkit include il supporto per più piattaforme. Anche se questo non significa che ogni singola piattaforma sia supportata, Microsoft ha fatto in modo che nessuna parte di codice del toolkit smetta di funzionare quando si scelgono altre piattaforme di destinazione per la build. L'affidabilità e l'estendibilità della progettazione modulare permettono alle app di supportare più piattaforme, ad esempio ARCore, ARKit e OpenVR. MRTK permette di astrarre dal dispositivo fisico in utilizzo e questo lo rende utile per creare delle applicazioni compatibili con diversi devices.

Dovendo lavorare con piattaforme mobili, è stato realizzato tenendo sempre in considerazione le prestazioni. Microsoft ha voluto assicurarsi che gli strumenti che ha messo a disposizione non creino difficoltà agli utenti.

Tramite questo strumento è possibile creare un'applicazione completa in breve tempo perché abilita la prototipazione rapida, con cui è possibile visualizzare immediatamente le modifiche apportate. La creazione rapida di prototipi è resa possibile grazie agli strumenti relativi all'interfaccia utente, all'esperienza utente e all'uso di prefabbricati.

MRTK supporta ufficialmente OpenXR e la versione di Unity 2020.3 LTS (Long Time Support), grazie all'ultima versione rilasciata. E' quindi possibile utilizzare tutti questi strumenti assieme per avere il massimo supporto allo sviluppo di applicazioni XR.

3.3 OpenXR

OpenXR è uno standard aperto royalty-free creato da **Khronos**, che fornisce un Application Program Interface (API) ad alte prestazioni per piattaforme e dispositivi di realtà aumentata (AR) e realtà virtuale (VR), noti collettivamente come eXtend reality (XR).

Figura 3.3: Logo [43]

Il Khronos Group, fondato nel 2000, è un consorzio focalizzato alla creazione di standard aperti per API libere da royalty per la realizzazione di media dinamici per un'ampia varietà di piattaforme e dispositivi.

Sono molti i membri che fanno parte di Khronos e hanno aderito alla creazione di OpenXR. L'immagine seguente mostra alcune delle aziende che hanno supportato pubblicamente OpenXR, ve ne sono molte altre.

Figura 3.4: Aziende che supportano pubblicamente OpenXR [44]

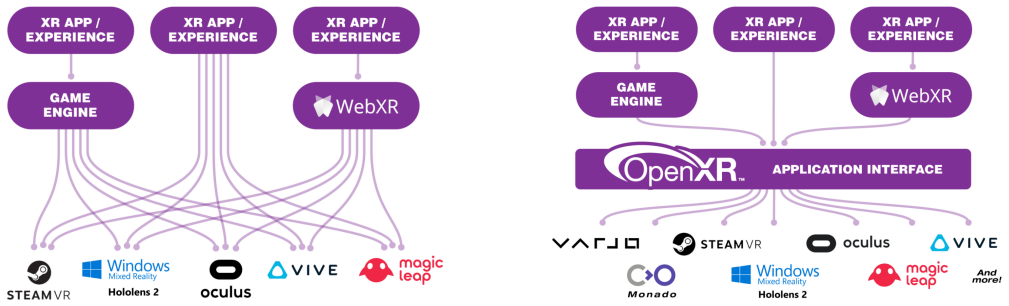
Perché è stato necessario riunirsi e collaborare per creare OpenXR? Il problema principale era dovuto alla cosiddetta frammentazione, ovvero ciascuna piattaforma aveva la propria interfaccia SDK (Software Development Kit) collegata all'hardware e ciascuna di queste era diversa dalle altre. Brent E. Insko (Intel) disse durante il SIGGRAPH [1] 2019: "se ci pensiamo un attimo, operazioni quali sapere la posizione della testa, conoscere dove sono le mani nel mondo reale, se e quali bottoni vengono premuti nel momento in cui si usa il controller, sapere quando le immagini dovranno essere mostrate nello schermo, queste sono tutte funzionalità comuni per le applicazioni di AR/VR". All'inizio c'era un ecosistema molto frammentato e questo creava numerosi problemi.

Innanzitutto aumentavano i costi degli sviluppatori in quanto dovevano scrivere diverse applicazioni che svolgevano la stessa identica funzione per ge-

[1] SIGGRAPH: abbreviazione di Special Interest Group on GRAPHics and Interactive Techniques ed è la conferenza sulla grafica computerizzata (CG) organizzata annualmente

stire le diverse piattaforme nei devices. In secondo luogo aumentavano molto i tempi e i costi di validazione perché spesso si avevano errori di compilazione e quindi doveva essere verificato tutto il codice per le diverse piattaforme.

Una volta riconosciuto il problema alla fine del 2016 e all'inizio del 2017 si formò il gruppo di lavoro di OpenXR in Khronos. L'obiettivo di OpenXR era quello di creare una singola interfaccia API comune a tutti. Consente quindi, di scrivere codice una volta sola, che è perciò portabile su un'ampia gamma di piattaforme hardware. Gli utenti finali ottengono le massime prestazioni indipendentemente da quale visore si stia usando. Nel grafico seguente è possibile vedere il problema della frammentazione prima dell'introduzione di OpenXR, in cui le applicazioni avevano bisogno di codice proprietario per ciascun dispositivo sul mercato. A destra, invece, si ha una visione dell'architettura ad alto livello dopo l'introduzione di OpenXR, in cui fornisce un accesso multipiattaforma e ad alte prestazioni in diversi dispositivi XR su più piattaforme.



(a) Prima di OpenXR (b) Dopo OpenXR

Figura 3.5: Situazione prima e dopo l'introduzione di OpenXR [45]

Dal rilascio di OpenXR è stato ampiamente utilizzato ed è utile perché permette alle applicazioni di funzionare su qualsiasi sistema che espone le API OpenXR e quindi rende possibile la portabilità. Nonostante tutto, si è ancora molto legati al tipo di dispositivo utilizzato, OpenXR cerca di fornire uno standard per risolvere il problema della frammentazione ma esistono, ad esempio, OpenXR per HoloLens2 e OpenXR per Magic Leap. Perciò non è possibile esportare direttamente un'applicazione senza apportare modifiche.

Capitolo 4

Sviluppo del progetto

Il sistema realizzato complessivamente è costituito da tre sottosistemi, i quali interagiscono tra di loro in modi diversi. E' stato realizzato un grafico che mostra i vari sottosistemi per mettere in evidenza come essi interagiscono fra di loro e quali tecnologie sono state applicate per ciascuno.

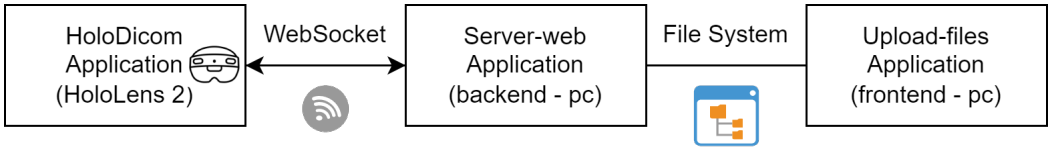


Figura 4.1: Sottosistemi del progetto

Ad alto livello i sottosistemi svolgono le seguenti funzionalità.

- `HoloDicom Application` è l'applicazione principale utilizzata direttamente dall'utente tramite il dispositivo HoloLens 2. Anch'esso composto da tre sottoparti. Si occupa della gestione del menù, della visualizzazione e manipolazione degli ologrammi e di creare la connessione per la comunicazione con il sottosistema `Server-web Application`.

- `Server-web Application` è stato creato allo scopo di effettuare il caricamento *run-time* dei modelli 3D per rispettare i requisiti richiesti in fase di analisi. Grazie ad una connessione è possibile lo scambio di messaggi tra le due applicazioni. `HoloDicom Application` effettua delle richieste che vengono soddisfatte dal `Server-web Application` il quale restituisce innanzitutto gli ultimi n file caricati (dove n è un numero reale maggiore a 0) e successivamente il modello che è stato chiesto dall'utente se ha cliccato un bottone. Le informazioni che servono vengono prese all'interno di una cartella presente nel file system in cui sono stati precedentemente salvati i file.

- `Upload-files Application` è un'interfaccia web realizzata per permettere il caricamento dei modelli 3D e i file DICOM che vengono salvati appunto in una specifica cartella, la stessa di cui si parlava precedentemente, che ha lo scopo di rendere più semplice il processo di caricamento per migliorare l'esperienza utente.

Il progetto completo si può trovare su `GitHub` ai seguenti repository.

- `https://github.com/GiuliaNardicchia/holodicom-application` (`HoloDicom Application`)

- `https://github.com/GiuliaNardicchia/web-application` (`Server-web Application` e `Upload-files Application`)

Le tecnologie che si è deciso di utilizzare per sviluppare il progetto lato *client*, come si è visto precedentemente nel Capitolo 3, sono innanzitutto l'ambiente di sviluppo Unity, in particolare la versione 2020.3 LTS. Il motivo per cui si è scelto di utilizzare la versione *Long Term Support* è perché vengono rilasciate molte versioni di Unity in breve tempo e quindi si preferisce usare quella che viene valutata dai creatori come la più stabile. Si è deciso di utilizzare il Mixed Reality Toolkit con il supporto ad OpenXR, per avere una base precostruita per realizzare l'applicazione per realtà mista e che allo stesso tempo potesse potenzialmente funzionare con diversi dispositivi. La versione di MRTK utilizzata è la più recente, ovvero la numero 2.7.3. Per la scrittura del codice in C# negli script si è usato Visual Studio 2019, utile anche e soprattutto per effettuare il *debugging* dell'applicazione. Per quanto riguarda invece l'applicazione lato *server* e la web app si è utilizzato Visual Studio Code con JavaScript e html come linguaggi.

4.1 HoloDicom Application

Per poter funzionare, l'applicazione ha bisogno di avere in input file e informazioni. Ad esempio, per visualizzare il menù, ha bisogno di sapere in input quanti bottoni deve creare. Per caricare il modello *runtime* ha bisogno che esista la cartella e che all'interno vi siano i file .obj. I file DICOM invece, dato che non è possibile il caricamento *runtime* per i motivi che verranno specificati in seguito, devono essere presenti nella cartella "StreamingAssets/DICOM/" di Unity. Il modello viene generato dai colleghi di biomedica ottenuto utilizzando un algoritmo di segmentazione ai file DICOM, i quali sono generati da un dispositivo diagnostico. In seguito alla segmentazione si ottiene un risultato poco vicino a quello che è in realtà. Per rendere l'organo in questione più

simile a quello originario, si deve effettuare un processo di *smoothing*, ovvero l'aggiunta di vertici e facce per dare l'idea di un oggetto dalla forma liscia, senza spigoli. Di conseguenza rende meno precisa la corrispondenza nella sovrapposizione del modello con le immagini DICOM e il tempo di caricamento risulta anche più lento.

4.1.1 Descrizione degli elementi della scena

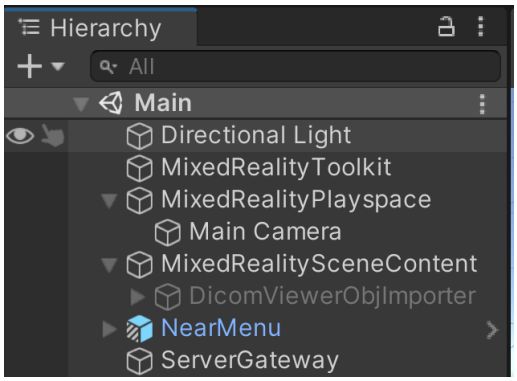


Figura 4.2: Elementi della scena in Unity

Da come si può vedere nella Figura 4.2, in questa applicazione gli elementi della scena sono:

- Le componenti importate tramite MRTK. `Directional Light` è la rappresentazione di una luce molto lontana che emette luce in una sola direzione. `MixedRealityToolkit` contiene tutte le funzionalità di cui si è parlato nel Capitolo 3.2, per la configurazione del profilo si è scelto di utilizzare `DefaultHololens2ConfigurationProfile` personalizzato. Il `MixedRealityPlayspace` contiene al suo interno `Main Camera`, nello specifico tramite una relazione genitore-figlio, con la quale si è in grado di mostrare il mondo all'utente.

- All'interno di `MixedRealitySceneContent`, che rappresenta il contenuto della scena, si trova l'oggetto `DicomViewerObjImporter` e si tratta dell'elemento principale dell'applicazione che tuttavia, dato che dev'essere creato in seguito, inizialmente è disabilitato e invisibile (come si può notare dal colore del nome).

- L'elemento `NearMenu` è la rappresentazione del menù da mostrare all'avvio dell'applicazione e gestisce il comportamento dei bottoni. Il colore

blu nel nome, nell'ambiente di sviluppo Unity, sta ad indicare che l'entità in questione è un *prefab*, ovvero un prefabbricato.

- L'oggetto `ServerGateway` è il componente che funge da server e interagisce con l'applicazione `Server-web Application` tramite scambio di messaggi grazie ad una connessione.

Nel seguente grafico è possibile visualizzare una rappresentazione del progetto tramite il diagramma delle classi.

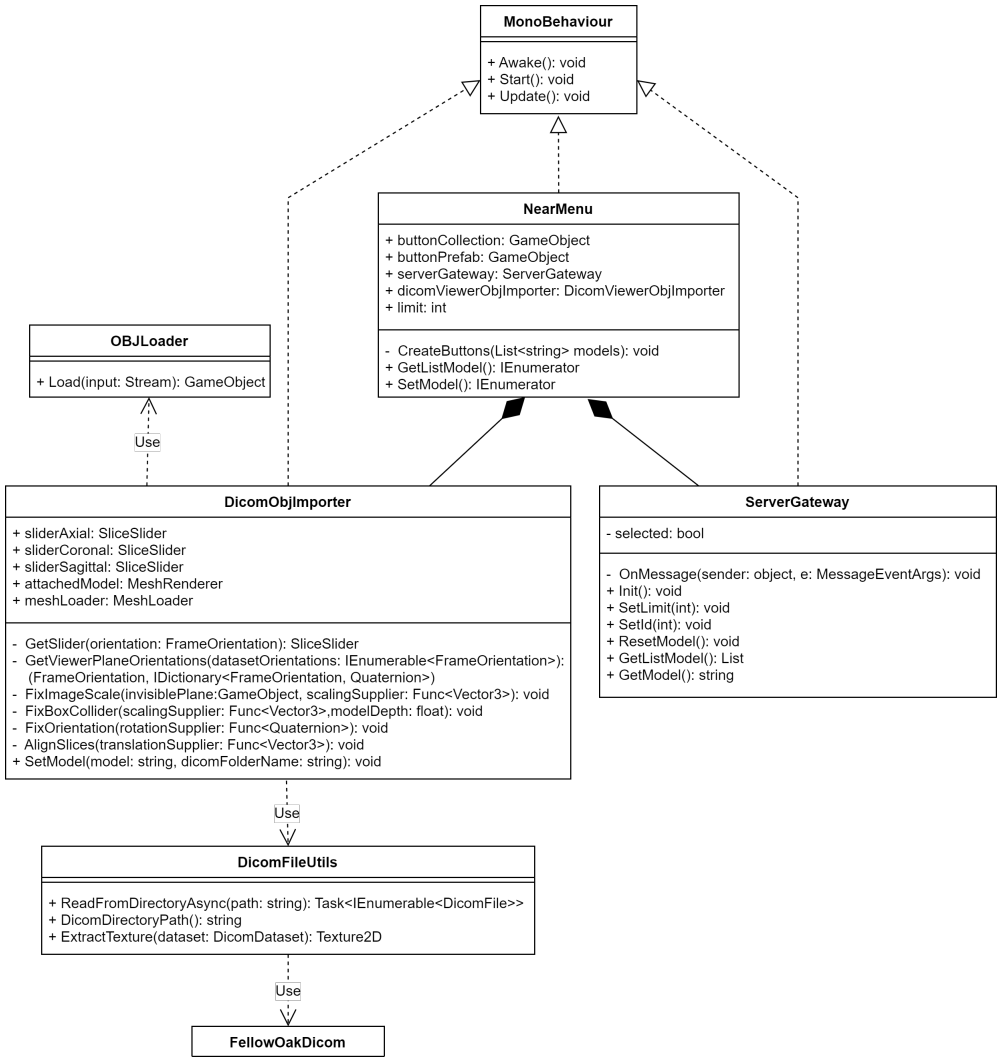


Figura 4.3: Diagramma delle classi

4.1.2 NearMenu

Per la realizzazione del menù è stato utilizzato il *prefab* contenuto nel pacchetto NearMenu appartenente a MRTK. Si tratta di un *asset* prefabbricato composto da tanti elementi, che funge da modello ed è utilizzabile per creare altre istanze dello stesso. Un NearMenu è un elemento di controllo UX che fornisce una raccolta di pulsanti. E' un oggetto mobile attorno alla posizione dell'utente ma è possibile bloccarlo o sbloccarlo, per permettere all'utente di averlo sempre con sé. Poiché è possibile afferrarlo e posizionarlo in punti specifici, non disturba l'interazione dell'utente con l'oggetto principale. In seguito, viene poi sistemata la rotazione e la posizione.

Figura 4.4: NearMenu di esempio nel formato 4x2 appartenente a MRTK [46]

Il NearMenu è stato realizzato con i seguenti componenti [47]:

- il *prefab* `PressableButtonHololens2`, che rappresenta un singolo bottone;
- `GridObjectCollection` che rappresenta una griglia in cui sono raccolti più bottoni;
- `ManipulatorHandler`, grazie al quale è possibile prendere e spostare il menù;
- `RadialViewSolver`, permette il comportamento *follow-me*.

Nell'immagine seguente è possibile vedere un esempio di come è strutturato un NearMenu e quali sono, in particolare, gli elementi grafici che lo compongono. Il primo componente evidenziato è il `ButtonPin` che rappresenta il bottone per bloccare/sbloccare il menù. Poco più in basso si può notare il componente

`Quad` in `Backplate` grazie al quale è possibile regolare la dimensione del pannello. La larghezza e l'altezza del backplate devono essere 0.032 * [Number of the buttons + 1]. Il secondo elemento evidenziato è il `ButtonCollection`, il quale rappresenta una collezione di bottoni, come si vede in figura ne include otto. `GrabVisualCue`, invece, è la componente grafica che permette la manipolazione del menù ed è composto dalle quattro direzioni (sinistra, destra, alto, basso).

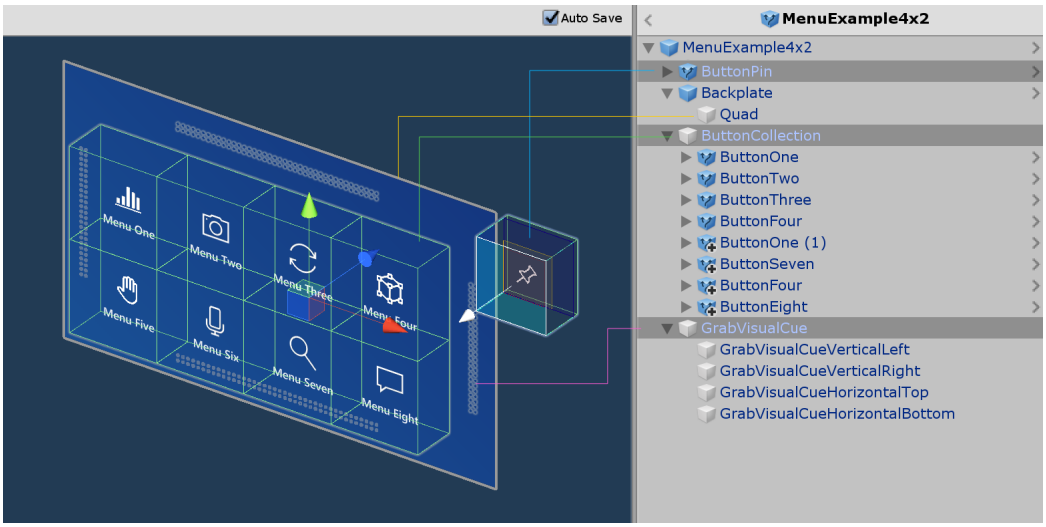


Figura 4.5: Struttura di un NearMenu [48]

Per gestire il comportamento del menù è stato creato uno script omonimo `NearMenu`. Trattandosi comunque di una componente grafica, la classe creata deve implementare la classe base `MonoBehaviour` la quale permette di assegnare lo script al `GameObject` nell'editor. Fornisce, inoltre, un'ampia raccolta di *event handler*, le quali permettono di eseguire codice in base a cosa accade durante il ciclo di vita del progetto. Ad esempio, definisce i metodi quali: *Awake*, *Start*, *OnEnable*, *OnDisable*, *Update*, *LateUpdate* e *FixedUpdate*. Ciascuna di queste funzioni di evento ha un ordine prestabilito.

Facendo riferimento ad un pattern architetturale molto utilizzato nella programmazione orientata agli oggetti, ovvero al pattern MVC (*Model View Controller*) in cui il model contiene i metodi per accedere ai dati utili all'applicazione, la view visualizza i dati contenuti nel model e si occupa dell'interazione con l'utente e il controller riceve i comandi dell'utente (in genere passati a lui tramite la view) e ha la facoltà di cambiare stato agli altri due componenti. Essendo riferito ad un elemento grafico, questo componente può fungere sia da View che da Controller. Possiede cinque campi pubblici e ciascuno ha la sua importanza.

Uno è chiamato `limit` e serve a impostare il numero massimo di modelli che si vuole far visualizzare nel menù. In altri due campi vengono salvati gli elementi grafici `ButtonCollection` e `ButtonPrefab` che sono di tipo `GameObject`. Quelli più importanti sono però `DicomViewerObjImporter` e `ServerGateway`.

Andando più nel dettaglio, il NearMenu inizializza il ServerGateway, imposta il numero di bottoni da visualizzare tramite il metodo `SetLimit` passando il valore `limit` preso in input e richiede una lista di nomi dei modelli da mostrare nel menù. Il `ButtonPrefab` serve a creare nuovi pulsanti all'occorrenza. Una volta che il NearMenu riceve il nome dei modelli che deve mostrare, per ciascuno viene generato appunto un nuovo bottone a partire dal prefab esistente e viene assegnato come figlio al `ButtonCollection`. Inoltre, quest'ultimo serve ad effettuare l'*upload* dell'interfaccia grafica per attuare le modifiche che altrimenti non verrebbero visualizzate.

Nel momento della creazione di ciascun bottone viene assegnato un *event listener*, ovvero un ascoltatore di eventi in modo tale che venga richiamata una determinata funzione alla pressione (*click*) del bottone. Si è associato un identificatore al bottone così da sapere quale viene eventualmente selezionato. Un oggetto di tipo *interactable* è in grado di percepire in input gli eventi ed essere *responsive*.

Quando l'utente clicca su un bottone, viene richiamata la funzione che recupera l'id del bottone selezionato (`SetId`) e parte la richiesta del modello al ServerGateway tramite una `StartCoroutine` che aspetta fin quando `GetModel` non è diverso da *null*. Prima, però viene anche effettuato il *reset* del modello in quanto, dalle specifiche richieste, l'applicazione deve rendere possibile il caricamento di più .obj durante la stessa sessione. Il NearMenu, una volta ottenuto il file del modello 3D, crea un'istanza dell'oggetto `DicomViewerObjImporter`, imposta il file e attiva l'oggetto.

Per effettuare le richieste al `ServerGateway` utilizza il metodo `StartCoroutine`. Una *coroutine* è una funzione che può essere messa in pausa e attendere che riprenda al verificarsi di una certa condizione. In questo caso è indispensabile richiamare una coroutine perché al NearMenu serve aspettare di avere le informazioni prima di procedere. Deve avere i nomi dei modelli per poterli mostrare nei bottoni e al click del pulsante deve aspettare di avere il file prima di far visualizzare il modello con il DICOM corrispondente.

4.1.3 ServerGateway

`ServerGateway` crea una connessione di rete tramite `WebSocket` all'indirizzo url e alla porta specificata. Questa classe può essere vista come il *Model* dell'applicazione, poiché possiede i metodi necessari ad ottenere i file e le informazioni che dovranno poi essere visualizzati. Possiede le funzioni per

impostare il numero massimo di modelli e l'id del modello selezionato passatogli dal NearMenu e i metodi che servono ad ottenere i nomi dei modelli e il file. Per importare `WebSocketSharp` [49] si è fatto uso di `NuGet`, un gestore di pacchetti che serve a consentire agli sviluppatori di condividere codice riutilizzabile. Viene fatto uso di una variabile booleana `selected` che viene definita `true` nel caso in cui sia stato selezionato il modello, `false` se al contrario non è successo. E' stato definito un metodo privato `OnMessage` per la ricezione di messaggi ed è in questo punto che viene effettuato il controllo del predicato `selected` allo scopo di distinguere il messaggio che viene ricevuto in seguito alle richieste.

4.1.4 DicomViewerObjImporter

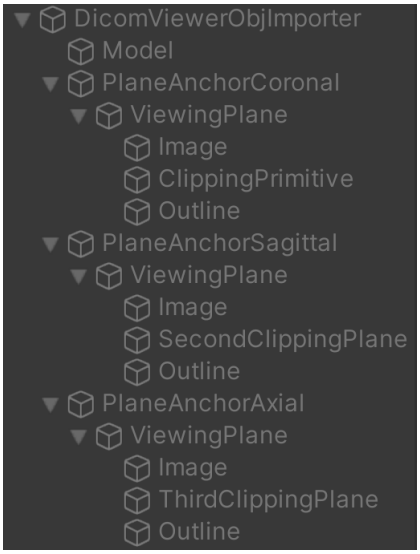


Figura 4.6: Elementi contenuti in `DicomViewerObjImporter`

Partendo sempre dagli elementi della scena che compongono `DicomViewerObjImporter` si può vedere come sia costituito da: `Model`, `PlaneAnchorCoronal`, `PlaneAnchorSagittal` e `PlaneAnchorAxial`. Il primo rappresenta l'elemento grafico nel quale viene caricata a *runtime* la *mesh* del modello 3D. Gli altri tre componenti, invece, simboleggiano i tre piani di taglio possibili per scorrere le immagini DICOM, come si è visto nel Capitolo 2.2.2, il sistema di coordinate anatomiche è costituito da tre piani: coronale, sagittale e assiale. Viene abilitato e mostrato uno dei piani in questione, in base alla tipologia di acquisizione delle immagini DICOM, se ad esempio viene effettuata una TAC (Tomografia Assiale Computerizzata), verrà abilitato e mostrato solo il piano assiale. E' anche possibile abilitarli

tutti e mostrare all'operatore l'organo da tre punti di vista diversi. Ciascun piano è composto dal genitore `ViewingPlane` e da tre elementi figli: `Image`, `ClippingPrimitive` e `Outline`. `ViewingPlane` rappresenta il *Bounding Box* dell'oggetto virtuale grazie al quale l'utente può effettuare la manipolazione: può spostarlo, ridimensionarlo e ruotarlo. `Image` raffigura la singola *slice* del file DICOM. `ClippingPrimitive` è il componente che permette il taglio sul modello. Possono assumere varie forme: piano, sfera e a forma di scatola, in questo caso si è scelto di utilizzare la forma *plane*. `ClippingPrimitive` dà la possibilità di specificare da quale lato della primitiva effettuare il taglio (intero o esterno) quando viene usato con gli *shader* MRTK, rappresentato dall'elemento `Outline`. Il framework MRTK mette a disposizione dello sviluppatore una moltitudine di *Standard Shader* [50], sistema in cui è possibile creare un materiale riflettente, trasparente, luminescente oppure opaco modificando in modo opportuno i parametri.

`DicomViewerObjImporter` si occupa, quindi, di tutta la gestione del caricamento del modello e dei file DICOM. Rappresenta il principale oggetto virtuale da visualizzare e che l'utente potrà manipolare. Si occupa inoltre di gestire la direzione dello *SliceSlider* a partire dall'orientazione dell'immagine, di allineare le *slices* e di sistemare la dimensione dell'immagine nello spazio. E' in grado di fare questo grazie all'utilizzo di due strumenti: Fo-dicom e Runtime OBJ Importer.

Fo-dicom

Il termine Fo-dicom deriva da **Fellow Oak dicom** ed è un toolkit semplice da usare per leggere e manipolare immagini DICOM e file. Si tratta di codice open-source derivante dall'Asset Store di Unity. E' stato importato attraverso `NuGet`, un gestore di pacchetti che serve a consentire agli sviluppatori di condividere codice riutilizzabile. E' la principale tecnologia usata per far visualizzare i file DICOM all'interno dell'ambiente Unity.

Runtime OBJ Importer

Runtime OBJ Importer è un pacchetto contenente codice open-source trovato sempre nell'Asset Store di Unity. Le sue caratteristiche sono che è veloce, semplice da usare, funziona sia nell'editor importando gli oggetti direttamente sia effettuando il caricamento runtime. Non supporta solo il caricamento della mesh di file con estensione .obj ma offre anche la possibilità di caricare .mtl (per importare il materiale) e caricamento della texture.

Una nota negativa del Runtime OBJ Importer è il fatto che per caricare la *mesh* all'interno del dispositivo HoloLens 2 è necessario apportare delle modifiche nelle impostazioni di Unity. Nelle impostazioni del progetto è necessario cliccare su `"Graphics"` e aggiungere `"Standard (Specular Setup)"` per includere lo *shader*. Se non viene abilitato, è possibile vederlo solo nell'ambiente di sviluppo Unity quando ci si trova in modalità `Play`. Queste modifiche non erano necessarie se il caricamento del modello avveniva direttamente in Unity come per i file DICOM.

Implementazione

Facendo riferimento al diagramma delle classi della Figura 4.3, si possono vedere i campi pubblici che possiede la classe `DicomObjImporter` associata al `GameObject` omonimo. I primi tre sono gli elementi sono gli `SliceSlider` che corrispondono ai tre assi di taglio. Il campo `attachedModel` è l'elemento grafico `Model` che viene passato in input dall'editor, mentre `meshLoader` rappresenta `DicomObjImporter` stesso.

Una volta che ottiene il modello passato tramite `SetModel`, viene creato un `MemoryStream`, necessario all'`OBJLoader` per la creazione dell'oggetto `loadedObj` per effettuare in seguito il caricamento. Dopo aver creato l'oggetto è essenziale recuperare da esso il `MeshFilter` e creare un *array* per combinare più *mesh* tra loro, chiamato `CombineInstance`. L'esigenza di combinare le *mesh* è nata dal fatto che quando si è provato ad effettuare il caricamento di un modello composto da due elementi come ad esempio i reni policistici, ne veniva caricato uno solo tra i due, problema che non si verificava con un solo elemento come l'atrio di un cuore. La *mesh* viene poi assegnata al modello grafico `attachedModel`. Per evitare che venga aggiunto due volte lo stesso oggetto virtuale visualizzabile uno con la *mesh* e uno come oggetto senza elaborazioni, quest'ultimo viene distrutto. Infine, si assegna la rotazione e la posizione del modello originale. Sono state effettuate molteplici prove su `MATLAB` per verificare che i modelli fossero stati creati effettivamente nella posizione corretta e con le dimensioni giuste prima di caricarle nell'applicazione.

Nel seguente listato è presente il codice relativo al caricamento del modello 3D.

```
var textStream = new
    MemoryStream(Encoding.UTF8.GetBytes(model.ToString()));
var loadedObj = new OBJLoader().Load(textStream);

MeshFilter[] meshFilters =
    loadedObj.GetComponentsInChildren<MeshFilter>();
```

```
CombineInstance[] combine = new CombineInstance[meshFilters.Length];

int i = 0;
while (i < meshFilters.Length)
{
    combine[i].mesh = meshFilters[i].sharedMesh;
    combine[i].transform =
        meshFilters[i].transform.localToWorldMatrix;
    meshFilters[i].gameObject.SetActive(false);
    i++;
}
attachedModel.GetComponent<MeshFilter>().mesh = new Mesh();
attachedModel.GetComponent<MeshFilter>().mesh.CombineMeshes(combine);
var modelMesh = attachedModel.GetComponent<MeshFilter>();

Destroy(loadedObj);

attachedModel.transform.rotation = modelMesh.transform.rotation;
attachedModel.transform.localPosition = modelMesh.transform.position;
```

Listato 4.1: Caricamento runtime .obj

I file di input relativi delle immagini DICOM vengono invece caricati nell'applicazione all'interno dello `StreamingAsset` nella cartella `DICOM`. Contrariamente al modello non viene importato runtime. Il motivo principale di questa scelta implementativa è che in seguito a varie prove non è stato possibile caricarlo runtime poiché i file erano molto numerosi e la latenza era molto sentita, richiedendo un tempo troppo lungo. Potrebbero esserci delle strategie per evitare questo problema che si rimandano al paragrafo degli sviluppi futuri. Le immagini DICOM vengono importate facendo uso di `DicomFileUtils` il quale a sua volta utilizza `FellowOakDicom`. Per ciascun file viene verificato che l'header non presenti anomalie e vengono ordinati in base al nome. Spesso il nome dei file è numerato, per questo è importante che siano in ordine in base all'orientazione delle immagini nello spazio di tipo LPS, perché altrimenti si avrebbe una visione falsata. Sono state effettuate parecchie prove in cui sono stati inseriti i file DICOM in ordine inverso rispetto all'orientazione dell'organo e in effetti si è visto che l'oggetto virtuale veniva caricato al contrario. Questo problema non si verifica con lo strumento che usano i colleghi di Ingegneria biomedica perché è stato costruito appositamente e per questo motivo vengono riconosciuti automaticamente. Una volta importati i file si ottiene l'oggetto `DicomFile` per avere tutte le informazioni che solitamente i trovano nell'header. Si è impostato il piano da abilitare per il taglio grazie alle informazioni sull'orientazione ottenute dalle proprietà di `DicomFile` e poi si è estratta

la texture e assegnata all'`imageRenderer` recuperato dal `viewingPlane`. In seguito sono state fatte delle operazioni di aggiustamento della scala dell'immagine, della dimensione del `BoxCollider` e infine sono state allineate tutte le *slices*.

Per rispettare uno degli obiettivi del progetto richiesti in fase di analisi dei requisiti è stato creato anche il bordo del modello. E' stato realizzato sfruttando il fatto che la `mesh` del modello all'interno è vuota e il meccanismo di visualizzazione del `viewingPlane` creando un ulteriore piano, ovvero l'`invisiblePlane`. Il meccanismo consiste nel creare appunto un piano invisibile che è posto di fronte a tutto, composto di un materiale che non taglia il modello. Poi, si deve porre il piano in cui si vedono le *slice* qualche millimetro più avanti composto di un materiale che invece taglia il modello in modo tale da rendere visibile il bordo.

Il `DicomViewerObjImporter` è stato il frutto del lavoro, del miglioramento e dell'aggiunta di funzionalità di un progetto precedentemente iniziato in fase di tirocinio da un collega di Ingegneria e Scienze Informatiche.

4.2 Server-web Application

Per la realizzazione di questa applicazione è stato fatto uso di `Node.js`, che è stato progettato per creare applicazioni di rete scalabili. Si tratta di un sistema runtime open source multipiattaforma orientato agli eventi per l'esecuzione di codice `JavaScript`. In particolare si è utilizzato `Express.js`, un server framework per applicazioni web per Node.js, ormai divenuto uno standard de facto.

Come si può vedere dal codice seguente, inizialmente vengono richieste tutte le risorse e inizializzate tutte le costanti utili a creare una connessione `WebSocketServer` per poter comunicare con il `ServerGateway`.

```
const express = require('express');
const http = require('http');
const WebSocket = require('ws');
const app = express();
app.use(express.static("files"));

const server = http.createServer(app);
const wss = new WebSocket.Server({ server: server });
```

Listato 4.2: Creazione dell'oggetto `WebSocketServer` per la connessione.

Successivamente il server si deve mettere in ascolto su una porta, che bisogna specificare. Si deve fare attenzione ad impostare una porta opportuna e dev'essere uguale a quella che si inserisce nel `SereverGateway` all'interno dell'applicazione `HoloDicomApplication`. In questo caso, la porta che si è impostata è quella contenuta nella variabile d'ambiente o se non è disponibile, la numero 8080.

```
server.on('request', (request, res) => {
  console.log("[REQUEST FROM] %o:%o", res.socket.remoteAddress,
      res.socket.remotePort);
});

server.listen(process.env.PORT || 8080, () => {
  console.log(`[SERVER STARTED] ${server.address().port}`);
});
```

Listato 4.3: Il server in attesa di una richiesta e si mette in ascolto sulla porta specificata.

Una volta stabilita la connessione con un *client*, il ServerGateway si mette in attesa di ricevere un messaggio. E' stato stabilito un protocollo di comunicazione tra le due applicazioni, ovvero si è definito un carattere che funge da prefisso per indicare limit (`l`) e l'id (`i`) seguiti da un segno di punteggiatura quale i doppi punti (`:`) e il messaggio effettivo. Una volta ricevuto un messaggio viene effettuato un controllo per verificare a quale dei due si riferisce e agisce di conseguenza.

Quando riceve il messaggio in cui si richiedono gli ultimi `limit` modelli caricati all'interno della cartella `files` effettua un ordinamento degli stessi in base agli ultimi che sono stati modificati e spedisce solo i più recenti. Quando invece, viene richiesto il modello, avendo salvato in un una variabile la lista dei nomi ordinati, viene recuperato il file da spedire tramite l'id.

```
wss.clients.forEach(function each(client) {
     if (client.readyState === WebSocket.OPEN) {

       var msg = data.toString().substring(0, 2);
       var value = data.toString().replace(msg, "");

       if (msg.normalize() === "n") {
         client.send("n");
       }
       if (msg.normalize() === "l:".normalize()) {

         models = getRecentFiles();
```

```
        models = models.slice(Math.max(models.length - value, 0));

        for (var i=0; i<models.length; i++) {
          client.send(models[i]);
        }
      }
      if ((msg.normalize() === "i:".normalize()) && (models !=
          null)) {
        var object = models[Number(value)];
        const content = fs.readFileSync(dir + "/" + object,
            {encoding:'utf8'});
        client.send(content);
      }
    }
  });
```

Listato 4.4: Server in attesa di un messaggio.

4.3 Upload-files Application

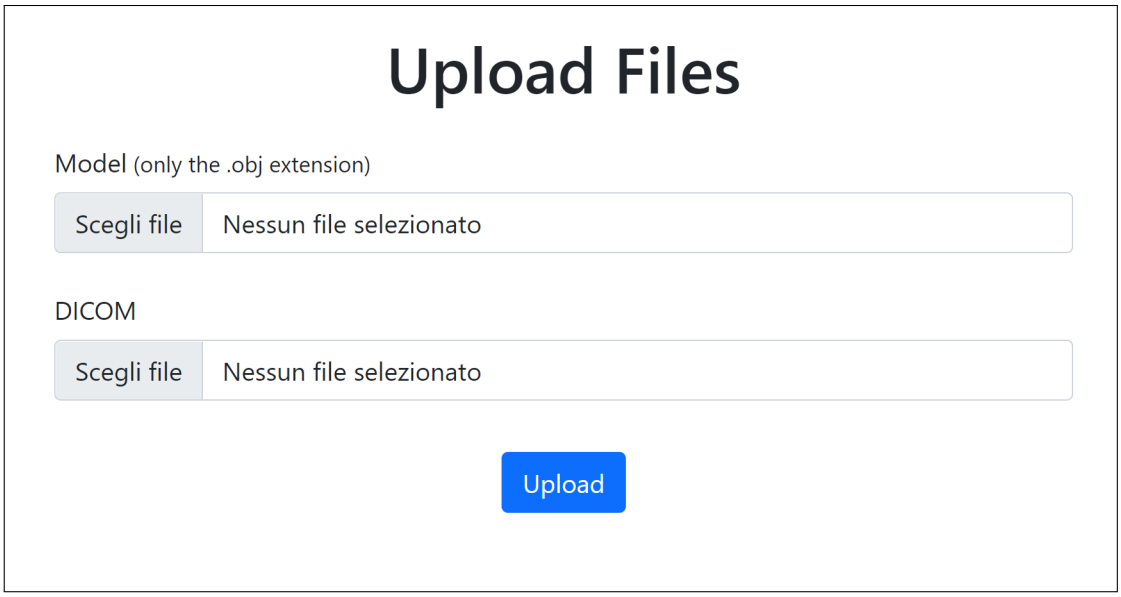


Figura 4.7: Pagina web Upload-Files

`Upload-files Application` è una interfaccia web creata sempre con Node.js ed Express.js. Nel file `index.js`, quando viene effettuata una richiesta

`GET` viene caricato il file `upload_files`. La pagina web è stata creata utilizzando `html` e per aggiungere qualche elemento di stile si è fatto uso di `Bootstrap`. E' possibile aggiungere un solo modello 3D avente estensione .obj e molteplici file DICOM. E' stato impostato l'inserimento di qualcosa in entrambi obbligatoriamente, altrimenti non viene permessa l'operazione di salvataggio. Una volta effettuato l'upload, avviene il salvataggio dei file all'interno della cartella `files`. I file DICOM vengono salvati in una nuova cartella che ha lo stesso nome del modello. Se i file sono stati caricati correttamente la pagina viene reindirizzata ad un altro url in cui si vede la scritta "*Upload file success*".

Capitolo 5

Validazione e sviluppi futuri

5.1 Validazione

Per validazione o convalida del software si intende quel processo di controllo di una determinata applicazione per valutare se essa risulta essere conforme agli usi previsti ed in particolare alle esigenze dell'utente [51]. E' un processo che si dovrebbe effettuare a prodotto finito, tuttavia è possibile realizzarlo anche durante il processo di sviluppo del prototipo. Ad esempio, si può validare l'applicazione se rispecchia i requisiti e gli obiettivi preposti in fase di analisi.

Al completamento del prototipo sono state effettuate tutte le prove del caso per verificare che tutti gli scenari d'uso fossero effettivamente funzionali e se l'applicazione fosse utile se venisse utilizzata dai medici e dagli operatori sanitari per svolgere le attività di diagnosi.

Per effettuare la validazione è stato utilizzato uno dei modelli con i corrispondenti file DICOM forniti dai colleghi di Ingegneria Biomedica, ovvero gli organi dei reni policistici. Di seguito varranno mostrati alcuni casi d'uso definiti in fase di analisi dei requisiti specificati nel Paragrafo 2.1.1.

Gli *screenshot* effettuati per mostrare il funzionamento dell'applicazione derivano dall'ambiente di sviluppo Unity nella modalità Play, non dal dispositivo HoloLens 2, benché il risultato sia identico, nelle immagini non è presente il mondo reale.

La prima cosa che si vede all'avvio dell'applicazione è il NearMenu, dalla figura è possibile vedere che è composto da 6 bottoni (numero specificato in input). Ciascun bottone presenta il nome di un file avente estensione .obj.

Figura 5.1: Menù con 6 bottoni

Il primo caso d'uso mostrato era relativo all'interazione con il menù, tra le operazioni che si possono fare si ha la possibilità di spostarlo, bloccarlo/-sbloccarlo per permettere al menù di seguire la posizione dell'utente o fissarlo in un punto e cliccare i bottoni. Nell'immagine 5.2 si è scelto il modello corrispondente ai reni policistici.

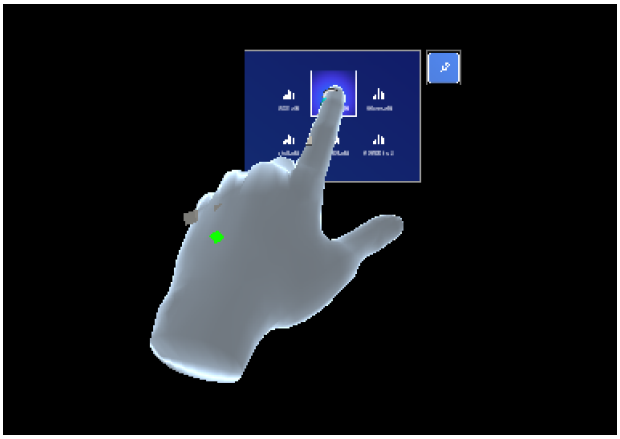


Figura 5.2: Click del bottone

Una volta che si clicca su un pulsante tattile, viene effettuato il caricamento del file DICOM unitamente al modello 3D. Nella figura successiva è possibile

vedere la prima *slice* del file DICOM, il modello si trova dietro. Si può notare in tutte le figure che seguiranno il fatto che il menù rimarrà sempre a disposizione dell'utente.

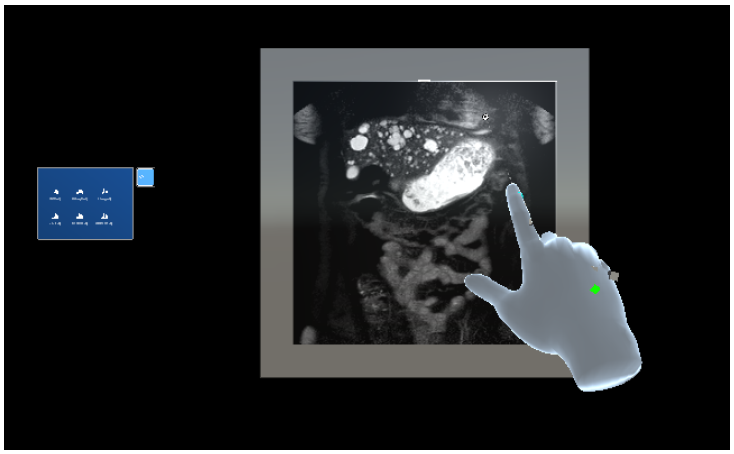


Figura 5.3: Caricamento file DICOM e modello 3D

Il secondo caso d'uso mostrato era relativo all'interazione con il modello selezionato. Anch'esso è manipolabile, si può ridimensionare: ingrandendolo o rimpicciolendolo e ruotandolo dai 4 lati del cubo.

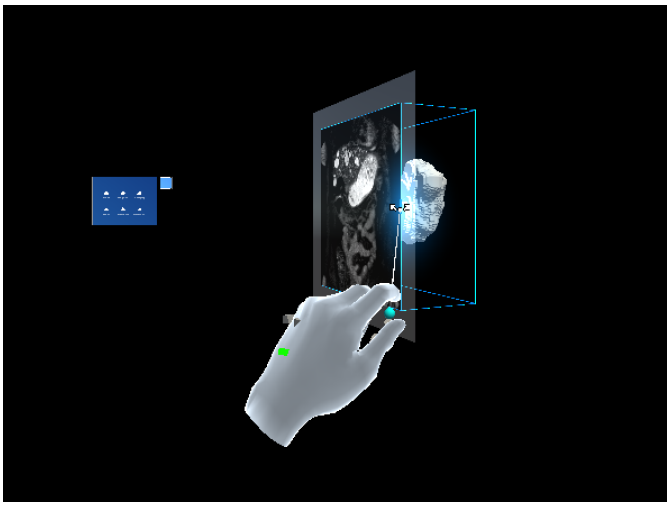


Figura 5.4: Esempio di rotazione dell'ologramma

E' possibile ruotare il cubo completamente e visualizzare il modello 3D che si trova sul retro.

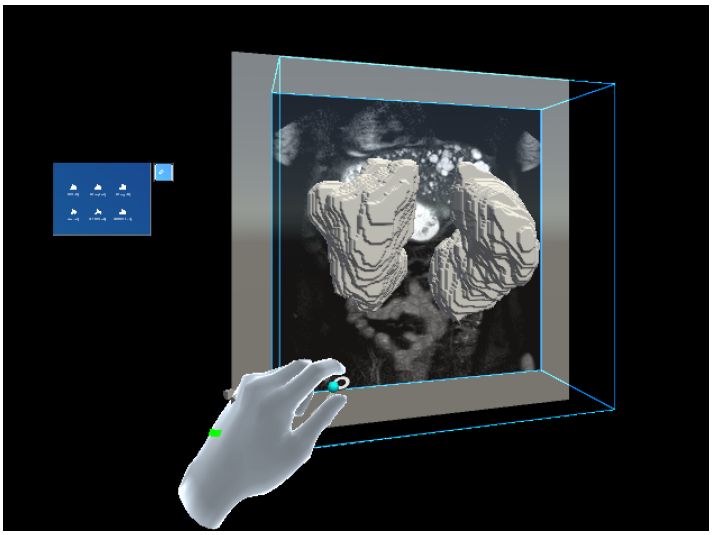


Figura 5.5: Modello 3D

Un'altra operazione che è possibile svolgere per interagire con il modello è tramite lo scorrimento delle immagini dei file DICOM. In questo esempio il piano di taglio utilizzato è quello coronale. Nell'immagine in Figura 5.6 si vede un taglio successivo a quello mostrato precedentemente in sovrapposizione al modello, il quale è possibile vederlo grazie al bordo bianco applicato.

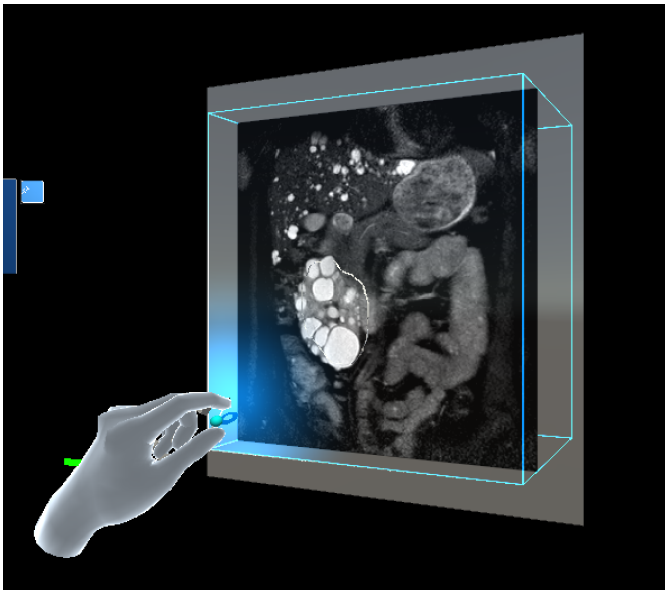


Figura 5.6: Scorrimento delle immagini

Continuando poi a scorrere le immagini si arriva ad un punto in cui si vedono entrambi i reni policistici.

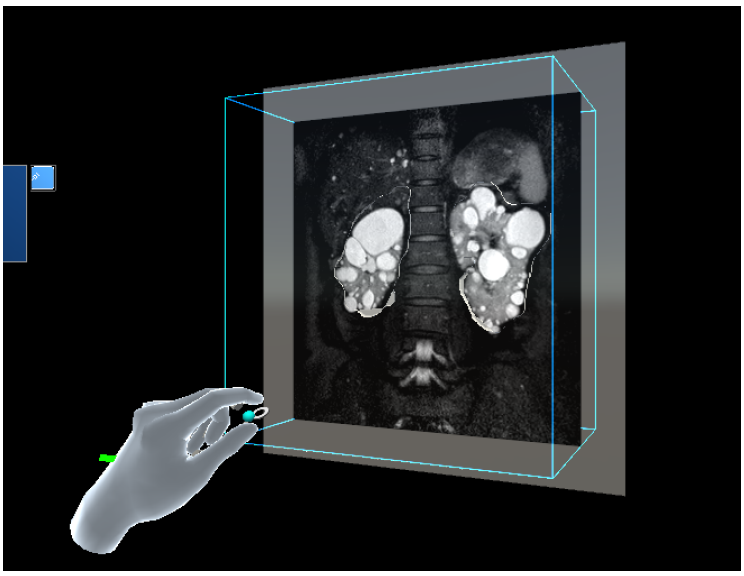


Figura 5.7: Caricamento file DICOM

5.2 Problematiche principali

In questo paragrafo si mostreranno le varie problematiche riscontrate durante la progettazione e implementazione del prototipo.

5.2.1 Elaborazione delle informazioni

Quando vengono generati i modelli a partire dai file DICOM vengono effettuate delle elaborazioni e delle conversioni da .stl a .obj che modificano delle informazioni. Da parte dei colleghi di Ingegneria Biomedica devono essere svolte delle ulteriori elaborazioni, in quanto il modello deve essere orientato, posizionato e scalato in base all'orientazione delle immagini nello spazio dei file DICOM, altrimenti il modello potrebbe non essere visualizzato correttamente. Allo stesso modo, anche le immagini DICOM devono essere orientate e il nome per ciascun file numerato in ordine rispetto al sistema di coordinate DICOM, altrimenti la ricostruzione in ologramma verrà visualizzata in ordine sbagliato o inverso (a seconda di come siano ordinati).

5.2.2 Ottimizzazione del tempo di caricamento dei modelli

I modelli ottenuti dalle immagini DICOM hanno un numero molto elevato di vertici, il che limita fortemente le performance su HoloLens 2. In particolare se viene effettuato lo *smoothing* per rendere più morbida la forma del modello, la latenza del caricamento è molto sentita, dato che si vengono a creare un numero maggiore di vertici e facce che compongono l'oggetto 3D.

5.2.3 Caricamento di file DICOM runtime

Nella versione corrente i modelli possono essere caricati tramite una richiesta al WebSocket, in formato .obj (formato Wavefront). Il caricamento dei file DICOM, invece, rimane limitato alla lettura della cartella contenuta in "Streaming Assets/DICOM/", questo perché dalle prove svolte, il caricamento di file molto lunghi e numerosi comporta il blocco dell'applicazione per molto tempo. Questo è dovuto soprattutto dal fatto che l'applicazione non può andare avanti nel processo se non ha completato prima l'importazione di tutti i file.

5.2.4 Corrispondenza tra immagini e modelli

Nella versione attuale del progetto l'allineamento dell'immagine viene effettuato centrando la `Bounding Box` del modello sull'asse di scorrimento del piano dell'immagine. Grazie a questo metodo si ottengono risultati abbastanza buoni ma per avere una corrispondenza perfetta andrebbero usati dei metodi più sofisticati.

5.3 Sviluppi futuri

Dopo aver parlato delle problematiche, adesso verranno presentate alcune idee per effettuare dei miglioramenti o nuove funzionalità per sviluppi futuri.

5.3.1 Caricamento di file DICOM runtime

Per quanto riguarda il problema del caricamento a run-time dei file DICOM, una soluzione potenzialmente non bloccante potrebbe essere quella di utilizzare un metodo di compressione di file: prendere la cartella, convertirla in zip e successivamente inviarla. Una volta arrivati, per ottenere i file originali usare un metodo di decompressione.

5.3.2 Metodo di visualizzazione del taglio nei modelli

Un altro aspetto che si potrebbe approfondire, è quello di permettere il taglio del modello da più angolazioni in aggiunta ai tre assi standard. Ad esempio effettuare anche un taglio in obliquo darebbe più spazio al medico per visualizzare alcune parti del volume che non riuscirebbe a notare solo da un punto di vista.

5.3.3 Modelli dinamici

Un possibile sviluppo futuro di questa applicazione potrebbe essere quella di caricare anche modelli dinamici oltre a quelli statici. In particolare sarebbe utile visualizzare il cuore di un paziente che batte e con che regolarità per effettuare diagnosi e migliorare il trattamento delle disfunzioni cardiache. Bisogna però prestare attenzione e domandarsi se sia effettivamente funzionale e utile allo scopo, dato che potrebbero verificarsi varie complicanze. Tra cui la latenza, il tempo di attesa potrebbe essere troppo grande per vedere i movimenti in real-time e il caricamento dell'interfaccia grafica potrebbe ritardare e quindi vedersi male.

5.3.4 Integrare informazioni funzionali

Una delle funzionalità che sarebbe più utile sviluppare in futuro è dare la possibilità di integrare delle informazioni funzionali per ciascun vertice del modello. Queste informazioni sono utili a mettere in risalto quelle che sono le particolarità del volume. In particolare se in un organo è presente un tumore, grazie a queste ulteriori informazioni è possibile colorare quella zona di rosso ad esempio evidenziandola dal resto, dato che è in scala di grigi.

5.3.5 Condivisione di informazioni

Come visto anche in letteratura, sono presenti molte applicazioni che svolgono anche la funzionalità con cui è possibile la condivisione di informazioni tra più dispositivi. Sarebbe vantaggioso aggiungere questa funzionalità per condividere file DICOM e modelli 3D oppure pareri e opinioni riguardo alla diagnosi tra più medici posizionati nello stesso ambiente o anche in ambienti diversi.

5.3.6 Ulteriori validazioni

Il risultato ottenuto è stato il frutto della collaborazione tra il nostro dipartimento e i colleghi di Ingegneria Biomedica. Si sono susseguiti numerosi

incontri per scambiarsi informazioni e opinioni. La validazione del prototipo è stata effettuata personalmente da tutto il team formatosi. Sarebbe proficuo se la provassero anche i medici dell'AUSL di Cesena con la quale il nostro dipartimento collabora per avere un riscontro sull'effettiva funzionalità del prototipo e se è utile allo scopo.

Conclusioni

Dal lavoro di tesi e dalle analisi effettuate è emerso che il prototipo sviluppato è uno strumento utile ai fini diagnostici, per permettere al medico o all'operatore sanitario di migliorare l'operato del proprio lavoro.

Non solo aiuta il medico a migliorare la precisione della diagnosi a velocizzare l'attività sanitaria e di conseguenza le successive terapie ma ha un vantaggio anche per il paziente che ottiene le cure in tempi più brevi e maggiormente efficaci.

E' stato creato per supportare il settore medico così da rendere più semplice e facile la visualizzazione dei file DICOM. Tutto ciò è reso possibile tramite la visualizzazione delle immagini con il corrispondente modello 3D in sovrapposizione ed è inoltre possibile manipolare gli ologrammi così da spostarli, ingrandirli o rimpicciolirli e ruotarli nell'ambiente.

Per questo tipo di progetto è necessaria una collaborazione con diversi dipartimenti, nel nostro caso è stato con Ingegneria Biomedica ma date le ampie possibilità di miglioramento e crescita sarebbe utile ampliare il team di ricerca per ottenere più funzionalità.

Nonostante il progetto sia stato valutato come utile, si tratta però di un prototipo che, come detto in precedenza, va migliorato e sarebbe maggiormente utile se venisse integrato con l'aggiunta di ulteriori funzionalità.

Ringraziamenti

Vorrei ringraziare tutte le persone che mi sono state accanto e che mi hanno accompagnato durante tutto il mio percorso.

Innanzitutto, vorrei ringraziare il Professor Alessandro Ricci e l'Ingegner Angelo Croatti, rispettivamente relatore e correlatore di questa tesi, perché mi hanno seguito per tutto il periodo di tirocinio e durante la stesura della tesi. Ringrazio sentitamente perché mi hanno trasmesso la passione e l'entusiasmo per questo mondo che mi era sconosciuto. Inoltre vorrei ringraziare la Professoressa Cristiana Corsi e l'Ingegner Claudio Fabbri ed i loro collaboratori per la costante cooperazione tramite scambio di informazioni, utili al completamento dell'elaborato.

Un ringraziamento speciale va alla mia famiglia che da sempre mi ha sostenuta in questo cammino. Grazie mamma, papà e Sara che non hanno mai smesso di credere in me e mi hanno sempre incoraggiato ed esortato a dare il meglio. Grazie per essere al mio fianco, per aver riposto in me la vostra fiducia e per avermi appoggiato sin dall'inizio.

Vorrei ringraziare anche tutti i miei amici e compagni di università che hanno condiviso con me le ansie, le preoccupazioni ma soprattutto le gioie e i successi ottenuti. Insieme ci siamo aiutati a superare tutte le difficoltà incontrate.

Infine, vorrei dedicare questo grande traguardo a me stessa perché non è stato facile e non mi sono mai data per vinta.

Grazie infinitamente a tutti.

Bibliografia

[1] J. Peddie, *Augmented reality: Where we will all live*. Springer, 2017.

[2] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented reality: A class of displays on the reality-virtuality continuum," *Telemanipulator and Telepresence Technologies*, vol. 2351, 01 1994.

[3] "Reality-virtuality continuum image." https://www.researchgate.net/profile/Jennifer-Beckmann/publication/326176523/figure/fig1/AS:647283586646020@1531335841308/Reality-Virtuality-Continuum-Milgram-Takemura-Utsumi-Kishino-1994.png.

[4] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoper. Virtual Environ.*, vol. 6, p. 355–385, aug 1997.

[5] "Spatial mapping." https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-mapping.

[6] E. Costanza, A. M. Kunz, and M. Fjeld, "Mixed reality: A survey," in *Human Machine Interaction*, 2009.

[7] "HoloLens 2 image." https://inclusify.de/wp-content/uploads/2021/10/HLS19_HoloLens2_leftFacing_tilt_RGB_2K_whiteBG.png.

[8] "HoloLens 2." https://www.microsoft.com/en-us/hololens/hardware.

[9] "Six Degree Of Freedom." https://en.wikipedia.org/wiki/Six_degrees_of_freedom.

[10] "Magic Leap One image." https://images.ctfassets.net/bl73eiperqoo/2ASQpSFFxY1ze9RxKS0tAI/cb7608510c024f5f56f4598355f81c3f/ML-Family-Shot-Blank-Template2_1.png.

[11] "Magic Leap One." https://www.magicleap.com/en-us/magic-leap-1.

[12] "Example image of head-up display." https://www.tuningblog.eu/wp-content/uploads/2019/06/Head-up-Display-nachr%C3%BCsten-Tuning.jpg.

[13] "Example image of hand-held display." https://d2sr9p9v571tfz.cloudfront.net/wp-content/uploads/2020/07/realta-aumentata.jpg.

[14] "Start designing and prototyping." https://docs.microsoft.com/en-us/windows/mixed-reality/design/design.

[15] "What is a hologram?." https://docs.microsoft.com/en-us/windows/mixed-reality/discover/hologram.

[16] M. Pell and T. Parisi, *Envisioning holograms: Design breakthrough experiences for mixed reality*. Apress, 2017.

[17] "Holographic frame." https://docs.microsoft.com/en-us/windows/mixed-reality/design/holographic-frame.

[18] "Optimal zone image." https://docs.microsoft.com/en-us/windows/mixed-reality/design/images/distanceguiderendering-950px.png.

[19] "Rendering (computer graphics)." https://en.wikipedia.org/wiki/Rendering_(computer_graphics).

[20] "Holographic rendering overview." https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/rendering-overview.

[21] "Coordinate systems." https://docs.microsoft.com/en-us/windows/mixed-reality/design/coordinate-systems.

[22] "Sistemi di coordinate." https://docs.microsoft.com/en-us/windows/uwp/graphics-concepts/images/leftrght.png.

[23] S. Ong, *Beginning Windows Mixed Reality Programming: For HoloLens and Mixed Reality Headsets*. Apress, 2017.

[24] "Spatial mapping image." https://docs.unity3d.com/es/2018.4/uploads/Main/spatialmapping_example.jpg.

[25] "Introducing instinctual interactions." https://docs.microsoft.com/en-us/windows/mixed-reality/design/interaction-fundamentals.

[26] "Shared experiences in mixed reality." https://docs.microsoft.com/en-us/windows/mixed-reality/design/shared-experiences-in-mixed-reality#get-started-building-shared-experiences.

[27] B. Spiegel, G. Fuller, M. Lopez, T. Dupuy, B. Noah, A. Howard, M. Albert, V. Tashjian, R. Lam, J. Ahn, F. Dailey, B. T. Rosen, M. Vrahas, M. Little, J. Garlich, E. Dzubur, W. IsHak, and I. Danovitch, "Virtual reality for management of pain in hospitalized patients: A randomized comparative effectiveness trial," *PLOS ONE*, vol. 14, pp. 1–15, 08 2019.

[28] "Witapp." https://www.witapp.it/.

[29] "Witapp lancia un innovativo software per la diagnostica." https://www.hlstampa.com/witapp-lancia-un-innovativo-software-per-la-diagnostica/.

[30] "Verima image." https://simzine.it/wp-content/uploads/2021/10/Medico-ologramma-1024x576.jpg.

[31] "Witapp image." https://www.hlstampa.com/wp-content/uploads/2019/03/Foto-Witapp_7.jpg.

[32] "Medivis." https://www.microsoft.com/en-us/hololens/industry-healthcare.

[33] "Medivis image." https://img-prod-cms-rt-microsoft-com.akamaized.net/cms/api/am/imageFileData/RE4FJ3O?ver=7b56&q=90&m=6&h=450&w=800&b=%23FFFFFFFF&l=f&o=t.

[34] O. S. Pianykh, *Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide*. Springer-Verlag, Berlin Heidelberg, 2012.

[35] "Understanding coordinate systems and DICOM." https://theaisummer.com/medical-image-coordinates/.

[36] "Human anatomy planes image." https://textimgs.s3.amazonaws.com/boundless-anatomy-and-physiology/human-anatomy-planes.svg.

[37] "STL (STereoLithography) File Format." https://en.wikipedia.org/wiki/STL_(file_format).

[38] "Install the tools." https://docs.microsoft.com/en-us/windows/mixed-reality/develop/install-the-tools?tabs=unity.

[39] "Welcome to Unity Learn." https://learn.unity.com/.

[40] "MRTK logo image." https://repository-images.githubusercontent.com/50605426/ce574819-6bf5-44c5-a174-f22f20ee0ce9.

[41] "MRTK packages." https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/packages/mrtk-packages?view=mrtkunity-2021-05.

[42] "MRTK Foundation image." https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/images/input/mrtk_package_foundation.png?view=mrtkunity-2021-05.

[43] "OpenXR logo image." https://docs.microsoft.com/en-us/windows/mixed-reality/develop/native/images/openxr.png.

[44] "OpenXR Ecosystem Update." https://www.khronos.org/assets/uploads/apis/OpenXR-EcoSystem-Update_Jul20.pdf.

[45] "Khronos group - OpenXR." https://www.khronos.org/openxr/.

[46] "Near Menu image." https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/images/near-menu/mrtk_ux_nearmenu.png?view=mrtkunity-2021-05.

[47] "Near Menu." https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/ux-building-blocks/near-menu?view=mrtkunity-2021-05.

[48] "Near Menu structure image." https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/images/near-menu/mrtk_ux_nearmenu_structure.png?view=mrtkunity-2021-05.

[49] "WebSocketSharp - GitHub." https://github.com/sta/websocket-sharp.

[50] "Standard Shader MRTK." https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/rendering/mrtk-standard-shader?view=mrtkunity-2021-05.

[51] "Validazione del software nella ISO 13485 : 2016." https://www.infoqual.it/la-validazione-del-software-nella-iso-13485-2016/.