POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Informatica

Tesi di Laurea Magistrale

Progettazione e Sviluppo di Ambienti di Realtà Virtuale e Aumentata Multiutente per l'Entertainment

Relatori
Prof. Andrea Sanna
Ing. Federico Manuri
firma dei relatori

. . . . . .
. . . . . .

Candidato
Damiano Oriti

firma del candidato

. . . . . .

Anno Accademico 2018-2019

Sommario

Con *realtà aumentata* (AR) si intende quello scenario in cui alla realtà vera sono sovrapposti oggetti virtuali, che aggiungono qualcosa al contesto reale in cui sono inseriti (da cui il termine *aumentata*). Esempi di AR sono il popolare videogioco Pokemon GO uscito nel 2016, e gli occhiali Google Glass prodotti dall'omonima compagnia e indirizzati al settore industriale e professionale.

Si parla, invece, di *realtà virtuale* (VR), quando gli oggetti virtuali costituiscono un mondo a sé e nascondono *interamente* il mondo vero. La VR è particolarmente affascinante nell'ambito dell'intrattenimento, per la sua capacità di immergere completamente la persona dentro il mondo virtuale. Non è un caso che la recente rinascita della realtà virtuale, che già aveva avuto grande risonanza negli anni '90, sia legata al mondo dei videogiochi e dell'intrattenimento in generale; si pensi, ad esempio, al visore per console da gioco PlayStation VR©, o all'Oculus Rift per PC.

Lo sviluppo tecnologico degli ultimi anni nei settori della realtà aumentata e della realtà virtuale, unito alla riduzione dei costi, ha riacceso l'interesse verso queste tecnologie, la cui storia è iniziata più di 60 anni fa. Inizialmente relegati a settori d'élite quali la ricerca universitaria, l'industria bellica, e l'industria aerospaziale, l'AR e il VR hanno raggiunto oggi una maturità che li rende non molto dissimili da come la fantascienza li dipingeva solo alcuni decenni prima.

I campi di applicazione sono molteplici, dalla medicina all'educazione, dall'industra al marketing, passando per l'intrattenimento, nella forma di videogiochi, spettacoli teatrali, o film. All'interno di una sala operatoria, la realtà aumentata promette di liberare lo spazio occupato dagli ingombranti monitor, fornendo le informazioni di cui il chirurgo e gli infermieri hanno bisogno direttamente di fronte a loro, in forma virtuale; inoltre, la realtà aumentata permetterebbe anche di visualizzare immagini tridimensionali come quelle ottenute tramite TAC, ma proiettate direttamente sul paziente.

In ambito militare, la realtà virtuale può sostituire l'addestramento sul campo delle truppe, eliminando, da una parte, i costi connessi all'uso di mezzi e strumentazione bellica, e riducendo, dall'altra, il rischio di subire incidenti gravi. Casi in cui la VR rappresenta una valida alternativa alle pratiche più tradizionali includono l'addestramento dei piloti tramite simulatori di volo e l'addestramento del personale medico. Inoltre, la tecnologia della realtà aumentata può costituire un'arma senza precedenti anche nel contesto di un conflitto reale, fornendo al soldato informazioni che possono essere determinanti per la riuscita della missione.

Anche i settori della cultura, del turismo e dell'educazione hanno accolto con entusiasmo l'avvento di queste nuove tecnologie, incoraggiati dalle promesse di miglioramento, se non anche di rivoluzione, nel modo in cui interagiamo, lavoriamo, ci divertiamo, e impariamo. L'idea per cui *un'immagine valga più di mille parole* può spiegare l'entusiamo in questo nuovo modo di vivere alcune esperienze; si pensi, ad esempio, alla possibilità di

vedere un dinosauro di milioni di anni fa prendere vita in un museo.

Sebbene queste tecnologie rivoluzioneranno, probabilmente, diversi aspetti della nostra vita, siamo ancora lontani dall'adozione di massa nelle scuole, negli ospedali, nelle fabbriche, e tanto più nelle nostre case. Inoltre, il mercato della realtà mista (concetto che include realtà aumentata e realtà virtuale) è frammentato in due segmenti, quello formato da chi possiede un dispositivo AR, e quello formato da chi possiede un dispositivo per VR, che non si intersecano, se non in minima parte.

Ad oggi, non esiste un pacchetto di strumenti software ben integrati destinato allo sviluppo di applicazioni AR e VR, in grado di supportare modalità multiutente e le caratteristiche peculiari dei dispositivi per realtà aumentata e realtà virtuale. Il lavoro svolto nell'ambito della presente tesi è volto a colmare tale vuoto, attraverso lo sviluppo di un *framework* che possa costituire un punto di partenza per applicazioni di realtà aumentata e virtuale multiutente destinate all'intrattenimento, e non solo.

Il framework, la cui prima versione supporta il visore AR HoloLens e visori VR come l'Oculus Rift, ha tutte le funzionalità essenziali per la creazione di applicazioni multiutente per realtà mista: sincronizzazione delle entità, gestione del tracciamento e dell'allineamento degli oggetti virtuali con il mondo reale, supporto a diversi sistemi di input. Inoltre, esso è stato progettato per funzionare al meglio nel contesto delle applicazioni per AR e VR; ad esempio, favorendo la fluidità generale piuttosto che la qualità grafica.

Al fine di mostrare le funzionalità del framework, è stato sviluppato un videogioco multigiocatore, in cui i partecipanti devono cooperare per raggiungere degli obiettivi comuni. Ambientato nel Dipartimento di Automatica e Informatica del Politecnico di Torino, il videogioco richiede che i giocatori distruggano dei mostri disseminati all'interno di alcuni locali del dipartimento. Grazie a un'area tracciata di circa 200 m^2 e con le potenzialità di supportare aree ancora più estese, questo videogioco è anche uno dei pochi esempi di applicazioni per HoloLens pensate per aree di grandi dimensioni.

Nell'ultima parte della tesi, sono stati condotti dei test mirati a valutare vari aspetti del framework e del videogioco sviluppato, tramite questionari standard, nonché misurazioni quantitative, finalizzate a valutare gli aspetti oggettivi dei sistemi testati. Durante i test, sono stati impiegati i dispositivi per cui il framework è stato progettato, cioè l'HoloLens prodotto da Microsoft e l'Oculus Rift prodotto da Facebook.

Per quanto riguarda l'applicazione, gli aspetti analizzati includono la game experience, la social presence e l'usabilità; invece, per quanto riguarda il framework, sono stati analizzati aspetti quali la perdita del tracciamento e la fluidità dell'applicazione, nel contesto delle applicazioni di rete.

Nel primo capitolo è introdotta la realtà mista, seguita da una discussione sullo sviluppo di applicazioni per AR e VR. Il secondo capitolo è dedicato allo stato dell'arte della realtà aumentata e virtuale; si parte dalla definizione formale, quindi segue una breve rassegna sulla storia di queste tecnologie, e, infine, sono discusse le principali applicazioni in vari campi, fra i quali la medicina, il militare, l'educazione e la cultura. I capitoli 3 e 4

presentano il lavoro svolto nell'ambito della presente tesi; in particolare, il terzo contiene un'esaustiva descrizione del framework, a partire dai requisiti di progetto, continuando con i moduli e la loro funzione, mentre il quarto è incentrato sull'applicazione realizzata utilizzando il framework. Il quinto capitolo presenta il caso d'uso e la discussione dei test condotti per valutare gli aspetti principali del framework e dell'applicazione. L'ultimo capitolo presenta le conclusioni finali, riportando, inoltre, alcuni possibili sviluppi futuri, finalizzati, principalmente, a migliorare la base esistente.

Ringraziamenti

Vorrei ringraziare il Prof. Andrea Sanna e l'Ing. Federico Manuri, relatori della presente tesi di laurea magistrale, per avermi guidato, con i loro preziosi consigli, nella realizzazione di questo lavoro, dalla definizione del progetto fino alla stesura dell'elaborato.

Un caloroso ringraziamento va al Dott. Francesco De Pace, per avermi seguito e supportato con grande pazienza in questi mesi di intenso lavoro.

Ringrazio, infine, tutta la mia famiglia e gli amici, per il sostegno datomi in questi duri anni di università, che mi hanno permesso di crescere sia umanamente che professionalmente.

Indice

Elenco delle tabelle

Elenco delle figure

Capitolo 1

Introduzione

Nel primo capitolo della presente tesi viene introdotta in breve la tecnologia della *realtà mista*, seguita da una breve analisi sulla creazione di applicazioni per realtà virtuale (VR, dall'inglese *Virtual Reality*) e aumentata (AR, dall'inglese *Augmented Reality*); infine, si proverà a giustificare la scelta di sviluppare un *framework* finalizzato ad agevolare lo sviluppo di applicazioni AR/VR.

1.1 Realtà mista

L'espressione realtà mista (MR, dall'inglese *Mixed Reality*) include tutto lo spettro di tecnologie che vanno dalla realtà aumentata alla realtà virtuale (si veda la figura 1.1.). La realtà è detta mista perché il mondo reale è arricchito da oggetti virtuali, spesso chiamati ologrammi nella terminologia della realtà aumentata. Il rapporto tra il reale e il virtuale determina la regione dello spettro della MR in cui ci troviamo.

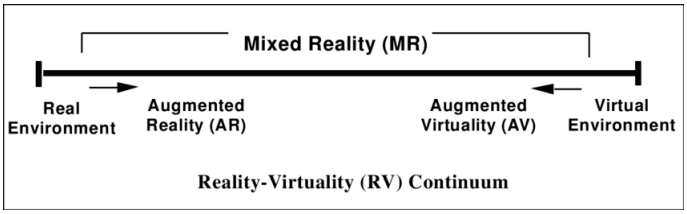

Figura 1.1. Rappresentazione del *Reality-Virtuality continuum*. Figura tratta da [23].

1.1.1 Realtà aumentata

Se l'esperienza di MR avviene nel mondo reale, in cui pochi oggetti virtuali sono sovrapposti al reale, si parla di realtà aumentata; l'aggettivo *aumentata* sta ad indicare proprio l'aggiunta di qualcosa che non c'è, senza nascondere l'esistente, che invece è arricchito con

nuove informazioni. Le possibilità non si riducono alla sola visualizzazione di qualcosa che prima non c'era; infatti, la tecnologia oggi esistente permette l'interazione con gli ologrammi, spalancando le porte a innumerevoli applicazioni pratiche che possono migliorare, e trasformare, la società come la conosciamo oggi.

I settori applicativi sono molteplici e includono, fra gli altri, l'istruzione [2], la medicina [33], l'industria [26] e *cultural heritage* [3].

Figura 1.2. Esempio di applicazione AR nel campo della medicina. Immagine tratta da [41].

1.1.2 Realtà virtuale

Viceversa, quando il mondo reale è completamente sostituito dal mondo virtuale si parla di realtà virtuale, intendendo che il virtuale diventa il luogo in cui ci si trova e la realtà *vera* non è più visibile, nascosta dietro il dispositivo VR.

La caratteristica principale della VR è l'immersività, cioè il sentirsi completamente catturati dal mondo virtuale, fino a dimenticarsi di quello reale. Quando l'esperienza è così simile al reale da ingannare i sensi, in special modo la vista, che è il nostro senso più sviluppato, il fruitore può sperimentare quella sensazione che prende il nome di *presenza*. Con questo termine si intende che il fruitore si sente trasportato nel mondo virtuale, e assume quei comportamenti che assumerebbe se si trovasse in una realtà uguale a quella ricreata al computer. Un esempio di ciò è l'esperimento (la stanza dell'esperimento è mostrata in figura 1.3) condotto dallo sviluppatore americano Valve, in cui a delle persone, le quali indossavano un visore VR, veniva chiesto di spostarsi da un cubo virtuale sul quale si trovavano e camminare nel vuoto. Pur sapendo che nel mondo vero c'era il pavimento tutto attorno al cubo virtuale, molti non riuscivano a fare il passo in avanti richiesto, avvertendo in alcuni casi un forte senso di vertigini.

Figura 1.3. La famosa stanza per i prototipi VR di Valve. Foto tratta da [32].

La VR non è solo contemplazione passiva di un mondo costruito al computer, ma è anche nuovi modi di interazione con gli oggetti virtuali. Per le persone nate negli ultimi cinquanta o sessanta anni, abituate ad usare un PC, o anche un semplice telecomando per cambiare il canale del televisore, l'interazione con la tecnologia ha sempre significato dover premere tasti o premere uno schermo; al più, per gli appassionati di videogiochi, oltre al tasto c'è la levetta o un grilletto. Comunque, l'interazione non è mai quella *meccanica* e analogica, che la vita di tutti i giorni ci insegna. Per spostarci, dobbiamo muovere le gambe in modo sincronizzato e mantenere l'equilibro per non finire a terra; per contrasto, in un videogioco, ci si muove tenendo premuto un tasto, o inclinando una levetta nella direzione verso cui vogliamo andare.

Uno degli ingredienti fondamentali della VR (ma anche della AR) è la possibilità di tracciare la testa della persona immersa nel mondo virtuale. Questa, in realtà, non è solo una possibilità, ma è anche una necessità, perché permette al fruitore di sentirsi trasportato in un altro mondo; inoltre, come si vedrà nel capitolo 2, tracciare la testa è fondamentale per evitare l'insorgere di sintomi legati a nausea.

Disponendo di una tecnologia capace di tracciare la testa, si può applicare la stessa tecnologia per tracciare le mani. Uno dei primi prodotti commerciali per VR erano dei guanti tracciati, chiamati DataGlove (si veda la figura 1.4), che permettevano di manipolare gli oggetti virtuali in modo naturale; ad esempio, si poteva afferrare una pallina da tennis virtuale semplicemente chiudendo la mano. Oggi, in quella che è una vera e propria rinascita del fenomeno VR dal periodo d'oro degli anni ottanta e novanta in cui sembrava che la VR dovesse diventare un prodotto di massa, il focus è passato dai guanti ai *motion controller*, dei dispositivi tracciati che possono simulare vari oggetti, come una pistola o una racchetta. Nelle applicazioni VR moderne, i motion controller permettono di

interagire, in un modo quasi naturale, con gli oggetti virtuali; ad esempio, si può azionare una leva virtuale afferrandone l'estremità e tirandola.

Figura 1.4. I DataGlove prodotti da VPL nel 1985.

1.2 Sviluppo di applicazioni per AR e VR

In questa sezione sono trattati i paradigmi di sviluppo delle applicazioni per AR e VR; in particolare, si vedranno gli aspetti relativi al *rendering*[1] degli oggetti virtuali, al processamento degli input e all'interazione, e alla presentazione delle immagini finali sullo schermo.

Prima di vedere gli aspetti specifici della AR e quelli specifici della VR, si vedrà la struttura di una generica applicazione per MR.

Un'applicazione per MR è un'applicazione grafica interattiva in tempo reale:

1. Grafica perché renderizza modelli tridimensionali (si veda la figura 1.5), composti da vertici, poligoni e texture.
2. Interattiva perché l'utente può fornire degli input all'applicazione, che risultano in modificazioni dello stato della stessa.
3. Tempo reale perché l'applicazione è aggiornata a una frequenza sufficientemente alta[2], al punto che le animazioni appaiono continue all'utilizzatore.

Il cuore dell'applicazione è il *main loop*, cioè un ciclo di aggiornamento infinito[3] in cui sono acquisiti e processati gli input dell'utente, viene aggiornato lo stato degli oggetti virtuali, e infine viene generato il frame[4] dalla GPU[5].

Altre operazioni importanti svolte dall'applicazione nel main loop sono la riproduzione dei suoni, i calcoli relativi alla fisica e il compositing dei frame.

[1] Termine inglese che si riferisce al processo di generazione dell'immagine a partire dalle primitive grafiche

[2] Maggiore o uguale a 60 Hz

[3] L'applicazione esce dal ciclo quando l'utente chiude l'applicazione

[4] L'immagine che viene inviata allo schermo e visualizzata

[5] Graphics Processing Unit; è il processore grafico dedicato

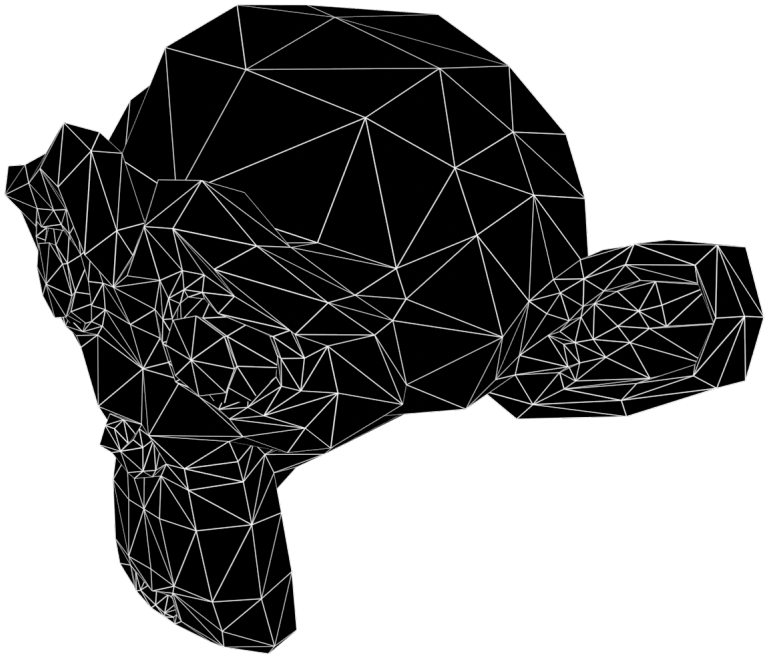

Figura 1.5. Modello 3D realizzato in Blender.

Realtà aumentata

Sviluppare applicazioni per realtà aumentata è sotto molti aspetti più complicato rispetto ad applicazioni tradizionali per desktop, smartphone o console da gioco, perché è imperativo tenere conto di tanti dettagli che possono fare la differenza tra un'applicazione utilizzabile e un'applicazione inutilizzabile.

Al contrario della VR, il mercato della realtà aumentata si muove verso dispositivi *standalone*[6], principalmente perché le caratteristiche della tecnologia la rendono particolarmente adatta ad applicazioni in cui essere connessi ad un cavo o non è possibile o rappresenterebbe un impedimento troppo grande. Un esempio è la manutenzione di macchinari tramite assistenza a distanza; se il dispositivo AR deve essere connesso a un desktop, questo dovrebbe essere vicino al macchinario su cui si devono effettuare le operazioni di manutenzione, e ciò non è sempre possibile.

Per la ragione appena esposta, quasi tutti i dispositivi AR oggi in commercio sono

[6]Un dispositivo che non richiede altri dispositivi per funzionare

standalone, con tanti vantaggi, ma anche alcuni svantaggi, non sempre trascurabili. Infatti, essendo di dimensioni ridotte, questi dispositivi non possono ospitare grosse unità di processamento, né le batterie che le dovrebbero alimentare; questo significa che la potenza computazionale disponibile per l'applicazione è molto limitata anche rispetto ad un desktop medio. Fortunatamente però, la caratteristica principale della realtà aumentata è che il numero di ologrammi da processare è spesso molto basso, quindi il carico sull'unità di processamento non è eccessivo.

Ciononostante, lo sviluppatore deve stare molto attento a non appesantire l'applicazione con modelli troppo dettagliati e texture a risoluzione elevata, in quanto un frame-rate[7] basso o incostante non è tollerabile, principalmente per due ragioni:

1. Rende più difficoltoso seguire con lo sguardo gli oggetti virtuali ed interagire in modo efficace con essi, rendendo l'applicazione inutilizzabile.

2. Rompe l'illusione che gli ologrammi facciano parte del mondo reale.

Ad oggi, i dispositivi per AR più diffusi sul globo sono gli smartphone (un'applicazione per AR è mostrata in figura 1.6), infatti quasi ogni persona possiede almeno uno smartphone personale. Sebbene gli smartphone siano dispositivi capaci di eseguire applicazioni di AR, le specifiche tecniche medie sono solo sufficienti per esperienze non troppo complesse dal punto di vista della fedeltà visiva.

Figura 1.6. Esempio di applicazione AR per smartphone. Foto tratta da [13].

Il limite tecnico non è tanto, o non solo, l'unità di processamento, quanto le capacità del sistema di tracciamento. Lo smartphone, infatti, non è nato come dispositivo per AR, e la soluzione comunemente adottata per il tracciamento, che fa uso di marker, non è molto robusta.

[7] Numero di frame generati al secondo

Anche dispositivi nati unicamente per l'AR, come l'HoloLens di Microsoft, sebbene vantino tecnologie e soluzioni più avanzate, funzionano solo in condizioni ideali e in ambiente controllato.

Lo sviluppatore deve tenere conto delle limitazioni delle soluzioni oggi disponibili, ad esempio progettando l'applicazione in modo che possa gestire situazioni di perdita del tracciamento[8], e sia in grado di recuperare quando il sistema di tracciamento è nuovamente attivo.

Un altro aspetto su cui concentrare l'attenzione è l'interazione con gli oggetti virtuali, che deve essere basata su paradigmi nuovi, che sfruttino la particolare interfaccia che il dispositivo scelto per lo sviluppo offre. Ad esempio l'HoloLens è un sistema multimodale, supportando diverse modalità di input, gesti e comandi vocali.

Realtà virtuale

Le applicazioni immersive[9] richiedono di essere attentamente studiate per trarre vantaggio dal particolare dispositivo per il quale sono pensate. Infatti, non tutte le soluzioni commerciali e non oggi disponibili offrono le stesse funzionalità e caratteristiche tecniche, quindi un'applicazione che funziona su uno specifico prodotto potrebbe non funzionare allo stesso modo su un altro prodotto. In particolare, i sistemi di tracciamento più comuni, basati su tecnologie ottiche, hanno caratteristiche diverse in termini di dimensione del volume in cui l'utente può muoversi liberamente, e persino nella possibilità di tracciare a 360°.

Un altro aspetto su cui lo sviluppatore deve concentrare la propria attenzione è la varietà di motion controller disponibili sul mercato, diversi per la qualità del tracciamento, numero e tipologia di tasti, forma ed ergonomia. Spesso lo sviluppatore è costretto a scegliere tra la necessità di supportare il più ampio spettro di dispositivi, per non tagliar fuori una parte dell'utenza, che è ancora una piccolissima percentuale del valore potenziale [22], o il supportare tutte le funzionalità avanzate offerte da uno specifico dispositivo.

Sebbene il rinato interesse verso la realtà virtuale degli ultimi anni sia stato avviato dall'**Oculus Rift**, un visore VR *tethered*[10], oggi sono disponibili sugli scaffali dei negozi sistemi VR standalone, che soffrono degli stessi problemi, precedentemente discussi, dei sistemi standalone per realtà aumentata, quali la scarsa potenza computazionale.

Lo sviluppatore si trova, quindi, a dover dosare attentamente il numero di vertici che l'applicazione deve gestire, al fine di non incorrere in cali di frame-rate, che hanno, invero, effetti ben peggiori rispetto al caso dell'AR; infatti, non solo il mondo virtuale non appare più solido e indistinguibile dal mondo reale, ma anche l'utente potrebbe manifestare i sintomi quali nausea, giramenti di testa, ecc. [37].

[8] Con perdita di tracciamento si intende che il dispositivo non sa più dove si trova

[9] Ci si riferisce alla realtà virtuale, in quanto l'immersività ne è la caratteristica principale

[10] Connesso con un cavo a un elemento esterno, come un PC

1.3 Motivazioni per lo sviluppo di un framework

Il mercato delle tecnologie della realtà mista è ancora piccolo e frammentato, non solo tra AR e VR, ma anche all'interno della stessa categoria. Negli ultimi anni, i maggiori *player* che operano nel settore della MR, tra cui Microsoft, Valve e Nvidia, per citarne alcuni, hanno fatto un notevole sforzo per semplificare la vita agli sviluppatori, migliorando gli strumenti software[11] disponibili, e soprattutto promuovendo degli standard che permettano di supportare dispositivi diversi senza dover scrivere del codice specifico per un dato dispositivo. Si pensi, ad esempio, all'iniziativa portata avanti dal **Khronos Group**[12] per un nuovo standard aperto indirizzato a AR e VR, *OpenXR*.

Inoltre, nella maggior parte dei casi, un'applicazione nasce con una sola piattaforma in mente, e, in quei rari casi in cui l'applicazione supporta sia AR che VR, spesso l'esperienza è asimmetrica, nel senso che la modalità per realtà aumentata sarà differente, anche in modo significativo, da quella pensata per realtà virtuale.

Il questo contesto il gruppo di ricerca guidato dal Prof. Andrea Sanna ha concepito l'idea di un framework che permettesse di semplificare il processo di realizzazione delle applicazioni indirizzate alla realtà mista.

Il framework nasce con il fine di fornire una base solida a quegli sviluppatori che vogliano creare applicazioni che supportino sia i dispositivi per realtà aumentata che quelli per realtà virtuale, in cui le due esperienze siano analoghe negli scopi e, soprattutto, nelle modalità di esecuzione.

Grazie al modo in cui gli oggetti virtuali sono percepiti nelle diverse forme di realtà mista, l'aspetto *sociale* delle applicazioni AR e VR è estremamente importante; infatti, dal punto di vista dell'utente, rende l'esperienza più appagante, mentre, dal punto di vista del ricercatore, è interessante per i meccanismi di interazione. Il ricercatore potrebbe essere interessato non solo ad approfondire la parte implementativa delle meccaniche di interazione, ma anche a studiare gli effetti dell'interazione utente-utente su aspetti quali la presenza e la percezione di consistenza degli oggetti virtuali con la realtà.

Le considerazioni sopraesposte hanno condotto alla scelta di integrare le fondamenta delle applicazioni multiutente nel framework, che quindi è fortemente orientato alla creazione di applicazioni in cui utenti AR e VR possano condividere lo stesso spazio di azione, interagire fra loro e con l'ambiente virtuale, con modalità simili, ma non uguali, in quanto ogni dispositivo può avere caratteristiche diverse non replicabili in alcun modo con gli altri dispositivi supportati.

Il framework qui discusso presenta un'architettura generale di tipo client-server, in cui il client costituisce l'applicazione AR/VR, mentre il server è una normale applicazione per desktop, che può essere eseguita su un qualsiasi PC, non necessariamente su un dispositivo AR o VR. Si è preferito adottare un'architettura client-server per tre ragioni principali:

[11] Software Development Kit, o SDK

[12] Consorzio americano *non-profit* nato con lo scopo di creare e promuovere standard aperti

1. Alcuni client[13] hanno limitata potenza computazionale: il server si prende in carico la maggior parte dei calcoli, alleggerendo il carico che grava sui client.

2. Banda limitata per i client: nel modello client-server, il client invia pacchetti solo al server, richiedendo in tal modo meno banda rispetto ad altre architetture (si veda 3.1.3).

3. Semplicità: poiché il framework è pensato principalmente come strumento per la ricerca, si è favorita la semplicità di sperimentazione, a discapito di altri aspetti. Ad esempio, l'architettura Peer-to-Peer (si veda la sezione 3.1.3) non avrebbe richiesto la presenza di un server dedicato, ma questo rappresenta un vantaggio trascurabile nell'ambito della ricerca. Al contrario, l'implementazione del modello Peer-to-Peer sarebbe stata più complicata rispetto al modello Client-Server.

I due aspetti principali del framework sono la sincronizzazione delle entità[14] tra server e client, e la gestione della vita dell'applicazione a partire dall'avvio, passando per la sessione[15], fino alla chiusura. Questi aspetti del framework ed altri saranno descritti dettagliatamente nel capitolo 3.

[13] Si ricorda che i client sono i dispositivi AR/VR

[14] Oggetto virtuale, caratterizzato da specifiche logiche di comportamento

[15] Fase della vita dell'applicazione in cui gli utenti sono effettivamente *in gioco*

Capitolo 2

Stato dell'arte

In questo capitolo sono analizzate la realtà aumentata e la realtà virtuale, a partire dalla definizione formale, proseguendo quindi con una breve rassegna delle tappe storiche più importanti che hanno portato allo stato attuale della tecnologia.

Segue la descrizione dei dispositivi oggi disponibili, e una breve analisi delle applicazioni pratiche di queste nuove tecnologie in vari settori dell'attività umana, dall'industria alla medicina, dal gaming alla cultura.

2.1 Realtà aumentata

La realtà aumentata può essere definita come l'aggiunta di elementi virtuali all'ambiente reale, da cui il termine aumentata. Volendo dare una definizione più precisa, si può partire da quella del 1994 di Paul Milgram e Fumio Kishino [23], che introdussero il concetto di *realtà mista* come "tutto ciò che è contenuto tra gli estremi del continuo della virtualità", in cui gli estremi sono la realtà pura, senza elementi virtuali, e la virtualità totale, in cui la realtà è completamente mascherata dal virtuale.

In questa definizione, l'AR è posizionato più verso l'estremo della realtà pura, poiché solo un numero ridotto di elementi virtuali sono presenti nella scena, che è in gran parte dominata dal reale (si veda la figura 2.1).

Gli elementi virtuali, comunemente riferiti con il nome di ologrammi, possono essere delle copie fedeli e indistinguibili di oggetti reali, ma anche testi, effetti sonori e messaggi audio, o pannelli e menù che tipicamente vediamo su uno schermo.

Così come interagiamo con la realtà attraverso il tatto, l'olfatto, l'udito, allo stesso modo sono possibili interazioni con gli ologrammi, in funzione del particolare dispositivo utilizzato: l'HoloLens, ad esempio, è in grado di rilevare dei gesti della mano che sono poi tradotti in input usati per manipolari gli ologrammi.

I vantaggi di stare dalla parte della realtà mista in cui si colloca l'AR, rispetto alla parte opposta, sono principalmente tecnici e di usabilità: dal punto di vista tecnico, il fatto che il rapporto tra il numero di ologrammi e quello di oggetti reali sia basso significa che

Figura 2.1. Pokemon GO esce nel 2016 e riceve un ottimo riscontro di pubblico. Foto tratta da [27].

i requisiti prestazionali dell'hardware sono ragionevolmente contenuti; inoltre, dal punto di vista dello sviluppo degli asset, un'applicazione per AR richiede meno modelli rispetto a una classica applicazione per desktop, console o smartphone.

Infine, grazie al fatto che l'utente ha continuamente il riferimento della realtà vera, è molto meno frequente l'insorgere di effetti collaterali negativi connessi al distacco dalla realtà, quali nausea e disorientamento, che tipicamente peggiorano quanto più ci si sposta dall'estremo dell'AR verso l'estremo opposto della realtà virtuale.

La definizione di realtà aumentata data sopra deriva da quella di realtà mista ed è, quindi, molto generale; volendo dare una definizione più accurata di AR, si può fare riferimento a quella proposta nel 1997 da Azuma [4], che riassume il concetto in esame indicandone tre caratteristiche:

1. Esso combina oggetti virtuali (gli ologrammi) con l'ambiente reale.
2. Gli oggetti virtuali sono disposti in modo coerente con l'ambiente in cui si trovano; ad esempio, un quadro virtuale sarà allineato alla parete.
3. L'applicazione gira in tempo reale e l'esperienza è interattiva.

Il primo punto non aggiunge nulla di nuovo rispetto alla definizione di Milgram e Kishino, al contrario del punto due, che introduce il concetto di coerenza tra virtuale e reale, con il quale si intende che la relazione tra gli oggetti virtuali e l'ambiente reale deve rispondere ad esigenze di coerenza semantica e di funzionalità; ad esempio, un menù

virtuale dovrebbe essere posto di fronte all'utente e non dovrebbe compenetrare gli oggetti reali o gli altri ologrammi, perdendo in tal modo *consistenza solida* (si veda la figura 2.2).

Figura 2.2. HoloLens permette di interagire con menù virtuali tramite i gesti.

Per poter disporre e allineare gli oggetti virtuali in modo desiderato, è necessario che l'applicazione software sappia dove si trovi l'utente in relazione agli oggetti stessi e all'ambiente circostante, cioè sia in grado di tracciare la posizione e l'orientamento dell'utente. Le diverse tecnologie impiegate per il tracciamento della posizione e dell'orientamento saranno discusse in dettaglio più avanti.

Le applicazioni AR, come da punto 3, sono *real-time*, nel senso che l'applicazione è continuamente aggiornata, processando gli input dell'utente e visualizzando i frame grafici ricreati tenendo conto della posizione dell'utente in ogni istante. In tal modo, si crea l'illusione che gli ologrammi siano connessi al mondo reale. Se, inoltre, l'esperienza è interattiva, cioè è possibile eseguire azioni che hanno un effetto sugli ologrammi, allora l'applicazione diventa più interessante e, soprattutto, potenzialmente più utile.

2.1.1 Storia

In questa sezione sono riportati, in ordine cronologico, gli eventi più importanti e i principali step evolutivi della realtà aumentata, a partire dagli anni 60 del secolo scorso ad oggi.

1. 1968, sviluppo del primo head-mounted display (HMD): Ivan Sutherland realizza il primo sistema per AR di tipo HMD con ottiche see-through, chiamato *Spada di Damocle* [35]. Il sistema, che era in grado di tracciare l'HMD (e, quindi, per estensione, la testa dell'utente che lo indossava), permetteva di mostrare dei modelli in *wireframe* generati al computer. Il curioso nome era legato al fatto che il sistema

fosse così pesante da dover essere sostenuto dal soffitto, da cui era calato per essere indossato dall'utente (si veda la figura 2.3).

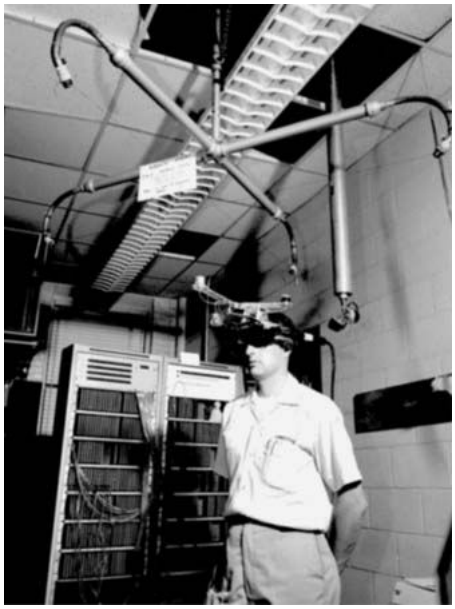

Figura 2.3. L'HMD ideato da Sutherland e denominato Spada di Damocle.

2. 1974, costruzione di un laboratorio di *realtà arfificiale* chiamato Videoplace: progettato da Myron Krueger e basato su video-proiettori, Videoplace immergeva gli utilizzatori in un ambiente virtuale interattivo.

3. 1990, Tom Caudell, un ricercatore che lavorava per Boeing, compagnia nota per la progettazione e produzione di aeroplani e velivoli in generale, conia il termine Realtà Aumentata (Augmented Reality).

4. 1992, sviluppo di un nuovo head-mounted display per applicazioni industriali: Tom Caudell e David Mizell, un altro ricercatore che lavorava per Boeing, realizzano un nuovo e più avanzato HMD impiegato nella costruzione di velivoli, come aiuto nell'assemblaggio delle parti di cavetteria [11].

5. 1992, costruzione di Virtual Fixtures (figura 2.4), uno dei primi sistemi completamente immersivi per AR: Louis Rosemberg progetta e sviluppa per l'Air Force americana un *esoscheletro* che permetteva di controllare dei macchinari per effettuare delle operazioni da remoto.

6. 1993, introduzione di KARMA: Steve Feiner e una squadra di studenti della Columbia University introducono un nuovo sistema per AR di tipo *knowledge-based*, cioè in grado di automatizzare la progettazione di presentazione che mostrano come effettuare una data operazione. Il sistema aveva la funzione di guida durante le operazioni di *maintenance* e riparazione [16].

Figura 2.4. Il sistema Virtual Fixtures ideato da Louis Rosemberg.

7. 1993, introduzione nel primo dispositivo *hand-held* per AR: Fitzmaurice sviluppa Chameleon [18], un dispositivo *spatially-aware* che veniva connesso a una workstation (all'epoca la potenza dei calcolatori portatili non era ancora sufficiente per applicazioni grafiche) e mostrava informazioni contestuali mentre l'utente si spostava da una parte all'altra dell'ambiente.

8. 1994, prima produzione teatrale che integra elementi virtuali: Julie Martin produce un'opera teatrale, chiamata Dancing in Cyberspace, in cui gli acrobati danzano in uno stage decorato da oggetti virtuali.

9. 1994, prima applicazione in campo medico dell'AR: State *et al.* dell'Università della North Carolina presentavano un'applicazione medica dell'AR che permetteva al medico di osservare il feto direttamente dentro la madre incinta.

10. 1995, sviluppo di un nuovo dispositivo hand-held per AR: presentato con il nome NaviCam, Jun Rekimoto e Nagao introducono un nuovo dispositivo hand-held, seppur ancora connesso a una workstation. La caratteristica più interessante di tale dispositivo era la presenza di una camera che permetteva la rilevazione di marker nell'ambiente, in corrispondenza ai quali venivano mostrate informazioni in una vista see-through [30].

11. 1996, Jun Rekimoto introduce CyberCode [31], il primo sistema AR che usa *marker 2D*.

12. 1996, introduzione della prima esperienza collaborativa in AR [36]: Schmalstieg *et al.* sviluppano un sistema che permette a più utenti di vedere e interagire con gli stessi elementi virtuali in modo collaborativo, grazie all'uso di un head-mounted display in grado di produrre immagini degli oggetti virtuali corrette tenendo conto della prospettiva dell'utilizzatore.

13. 1997, Il governo giapponese e Canon avviano una *joint venture* finalizzata al finanziamento del Mixed Reality Systems Laboratory, tra i cui risultati si ricorda COASTAR, il primo head-mounted display basato su tecnologia stereo video see-through.

14. 1997, presentazione di Touring Machine [17], il primo sistema AR per l'esterno: sviluppatto da Feiner *et al.* alla Columbia University, tale sistema, che richiedeva uno zaino per il trasporto, era basato sulla tecnologia GPS e tracking dell'orientamento.

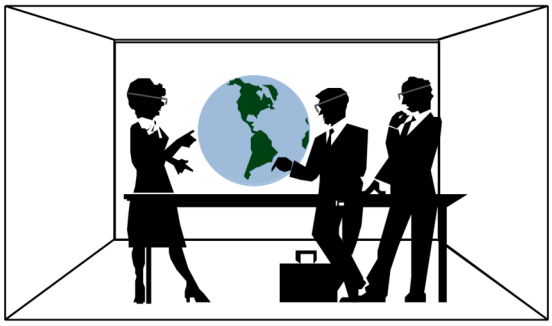

Figura 2.5. Illustrazione del concetto di esperienza collaborativo in AR. Figura tratta da [36].

15. 1998, La Nation Football League americana fa uso dell'AR durante una partita.

16. 1998, viene sviluppato ARQuake [43], versione AR del celebre videogioco rilasciato da ID Software, utilizzando Tinmith, un sistema per AR per l'esterno (si veda figura 2.6) costruito da Thomas *et al.* all'Università dell'Australia del Sud.

Figura 2.6. ARQuake, versione AR del celebre sparatutto originariamente sviluppato da ID Software.

17. 1998, introduzione del sistema per la telepresenza *The Office of the Future* [28]: sviluppato da Raskar *et al.* all'Università della North Carolina, faceva uso di molte delle tecnologie oggi usate per l'AR.

18. 1999, viene rilasciato, con il nome ARToolkit [19], il primo software open-source per AR: sviluppato da Hirokazu Kato e rilasciato attraversio l'HIT Lab dell'Università di Washington, ARToolkit include una libreria per il tracciamento di *fiducial marker*, che possono essere impiegati per sovrapporre oggetti virtuali su immagini reali. Nonostante esistano soluzioni più nuove, ARToolkit è ancora molto usato.

19. 2003, Wagner e Schmalstieg introducono il primo sistema hand-held per AR completamente integrato (non richiede una workstation esterna).
20. 2008, Wagner *et al.* introducono il primo sistema di tracciamento basato sul riconoscimento delle *feature* effettivamente utilizzabile per smartphone, precursore di Vuforia [1].
21. 2014, Google rilascia la versione *beta* di Google Glass.
22. 2016, Microsoft rilascia l'HoloLens, un prodotto destinato principalmente all'industria e alla ricerca, dato il suo costo elevato. Nello stesso anno, la compagnia Meta rilascia il Meta 2 Developer Kit.
23. 2016, viene rilasciato Pokemon Go (si veda la figura 2.1), uno dei videogiochi AR di maggior successo.
24. 2017, Google lancia ARCore, un SDK[1] per lo sviluppo di applicazioni AR su android. Nello stesso anno Apple annuncia ARKit.

2.1.2 Dispositivi

In questa sezione saranno presentati i principali dispositivi per AR disponibili oggi.

HoloLens

Prodotto da Microsoft e rilasciato nel 2016, HoloLens è un head-mounted display per AR, che include un innovativo sistema di tracciamento basato su sensori di profondità, delle camere RGB, un processore integrato e speaker integrati.

La tecnologia per il tracciamento è un'evoluzione di quella usata da un altro prodotto Microsoft, la Kinect, rilasciata nel 2010 come accessorio per la console da gioco Xbox 360; per determinare la posizione in relazione all'ambiente circostante, HoloLens sfrutta due camere di profondità, con le quali viene fatta una ricostruzione tridimensionale dell'ambiente e simultaneamente viene localizzato l'HMD. Tale tecnica è una forma di *Simultaneous localization and mapping* (si veda 2.1.3).

Per visualizzare gli ologrammi, l'HoloLens fa uso di due lenti trasparenti che permettono di vedere direttamente il mondo reale, al contrario degli smartphone, in cui la realtà è prima catturata dalla videocamera e poi riproposta sul display in modo aumentato.

L'interazione con gli ologrammi può avvenire in vari modi, in quanto il prodotto Microsoft supporta non solo il riconoscimento di gesti, ma è anche in grado di capire comandi vocali. Inoltre, con il visore è incluso un pulsantino, chiamato clicker (mostrato in figura 2.7), che si connette, senza cavi, all'HoloLens tramite bluetooth; il clicker può essere utilizzato in alternativa ai gesti o alla voce per fornire un input.

[1] Software Development Kit

Figura 2.7. Il clicker, venduto assieme all'HoloLens, è un pulsante che permette di dare un input in modo più tradizionale rispetto ai gesti o alla voce.

Meta

Meta è un HMD prodotto dall'omonima compagnia fondata nel 2013 da Meron Gribetz, quando egli era ancora uno studente della Columbia University. Al contrario di HoloLens, il Meta richiede un PC esterno per funzionare, non avendo una processing unit dedicata al suo interno.

La prima versione del Meta fu finanziata con successo tramite *Kickstarter*[2]; nel 2014, una versione dell'HMD destinata agli sviluppatori fu rilasciata con il nome Meta 1 Developer Kit.

Nel 2015, la compagnia riesce a raccogliere 23 milioni di dollari da una *venture capital firm*, e nell'anno successivo, in seguito alla presentazione della versione 2 del dipositivo, Meta riceve altri 50 milioni di dollari.

La versione 2 del Meta, chiamata semplicemente Meta 2, offre un campo visivo molto superiore rispetto all'HoloLens, pari a 90°; i display hanno una risoluzione di 2.5K e frequenza di aggiornamento pari a 60 Hz.

L'HMD include una videocamera RGB frontale, ed è in grado di tracciare la posizione della testa. Il Meta può tracciare le mani e, analogamente all'HoloLens, riconoscere alcuni gesti permettendo l'interazione con gli ologrammi.

Assieme al dispositivo, il produttore ha rilasciato l'SDK ufficiale per lo sviluppo su Unity.

2.1.3 Tecnologia

Aumentare la realtà in modo credibile e funzionale richiede che tutto il sistema AR, che include sia l'hardware che il software, soddisfi determinate specifiche, quali l'accuratezza e robustezza del sistema di tracciamento, la qualità della riproduzione degli ologrammi, e la velocità nella risposta agli input.

[2]Piattaforma online per il *crowd-funding*

Figura 2.8. Meron Gribetz, fondatore di Meta, presenta il Meta 2 durante un Ted talk.

Il sistema di tracciamento ha particolare importanza, in quanto da esso dipende il corretto allineamento degli oggetti virtuali con quelli reali; un'allineamento non ottimale può non solo rompere l'illusione di guardare a un oggetto effettivamente presente nella scena, ma anche rendere l'applicazione inutilizzabile. Per posizionare correttamente gli ologrammi, l'applicazione fa affidamento sulla misura della posizione e della rotazione della testa (cioè il punto dal quale è osservata la scena) che il sistema di tracciamento fornisce al software.

I valori misurati sono applicati alla camera della scena 3D, che rappresenta un ipotetico osservatore virtuale (in questo caso l'utente), quindi l'engine grafico esegue il rendering (cioè costruisce un'immagine, il frame, processando i modelli 3D dati sotto forma di poligoni e texture) della scena e questa viene sovrapposta alla realtà in modi diversi in funzione della particolare tecnologia impiegata.

Hand-held

Con *hand-held* si intende un dispositivo che viene tenuto in mano, come un tablet o uno smartphone. Proprio lo smartphone che quasi ogni persona al mondo possiede è un esempio di sistema AR, in quanto contiene tutti i componenti hardware fondamentali per creare un'esperienza di realtà aumentata:

1. Videocamera: gli smartphone, anche quelli di fascia bassa, sono muniti di almeno una videocamera posteriore.

2. Sensori (giroscopio, accelerometro): sono molto comuni in smartphone di fascia medio-alta.

3. Processori e memoria: l'hardware degli smartphone moderni è comparabile ai PC di alcuni anni fa, in termini di frequenza di *clock* e numero di core della CPU e quantità di RAM.

4. Schermo: nei dispositivi di fascia alta sono comuni risoluzioni uguali o superiori al full HD[3].

Uno smartphone o tablet può essere trasformato in un device per AR nel seguente modo: la videocamera può essere usata per catturare in real-time la scena reale, mentre i sensori possono essere impiegati per determinare la rotazione dello smartphone. Il processore interno renderizza gli oggetti virtuali, quindi combina il render prodotto al frame catturato dalla videocamera; inoltre, la posizione del dispositivo può essere messa in relazione con i riferimenti eventualmente presenti nella scena. Infine, l'immagine finale composta può essere presentata a schermo.

Figura 2.9. Esempio di applicazione AR su tablet. Immagine tratta da [40].

Wearable

Ai dispositivi wearable[4] appartengono gli head-mounted display, che possono essere indossati come dei normali occhiali, con l'ovvio vantaggio di tenere le mani libere per compiere

[3]Full HD equivale a 1920x1080 pixel

[4]In italiano, indossabili

task di vario genere.

Uno degli svantaggi degli HMD è di essere scomodi da indossare e stancanti da usare per lunghe sessioni, in quanto i dispositivi attualmente presenti sul mercato sono di dimensioni non trascurabili e pesanti.

I dispositivi indossabili possono essere di due tipi:

1. Optical See-through

2. Video See-through

Optical see-through Si definisce *Optical see-through* un display trasparente o semitrasparente che permette all'utente di vedere il mondo reale in modo naturale, anche se filtrato dagli strati che compongono lo schermo. I vantaggi (e gli svantaggi) di un tale device sono molteplici, a partire dalla sicurezza; infatti, anche in caso di malfunzionamento, l'utente continua a vedere il mondo reale e può reagire immediatamente ed efficaciemente in situazioni impreviste. Dal punto di vista della qualità dell'immagine, poiché l'utilizzatore vede il mondo reale direttamente, non c'è alcuna perdita di definizione, come invece succede nelle soluzioni video see-through.

Gli svantaggi principali sono il campo visivo limitato, che risulta in un effetto *tagliato* degli ologrammi, e la scarsa luminosità e constrasto, che rendono i dispositivi basati su questa tecnologia inadeguati per l'uso all'aperto, o comunque in un ambiente fortemente illuminato.

Video see-through Questa tecnologia non è solo meno costosa della precedente, ma è anche molto più semplice costruire dispositivi basati su di essa.

L'immagine del mondo reale catturata da una videocamera è combinata con l'immagine degli oggetti virtuali e riprodotta sullo schermo; ciò comporta una serie di vantaggi e svantaggi.

Tra i primi, il più importante è la possibilità di utilizzare il device in luoghi molto luminosi, perché è facile ottenere un'immagine chiara e nitida degli ologrammi; inoltre, per una resa ottimale, è possibile misurare la luminosità della scena, al fine di integrare al meglio gli ologrammi con l'ambiente reale [21].

Dall'altra parte, la definizione dell'immagine soffre per la vicinanza al display, che a pochi centimetri dagli occhi mostra la struttura dei pixel. Soprattutto esiste un problema dovuto all'effetto di parallasse che può causare disorientamento nell'utilizzatore: infatti, per limiti fisici, la videocamera non può essere posta in corrispondenza degli occhi, perciò l'immagine catturata differisce da ciò che vedrebbero gli occhi naturalmente. Sono state sviluppate delle tecniche per risolvere questo problema tramite allineamento delle immagini catturate dalla videocamera [38].

Infine, quando un solo schermo è utilizzato per entrambi gli occhi, l'affaticamento della vista è significativamente maggiore rispetto all'uso di due schermi separati [15].

Proiettori

Il paradigma della realtà aumentata proiettiva fa uso di proiettori che sono in grado di ricreare delle immagini direttamente sulle superfici di oggetti, che possono essere non solo piani e colorati in modo uniforme (ad esempio le pareti), ma anche complessi [8].

Rispetto alle altre tecnologie viste, l'AR basata sulla proiezione ha dei vantaggi non trascurabili, a partire dal fatto che non è necessario indossare scomodi e/o pesanti HMD, che possono portare ad un eccessivo affaticamento dell'utilizzatore dopo lunghe sessioni. Un altro vantaggio è che il campo visivo coperto è molto più ampio, con ricadute positive sull'utilità e sull'usabilità del sistema AR. Infine, un altro vantaggio da non sottovalutare è il fatto che i proiettori, essendo tipicamente alimentati tramite la rete elettrica, non hanno il problema della durata limitata delle batterie.

Gli svantaggi più significativi sono la necessità di ricalibrare i proiettori ogniqualvolta l'ambiente subisce modifiche, o cambia la distanza delle superfici dai proiettori. Fortunatamente, è possibile automatizzare la calibrazione dei proiettori tramite l'uso di camere [29].

Altri aspetti negativi sono la bassa luminosità e contrasto, che limitano l'uso dei proiettori ad ambienti chiusi.

Sistemi di tracciamento

Come detto nell'introduzione alla tecnologia dei sistemi AR, migliore è l'accuratezza nel tracciamento, migliore sarà l'allineamento degli oggetti virtuali rispetto al reale. In questa sezione saranno discussi i sistemi di tracciamento disponibili, analizzando più dettagliatamente le principali soluzioni oggi adottate dai dispositivi AR disponibili sul mercato e non.

Per quanto riguarda la posizione, le soluzioni più comuni sono basate su tracker ottici, in cui le immagini catturate da una o più camere sono processate alla ricerca di riferimenti, ad esempio marker, che permettano di determinare la posizione della testa rispetto ai riferimenti stessi. Altre tecnologie comunemente impiegate sono basate su misure inerziali, magnetiche o meccaniche. La rotazione è spesso ottenuta tramite giroscopi, comunemente presenti negli smartphone.

Marker-based Quando la localizzazione della HMD o del device hand-held avviene per mezzo di marker opportunamente disseminati nell'area di utilizzo del sistema AR, si parla di sistema di tracciamento *marker-based*.

I vantaggi di un tale sistema sono la semplicità e il costo ridotto, a discapito del fatto che è necessario un elemento aggiuntivo, il marker per l'appunto, per poter utilizzare il sistema.

Simultaneous localization and mapping Il problema della ricostruzione di un'area e della simultanea localizzazione di un attore nell'area stessa prende il nome di Simultaneous localization and mapping, o SLAM.

Il sistema di tracciamento di HoloLens è un esempio di SLAM, in cui l'HMD è continuamente tracciato mentre l'area circostante l'utente è mappata e ricostruita in 3D.

2.1.4 Aree applicative

I campi di applicazione della tecnologia AR sono molteplici, e spaziano dall'intrattenimento, alla medicina, passando dall'architettura, fino al militare. Più i prezzi dei prodotti della fascia consumer si riducono, più l'AR diventa abbordabile e smette di essere qualcosa unicamente relegata all'ambito della ricerca e del militare, diffondendosi tra il grande pubblico.

Gaming

Nell'industra dell'intrattenimento, che è un settore dal valore di centinaia di miliardi di euro, le applicazioni AR stanno gradualmente prendendo piede, specialmente in ambito gaming, con particolare riferimento agli smartphone.

Sebbene gli smartphone di oggi abbiano una potenza computazionale invidiabile, se confrontata con i primi cellulari o anche con i primi modelli di smartphone, e sia possibile trovare applicazioni ludiche graficamente avanzate, quali Asphalt 9 Legends e Modern Combat 5: Blackout, la particolare natura delle applicazioni AR, che richiede di aggiungere (e quindi processare) solo un numero limitato di oggetti virtuali allo sfondo reale, è estremamente conveniente, anche solo dal punto di vista del consumo della batteria, per non parlare dei costi di sviluppo più contenuti rispetto ad applicazioni 3D classiche.

Inoltre, la caratteristica di portabilità degli smartphone, connessa al fatto che sentiamo il bisogno di tenere lo smartphone sempre con noi, favorisce applicazioni in cui si trae vantaggio dagli spazi aperti. Si pensi, ad esempio, al gioco Pokemon GO; in questo caso, l'intero pianeta in cui viviamo diventa un immenso parco giochi.

Cultura e turismo

I musei possono trarre vantaggio da tecnologie come la Realtà Aumentata, sia come driver per attirare i più giovani, che come strumento per migliorare e rendere più fruibili le proprie offerte culturali.

Per esempio, un modo di utilizzare l'AR può essere un'applicazione per smartphone che, inquadrata un'opera con la fotocamera, fornisca una descrizione, in forma testuale o audio, dell'opera stessa. Ciò potrebbe aiutare ad ottimizzare gli spazi del museo, liberando molti spazi che potrebbero essere occupati da tante opere d'arte che spesso rimangono "nascoste".

Il turismo è un altro di quei settori in cui la diffusione degli smartphone e l'AR possono migliorare l'esperienza notevolmente per i viaggiatori di tutti il globo. La realtà aumentata è uno strumento potenzialmente molto utile perché può aiutare il turista a capire dove si trova, può indicargli luoghi di interesse e condurlo nel posto desiderato percorrendo la strada più breve.

Si pensi, ad esempio, alla possibilità di potenziare e rendere più intuitive applicazioni quali Google Maps, sostituendo la classica vista dall'alto con le indicazioni per raggiungere una data destinazione, con una più intuitiva vista in prima persona delle strade, aumentata con le indicazioni su dove andara (si veda la figura 2.10). Non a caso Google sta lavorando proprio su una versione di Google Maps per AR.

Figura 2.10. Versione AR di Google Maps. Foto tratta da [42].

Istruzione

In ambito educativo, la realtà aumentata è un valido strumento di cui insegnanti ed educatori, sempre alla ricerca di nuove soluzioni per migliorare l'offerta scolastica, possono trarre vantaggio per rendere l'insegnamento non solo più efficace, ma anche più efficiente. Infatti, gli studenti sono molto più portati a capire e memorizzare tramite l'esperienza diretta e la pratica, di cui gli ologrammi sono un valido sostituto [14].

La capacità, offerta dalla tecnologia AR, di arricchire la scena con testi, suoni, modelli 3D, immagini e video, rappresenta uno stimolo aggiuntivo per lo studente, che si sente più coinvolto durante la lezione. Inoltre, i segnali audiovisivi di cui è inondato sono un aiuto per comprendere meglio l'argomento trattato.
Lo studente si sente più coinvolto per la possibilità di interagire con gli ologrammi, avendo un ruolo attivo nel processo di apprendimento [24].

Esempi di AR includono ricostruzioni di eventi storici, quali battaglie e assedi, in cui gli studenti possono osservare in prima persona l'evoluzione della battaglia, oppure applicazioni di chimica in cui è possibile visualizzare molecole in tre dimensioni [6].

Architettura

Architetti e progettisti devono spesso fare i conti con la difficoltà nel comunicare ai clienti, in modo chiaro e comprensibile, le idee di progetto, al fine di avere un riscontro, sia esso positivo o negativo, sui vari aspetti del progetto proposto. La realtà aumentata può essere

di grande aiuto in ciò, permettendo al committente di vedere in anticipo una ricostruzione in 3D del lavoro commissionato, prima che questo sia portato a termine.

Uno dei modi in cui l'AR può essere sfruttata dagli architetti è la visualizzazione in tre dimensioni delle opere progettate, come oggetti virtuali sovrapposti al mondo reale. Persone interessate al progetto a vario titolo, quali committenti e responsabili, possono valutare con i propri occhi, muovendosi nell'ambiente reale, la validità dell'opera proposta, in termini di funzionalità, estetica e coerenza con l'ambiente circostante.

L'utilità della realtà aumentata non è solo limitata alla fase di progettazione dell'opera, ma anche alle fasi della costruzione e della manutenzione una volta che l'opera è completata. Una tecnica molto utile è la visualizzazione di strutture nascoste quali cavi elettrici, condutture per il gas e per l'acqua, in modo analogo a una radiografia, ma in tempo reale.

Nonostante l'AR sia molto interessante dalla prospettiva dell'architetto, essa è poco presente nei processi di progettazione e realizzazione delle opere architettoniche, soprattutto perché manca un'adeguata integrazione con gli altri strumenti, quali ad esempio i software CAD[5], e processi propri dell'architettura [9].

Infine, la realtà aumentata può avere un importante ruolo anche nella preservazione del patrimonio architettonico; un esempio in tal senso è l'applicazione CityViewAR [20] sviluppata, in seguito al terremoto di Christchurch, dall'Università di Canterbury. Essa ha una duplice funzione: aiutare, da una parte, gli ingegneri a visualizzare gli edifici distrutti dal terremoto, mentre dall'altra serve come memoria storica, ricordandoci la forza distruttiva della natura.

Industria

L'industria è uno di quei settori che più attivamente ha abbracciato la novità dell'AR, con lo scopo di ridurre i costi, aumentando l'efficienza dei processi produttivi.

L'uso della realtà aumentata non è solo limitato alla fase di produzione, ma anche al *maintenance* di macchinari, e più in generale di sistemi che richiedono controlli costanti per il corretto funzionamento e la prevenzione di guasti.

Un esempio di impiego dell'AR in tal senso è il maintenance remoto, in cui un tecnico addestrato guida a distanza l'operatore che fisicamente si trova nei pressi del macchinario, indicando, o anche mostrando direttamente, le operazioni da svolgere [10].

Militare

In ambito militare, la Realtà Aumentata può non solo costituire uno strumento per il *training*, ma anche essere un'arma da sfruttare a proprio vantaggio sul campo di battaglia.

Quando usata per l'addestramento dei soldati, si parla di Battlefield Augmented Reality System (BARS): sul campo di addestramento reale sono disseminati i vari elementi

[5] Computer-Aided Design

bellici al fine di ricostruire uno scenario di guerra realistico. Uno dei vantaggi principali è sicuramente economico: l'uso di oggetti virtuali ha un costo inferiore rispetto a dispiegare l'arsenale e i mezzi militari. Ma il vero vantaggio risiede nella possibilità di ricostruire scenari di guerra sempre più unici in cui le forze dell'ordine si ritrovano oggi ad affrontare operazioni molto diverse e complesse.

Diverso è il contesto dell'impiego della tecnologia AR durante gli scontri reali, in quanto il malfunzionamento dei dispositivi impiegati sul campo di guerra può causare il ferimento di un soldato e il fallimento della missione. Un possibile uso dell'AR in tale contesto sono gli smart glasses che forniscono al soldato informazioni aggiuntive sugli obiettivi della missione, sullo stato dei compagni e su possibili minacce incombenti.

A differenza di altri ambiti, i requisiti dei dispositivi impiegati in ambito militare sono molto alti, per quanto detto sopra, sia in termini di potenza computazionale, che in termini di robustezza e accuratezza del sistema di tracciamento.

Medicina

In campo medico, la Realtà Aumentata ha moltissime applicazioni, sia nella sala operatoria di un ospedale, sia nella camera da letto o palestra dove un paziente fa riabilitazione in seguito all'operazione.

Un esempio di impiego dell'AR per favorire il recupero dagli infortuni è la riabilitazione motoria: al paziente viene mostrato in modo chiaro ed intuitivo come effettuare un dato esercizio o movimento, in tal modo egli è in grado di capire più facilmente come eseguire l'esercizio e migliora in maniera molto più efficace.

In sala operatoria, l'AR può essere uno strumento di supporto molto utile per chirurghi e infermieri: gli infermieri sono aiutati nell'individuare più facilmente le vene o localizzare gli strumenti, mentre il chirurgo può visualizzare i dati del paziente direttamente sovrapposti al paziente, liberando lo spazio da monitor che possono rappresentare un ostacolo durante l'operazione chirurgica.

Stanno diventando sempre più comuni l'assistenza sanitaria a distanza e la telemedicina, cioè la possibilità, da parte del medico, di assistere e visitare un paziente a distanza, tramite la rete internet e, per l'appunto, la tecnologia AR. I vantaggi sono molteplici, dal costo economico ridotto sia per il paziente, che non ha più bisogno di recarsi fisicamente dal medico di fiducia, e non è costretto ad attendere il proprio turno in ambulatorio o nello studio del medico, che per il medico, che può ottimizzare il tempo della visita.

2.2 Realtà virtuale

La Realtà Virtuale (dall'inglese Virtual Reality, o VR) si trova all'estremo opposto del continuum della Realtà Mista rispetto all'AR, poiché l'utente è *completamente* immerso nel mondo virtuale.

2.2.1 Storia

In questa sezione sono riportati, in ordine cronologico, alcuni degli eventi e delle scoperte più importanti relativi alla Realtà Virtuale, dagli anni '60 del secolo scorso ad oggi:

1. 1960, primo head-mounted display per Realtà Virtuale inventato dal regista e inventore Morton Heilig: il dispositivo offriva uno schermo stereoscopico con ampio campo visivo e speaker stereo, ma nessun sistema di tracciamento.

2. 1962, primo esempio di sistema multimodale: negli anni 50 Morton Heilig inventa una macchina, chiamata Sensorama e brevetta nel 1962, per arricchire, tramite audio binaurale, effetti di vibrazione e odori, un film a colori. Durante la proiezione del film, in momenti prefissati e legati al contesto della storia, lo spettatore è stimolato tramite odori, vento artificiale generato da un sistema di ventole, suoni binaurale generati da speaker stereo, e vibrazioni generate da una sedia vibrante.

3. 1965, introduzione del concetto 'The Ultimate Display' [34]: Ivan Sutherland propose l'idea di un mondo sintetico in cui l'utente può interagire in modo realistico con gli oggetti virtuali attraverso suoni, tatto, olfatto. L'utente utilizzerebbe un HMD per immergersi nel mondo virtuale, che sarebbe processato in tempo reale da un calcolatore [34].

4. 1968, sviluppo del primo head-mounted display (HMD): Ivan Sutherland realizza il primo sistema per AR di tipo HMD [35] con ottiche see-through, chiamato *Spada di Damocle*. Il sistema, che era in grado di tracciare l'HMD (e, quindi, per estensione, la testa dell'utente che lo indossava), permetteva di mostrare dei modelli in *wireframe* generati al computer. Il curioso nome era legato al fatto che il sistema fosse così pesante da dover essere sostenuto dal soffitto, da cui era calato per essere indossato dall'utente.

5. 1974, costruzione di un laboratorio di *realtà arfificiale* chiamato Videoplace: progettato da Myron Krueger e basato su video-proiettori, Videoplace immergeva gli utilizzatori in un ambiente virtuale interattivo.

6. 1978, sviluppo di VCASS [44], un simulatore di volo con HMD: sviluppato da Hollace H. Warner e Thomas Furness negli Armstrong Medical Research Laboratories dell'Air Force americana, *Visually Coupled Airborne Systems Simulator* era un avanzato simulatore di volo in cui il pilota indossava un HMD nel quale venivano proiettate informazioni generate al computer quali mappe 3D, immagini radar e dati di volo.

7. 1984, costruzione di VIVED (figura 2.11), un HMD stereocopico monocromatico: la *National Aeronautics and Space Administration* (NASA) costruisce *Virtual Visual Environment Display*, un HMD basato su tecnologia commerciale.

8. 1985, messa in commercio del primo prodotto commerciale VR: il produttore VPL inizia a vendere DataGlove, un guanto tracciato in grado di ricostruire i movimenti della mano (si vedano le figure 2.12 e 1.4).

Figura 2.11. Il Virtual Visual Environment Display realizzatato dalla **NASA**.

9. 1986, viene coniato il termine Realtà Virtuale, da parte di Jaron Lanier, fondatore di Visual Programming Lab (VPL).

10. 1988, messa in commercio del primo HMD per VR: VPL produce il primo HMD per consumer, chiamato EyePhone (si veda la figura 2.12).

11. 1989, commercializzazione di BOOM da Fake Space Labs: BOOM è un box contenente due schermi CRT visibili attraverso due appositi buchi; l'utilizzatore poteva muoversi nel mondo virtuale, in quanto un sistema meccanico permetteva di tracciarne lo spostamento.

12. 1991, iniziano a diffondersi le prime sale giochi in cui è possibile giocare a videogiochi per VR su macchine ad hoc, chiamate Virtuality Group Arcade Machines.

13. 1992, viene presentato CAVE [12], un sistema per realtà virtuale che non fa uso di HMD: *CAVE Automatic Virtual Environment* utilizza dei proiettori per generare immagini sulle pareti di una stanza (si veda la figura 2.13), risultando in immagini di risoluzione più elevata e un più ampio campo visivo. L'utente deve comunque indossare degli occhiali LCD di tipo *shutter* per ottenere una vista stereoscopica e percepire la profondità.

Figura 2.12. Foto di Eyephone e Dataglove prodotti da VPL.

Figura 2.13. CAVE permette a più utenti di condividere la stessa esperienza di VR.

14. 1993, la compagnia giapponese SEGA annuncia lo sviluppo di un nuovo visore per VR, con schermi LCD, sound stereo e tracciamento della testa: sebbene il produttore e sviluppatore sviluppi quattro videogiochi per il dispositivo, esso non sarà mai messo in commercio.

15. 1995, Nintendo rilascia Virtual Boy, la prima console portatile in grado di visualizzare grafica 3D, sebbene solo in rosso e nero.

16. 2012, viene lanciato il kickstarter di Oculus Rift, un nuovo visore VR prodotto dalla neonata compagnia Oculus: il kickstarter ha un enorme successo, raccogliendo 2.5 M$ e riaccendendo l'interesse per la Realtà Aumentata.

17. 2014, Google rilascia Google Cardboard, un visore low-cost per smartphone in cartone: il visore ha la sola funzione di contenitore per lo smartphone, in quanto quest'ultimo dispone di tutti gli elementi principali di un sistema VR, quali il tracciamento (limitatamente alla rotazione tramite giroscopio), l'hardware per il processing e il display.

18. 2016, Oculus rilascia la prima versione consumer del Rift (nome in codice CV1, cioè Consumer Version 1); nello stesso anno, HTC, in collaborazione con Valve, rilascia il visore HTC Vive, che include anche i primi motion controller basati sulla tecnologia per il tracciamento *Lighthouse* sviluppata da Valve.

2.2.2 Dispositivi

In questa sezione viene fatta una breve rassegna dei principali dispositivi VR in commercio.

Oculus Rift CV1

Dopo aver prodotto due modelli indirizzati agli sviluppatori di software, denominati per l'appunto DK1 e DK2 (Developer Kit), Oculus rilascia la prima versione destinata al mercato consumer dell'Oculus Rift (si veda la figura 2.14).

Caratteristiche tecniche Il visore utilizza un sistema a doppio display, con risoluzione 1200×1080 pixel, sistema di tracciamento ottico e speaker stereo.

I display, aggiornati a una frequenza di 90 Hz, sono di tipo AMOLED, una variante della tecnologia OLED comunemente impiegata nella produzione di display di piccole dimensioni, come quelli presenti negli smartphone. La scelta della tecnologia OLED, rispetto alla meno costosa tecnologia LCD, è giustificata dalla velocità di risposta molto inferiore dell'OLED, che permette, tramite una tecnica chiamata *low persistence* o *strobing*, di ridurre il motion blur dovuto alla persistenza dell'immagine sulla retina.

Grazie alla presenza di due display distinti, la distanza interpupillare tra i display può essere regolata tramite un sistema meccanico.

Sistema di tracciamento Il sistema di tracciamento, denominato *Constellation Tracking System*, è di tipo ottico e utilizza due sensori (mostrati in figura 2.15) in grado di rilevare la posizione dei led infrarossi che ricoprono il visore. A partire da tali valori, il sistema di tracciamento determina la posizione e rotazione del visore, in relazione al sensore. Questo è un esempio di sistema di tracciamento outside-in, in cui i sensori sono esterni.

Figura 2.14. Il visore VR Oculus Rift CV1 rilasciato nel 2016.

Figura 2.15. Sensore ottico *Constellation* incluso con Oculus Rift CV1.

Oculus Rift S

Nel 2019 Oculus rilascia sul mercato una nuova versione del Rift, denominata Rift S. A differenza della versione precedente, il nuovo visore ha un singolo schermo LCD con risoluzione 2560×1440 pixel e refresh-rate pari a 80 Hz.

Oltre al nuovo display, Rift S vanta anche un nuovo sistema di tracciamento di tipo inside-out, denominato *Oculus Insight*. Utilizzando cinque camere poste sul visore, Rift S è in grado di tracciare la propria posizione e rotazione in relazione all'ambiente circostante; inoltre, esso è in grado di tracciare i led a infrarossi dei controller.

Oculus Quest

Fin dal 2016 Oculus annuncia l'interesse verso dispositivi *standalone*, con l'obiettivo dichiarato di entrare direttamente nella fascia mobile del mercato del VR. Nel 2018 viene prima rilasciato il visore Oculus Go, seguito nel 2019 dall'Oculus Quest.

Caratteristiche tecniche Basato sul chip Qualcomm Snapdragon 835, l'Oculus Quest è un dispositivo standalone che offre un sistema di tracciamento a sei gradi di libertà (posizione e rotazione), motion controller, display OLED a risoluzione 1440×1600 e frequenza di aggiornamento pari a 72 Hz.

HTC Vive

Frutto delle ricerche svolte da Valve in ambito VR, l'HTC Vive è il risultato della collaborazione tra la compagnia taiwanese HTC e lo sviluppatore americano Valve.

Caratteristiche tecniche Le caratteristiche del visore sono in larga parte uguali a quelle dell'Oculus Rift CV1: il visore sfoggia un doppio schermo AMOLED con risoluzione pari a 1200×1080 pixel e refresh-rate a 90 Hz, campo visivo di circa 110° (misura lungo la diagonale). La differenza principale è data dal sistema di tracciamento, detto *Lighthouse*, che permette al dispositivo di supportare una modalità di tracciamento a 360 gradi in un volume di pochi metri quadri, detta *room-scale*.

Sistema di tracciamento Il sistema di tracciamento Lighthouse, sviluppato da Valve, fa uso di base station che generano dei fasci laser; dei sensori fotorecettori, disseminati sul visore e sui controller (o, più in generale, sull'oggetto da tracciare), rilevano la loro posizione nello spazio, permettendo di determinare la posizione e rotazione dell'oggetto tracciato.

Samsung Gear VR

Nel 2015 Samsung rilascia il Samsung Gear VR, prodotto in collaborazione con Oculus, come accessorio per la sua linea di smartphone di fascia alta.

Il Samsung Gear VR può essere visto come una versione molto più avanzata di Google Cardboard, in quanto costituisce un contenitore in cui inserire il proprio smartphone Samsung, che fungerà da display e *processing unit*.

2.2.3 Aree applicative

Intrattenimento e cultura

Negli ultimi anni, grazie soprattutto alla nuova ondata di visori commerciali iniziata da Oculus Rift, si è riacceso nel grande pubblico l'interesse verso la realtà virtuale, in particolare in ambito gaming e più in generale nell'ambito dell'intrattenimento. Inoltre, altre attività e aree per natura più orientate all'educazione, si sono aperte alla novità della VR per provare ad attirare il pubblico più giovane, solitamente più attento alle novità tecnologiche; quindi possiamo trovare musei e parchi a tema virtuali, esibizioni teatrali interattive e altro ancora.

Il driver principale che accomuna queste nuove esperienze è il senso di immersione e coinvolgimento del pubblico, che non è più parte passiva, ma può avere, se lo desidera, un ruolo attivo. L'interazione diretta che permettono le nuove tecnologie di realtà virtuale e aumentata diventa un mezzo per comunicare informazioni in modi nuovi e interessanti. Si pensi, ad esempio, a un museo che esponga ricostruzioni al computer di strumenti e macchinari del passato; indossando dei visori VR, il pubblico può osservare da diverse angolazioni la ricostruzione 3D, come se avesse di fronte l'oggetto originale; inoltre, poiché si tratta di un modello 3D, è possibile permettere al pubblico di interagire con lo strumento o macchinario riprodotto, aumentando da una parte il coinvolgimento, e dall'altra l'aspetto di apprendimento.

Esempi simili a quello del macchinario riprodotto in 3D sono le ricostruzioni al computer di elementi architettonici, o anche di interi edifici storici e antichi, ormai non più esistenti, in quanto distrutti dal tempo o da disastri naturali o causati dall'uomo. Tramite la VR, diventa quindi possibile riportare alla vita ciò che non esiste più, e ammirarlo con i propri occhi in un modo più naturale rispetto, ad esempio, a guardarne una riproduzione su un televisore. Inoltre, grazie al sistema di tracciamento dei moderni visori VR, è possibile muoversi liberamente in uno spazio per osservare la ricostruzione da diversi punti di vista.

Per quanto riguarda il gaming, l'adozione della tecnologia VR ha aperto la strada a nuovi generi e forme di interazione, in cui il giocatore non è più seduto in posizione quasi fissa a premere i tasti di un controller tradizionale, ma deve stare in piedi e muoversi nell'area di gioco, eseguendo le azioni di gioco in modo più naturale ed intuitivo. Uno dei principali vantaggi di questo nuovo approccio all'interazione ludica è legato alla salute fisica; richiedendo al giocatore di muoversi, lo si stimola ed incoraggia all'attività fisica, con ripercussioni positive sulla salute.

I generi che traggono più giovamento dalla VR sono quelli in cui il giocatore si trova dentro un veicolo (come nei giochi di guida), un velivolo (come nei simulatori di volo) o

anche in una macchina di fantasia (come può essere un Mech), in quanto è molto forte il senso di presenza, e, soprattutto, i sintomi negativi della nausea tendono a manifestarsi più raramente quando l'utente ha un riferimento solido (ad esempio, il cruscotto dell'auto) nel proprio campo visivo.

Altri generi particolarmente adatti alla VR sono quelli in cui il giocatore ha una prospettiva in prima persona e può muoversi in modo naturale.

Siccome i videogiochi tradizionali sono pensati assumendo che l'avatar del giocatore possa percorrere grandi distanze, l'integrazione della realtà virtuale non è sempre immediata; anzi, in molti casi non è proprio possibile supportare un dispositivo VR, ammenoché le meccaniche di gioco, specialmente quelle legate al movimento, non siano pesantemente ripensate e adattate (o rifatta appositamente) alla VR.

I metodi di locomozione oggi più impiegati sono qui elencati:

1. Movimento classico tramite levetta analogica: semplice da implementare, ma va assolutamente evitato in quanto induce nausea.

2. Movimento naturale: il giocatore si muove fisicamente nell'area di gioco. L'unico, e grosso, difetto è che l'area di gioco è limitata dallo spazio che l'utente ha a disposizione, e da altri impedimenti, quali i cavi.

3. Teletrasporto: il giocatore indica, ad esempio tramite puntatore sul motion controller, il punto in cui si vuole spostare, quindi da il comando tramite opportuno tasto. Buon sistema per prevenire la nausea, ma non sempre adatto al tipo di gioco (si pensi ad un gioco competitivo in cui i partecipanti si sparano a vicenda; il teletrasporto renderebbe difficile seguire il bersaglio).

4. Portali: simile al teletrasporto, con la differenza che l'utente crea una coppia di portali, uno dei quali in prossimità di dove si trova, e l'altro dove vuole andare; il giocatore entra nel primo spostandosi fisicamente, e si ritrova fuori dal secondo.

5. ArmSwinger: questo metodo di locomozione permette di muoversi per lunghe distanze, cercando al contempo di non indurre nausea. Il giocatore deve fare oscillare le braccia come se stesse camminando o correndo, ma rimanendo fisicamente fermo sul posto, mentre l'avatar si muove proporzionalmente alla frequenza di oscillazione.

Istruzione

Vivendo in una società tecnologica, le giovani generazioni hanno familiarità nell'uso di tutte le forme di tecnologia, crescendo, fin dalla fanciullezza, a contatto con dispositivi tecnologici più o meno avanzati. A differenza degli adulti, che tendono ad essere più timorosi di fronte alla tecnologia, i giovani mostrano molta meno esitazione nel provare le ultime novità di cui sentono parlare grazie alla pubblicità o al passaparola. Negli ultimi anni la scuola, in tutti i suoi diversi livelli dalla scuola materna all'università, hanno iniziato a includere la VR e l'AR nei processi di insegnamento, al fine di migliorare l'offerta didattica.

Usi delle tecnologie di realtà mista includono i laboratori virtuali [39], in cui gli studenti possono interagire non solo con la materia (virtuale) del laboratorio, ma anche fra loro, in modi nuovi e stimolanti.

Un esempio può essere un laboratorio di chimica, in cui gli studenti possono osservare, ricostruite in tre dimensioni, le molecole e i tipi di legami tra atomi, da un punto di vista che rende molto più intuitivo comprendere l'oggetto in esame.

Un altro esempio di integrazione della VR nella didattica è la possibilità di osservare il sole e i pianeti, così come gli altri corpi celesti e perfino le galassie, in 3D, per capire, ad esempio, come avvengono i moti di rivoluzione e rotazione dei pianeti.

Architettura

Quando si acquista una casa, si rifa la cucina o il bagno, o più in generale quando si ristruttura casa, sta diventando sempre più frequente che il committente abbia la possibilità di vedere in anticipo (cioè prima che i lavori inizino) il risultato finale del lavoro, ricostruito fedelmente in 3D, talvolta anche in modo immersivo, cioè utilizzando un casco per la Realtà Virtuale.

Questa pratica, detta *architectural visualization*, permette non solo al cliente di valutare il risultato, ma anche all'architetto di verificare la validità del proprio progetto. Inoltre, essendo la ricostruzione virtuale molto meno dispendiosa e *time-consuming*, è possibile apportare modifiche, anche frequenti, al progetto originale; valutando diversi fattori e aspetti del progetto prima dell'effettiva costruzione, si riducono grandemente gli errori progettuali.

Visitare virtualmente ciò che deve essere realizzato permette, infine, di valutare in prima persona dimensioni e spazi, che sarebbero, altrimenti, difficilmente stimabili da un disegno tecnico per una persona comune.

Un altro settore in cui la capacità di osservare il prodotto finale prima che questo sia effettivamente creato può essere estremamente utile è quello della costruzione di grandi edifici e grattacieli. Ad esempio, fare una simulazione della costruzione può aiutare ad ottimizzare il processo di costruzione, oltre che a prevenire eventuali criticità prima che queste si manifestino, riducendo in tal modo il tempo necessario per la realizzazione dell'edificio.

Produzione e marketing

Nella fase di produzione di un prodotto, processi basati su nuove tecnologie quali la realtà virtuale possono sostituire processi tradizionali ritenuti più costosi, insicuri ed inefficienti. Ad esempio, prima di raggiungere la forma finale, un prodotto subisce una serie di modifiche, evolvendo dall'idea iniziale fino a diventare il prodotto che si trova sugli scaffali del proprio negozio di fiducia. Il processo di prototipazione può essere estremamente costoso e lungo, anche se necessario per raggiungere gli standard qualitativi richiesti dal cliente. In tutto ciò, la VR può essere una soluzione molto conveniente dal punto di vista economico e del tempo di prototipazione, e per tale ragione sta gradualmente sostituendo, laddove possibile, il processo di prototipazione tradizionale.

In altri casi, il costo e il tempo di produzione non sono gli unici fattori determinanti nella scelta del tipo di processo da adottare, ma anche la sicurezza di ingegneri, operai e del personale dell'azienda e/o istituzione in generale. La VR, essendo per natura un ambiente simulato, elimina molti dei rischi connessi alla produzione di macchinari potenzialmente pericolosi. Infatti, un operaio può essere formato in ambiente virtuale prima di entrare a contatto con il macchinario vero e proprio; invece, in fase di produzione, alcune operazioni possono essere eseguite in ambiente virtuale per prevenire eventuali complicazioni non previste, prima di essere effettuate sul macchinario reale.

Sfogliando il catalogo di un negozio online è spesso possibile vedere i prodotti venduti, da diverse angolazioni, scorrendo le foto messe a disposizione del consumatore, sia che si tratti di un negozio di elettronica, sia che si tratti di un negozio di abbigliamento. Un modo alternativo di valutare un prodotto in vendita, almeno dal punto di vista estetico, sarebbe possibile sfruttando la VR; ogniqualvolta l'utente seleziona un prodotto, un modello 3D dello stesso viene scaricato automaticamente dalla rete, e l'utente può osservarlo in tre dimensioni tramite un visore VR [7], come se lo avesse di fronte a sé in un negozio fisico.

Nella moda e nell'industria dei capi d'abbigliamento in generale, la capacità di visualizzazione permessa dalle tecnologie di realtà mista può essere uno strumento utilissimo non solo in fase di vendita, ma anche in fase di produzione; si pensi, ad esempio, alla possibilità per lo stilista di valutare la sua nuova idea direttamente su un modello virtuale.

Militare

In campo militare, la realtà virtuale è impiegata come strumento per l'addestramento; essa è particolarmente utile per addestrare i soldati ad affrontare in modo appropriato scenari di conflitto e situazioni inusuali e pericolose, senza incorrere il rischio di essere feriti gravemente o morire.

Rispetto a metodi di addestramento tradizionali, la simulazione in VR è meno costosa e non mette a rischio l'incolumità dei soldati. I dispositivi comunemente impiegati dai soldati nelle sessioni di addestramento sono head-mounted display e data-glove, entrambi tracciati per permettere l'interazione con gli oggetti virtuali.

Attività di addestramento tipiche includono

1. Addestramento del personale medico.
2. Simulazione di conflitto armato.
3. Simulazione di volo per piloti.
4. Simulazione di guida.

La realtà virtuale può essere d'aiuto nel trattare i sintomi causati dal disordine da stress post-traumatico (PTSD), una condizione che accomuna molti veterani che hanno sofferto dei traumi o condizioni psicologiche particolarmente difficili sul campo di battaglia. Per insegnare ai pazienti a gestire i sintomi del PTSD, questi sono sottoposti a situazioni che scatenino in modo graduale il disturbo, cosicché possano imparare a controllarlo, in un ambiente che percepiscono sicuro.

Capitolo 3

Framework

In questo capitolo è presentato il lavoro svolto nell'ambito della presente tesi, partendo dai requisiti di progetto, proseguendo con la descrizione del framework e dei suoi componenti. Infine, saranno esaminati separatamente le parti relative al server e al client.

Poiché il client è stato pensato per poter essere eseguito su diversi dispositivi, ha un grado di complessità maggiore del server, per tale ragione sarà analizzato dapprima quest'ultimo.

3.1 Requisiti

Questa tesi nasce dall'idea di realizzare applicazioni in cui utenti VR e AR possano condividere la stessa esperienza, anche se in un contesto diverso. A tal fine, si è pensato di realizzare un framework che agevolasse lo sviluppo di nuove applicazioni.

Tale framework dovrebbe permettere di aggiungere un nuovo insieme di regole, che definiscono la modalità di funzionamento del software, e nuove entità, che costituiscono gli oggetti con cui gli utenti possono interagire, senza richiedere alcun intervento[1] da parte del programmatore sul codice preesistente.

Data l'enorme popolarità, sia in ambito ludico, che, soprattutto, in ambito industriale e accademico, la scelta dell'engine su cui il Framework si appoggia è ricaduta su Unity.

3.1.1 SDK per AR e VR

Poiché le applicazioni devono supportare sia i dispositivi VR che quelli AR, si è deciso di utilizzare il Mixed Reality Toolkit v2 (MRTK), un toolkit open-source sviluppato da Microsoft.

[1] Oppure un numero ridotto, nel caso in cui il programmatore debba apportare modifiche al framework

Prima di descrivere nel dettaglio l'MRTK, si presenteranno brevemente le alternative prese in considerazione.

OpenVR (SteamVR) OpenVR è il software development kit (SDK) alla base di SteamVR, la piattaforma per la realtà virtuale sviluppata da Valve. OpenVR supporta nativamente il visore HTC Vive, e indirettamente, tramite il *runtime* di Oculus, il visore Oculus Rift e i relativi motion controller, l'Oculus Touch.

Unity supporta, tramite *plug-in*, OpenVR e SteamVR.

Oculus SDKs L'Oculus PC SDK permette lo sviluppo di applicazioni destinate unicamente ai visori Oculus per PC, tra cui il Rift e il Rift S, mentre l'Oculus Quest Native SDK è appositamente pensato per l'Oculus Quest, un visore *all-in-one* che non richiede un PC per funzionare.

Come SteamVR, l'Oculus SDK è disponibile per Unity sotto forma di plug-in.

Mixed Reality Toolkit v2

Il Mixed Reality Toolkit (MRTK) nasce come evoluzione dell'HoloToolkit, un toolkit pensato per lo sviluppo di applicazioni AR su dispositivo Microsoft HoloLens. Al contrario del predecessore, l'MRTK, attualmente disponibile solo per Unity, permette la costruzione di applicazioni destinate non solo all'HoloLens (e HoloLens 2), ma anche a dispositivi VR, sia quelli della famiglia del Windows Mixed Reality, che quelli supportati da OpenVR (notoriamente, Oculus Rift e HTC Vive).

Funzionalità L'MRTK copre molti degli aspetti di un'applicazione per realtà mista, tra i quali la gestione degli input (tramite motion controller, gesti, voce e sguardo), sistema di tracciamento (*Spatial Mapping* di HoloLens e *eye tracking* di HoloLens 2), interfacce 3D (include pulsanti 3D, check-box, slider, etc.), sistemi di locomozione per VR (ad esempio il teletrasporto) e boundary-system (cioè definizione del volume di gioco).

Architettura Il toolkit è stato concepito come una struttura modulare (si veda la figura 3.1), in cui un dato componente o modulo può essere rimosso e sostituito da un modulo custom. Ogni modulo è, inoltre, configurabile tramite un profilo dedicato (si veda la figura 3.2).

Gestione degli Input Analogamente ad altri SDK (ad esempio SteamVR) e toolkit, gli input sono gestiti in modo da assicurare la *forward-compatibility*[2] dell'applicazione. Piuttosto che legare la logica dell'applicazione agli input di un particolare dispositivo, richiedendo in tal modo che l'applicazione sia aggiornata per supportare ogni dispositivo futuro, il toolkit fa uso del concetto di *azione*; un azione può essere generata, ad esempio, premendo il tasto di un controller. Legando la logica alle azioni, invece che agli input,

[2]Compatibilità dell'applicazione con i dispositivi non ancora presenti sul mercato

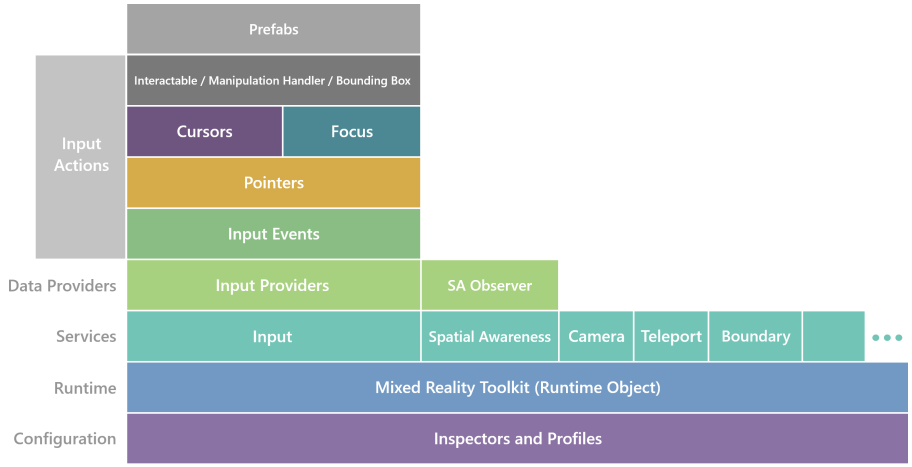

Figura 3.1. Architettura del Mixed Reality Toolkit.

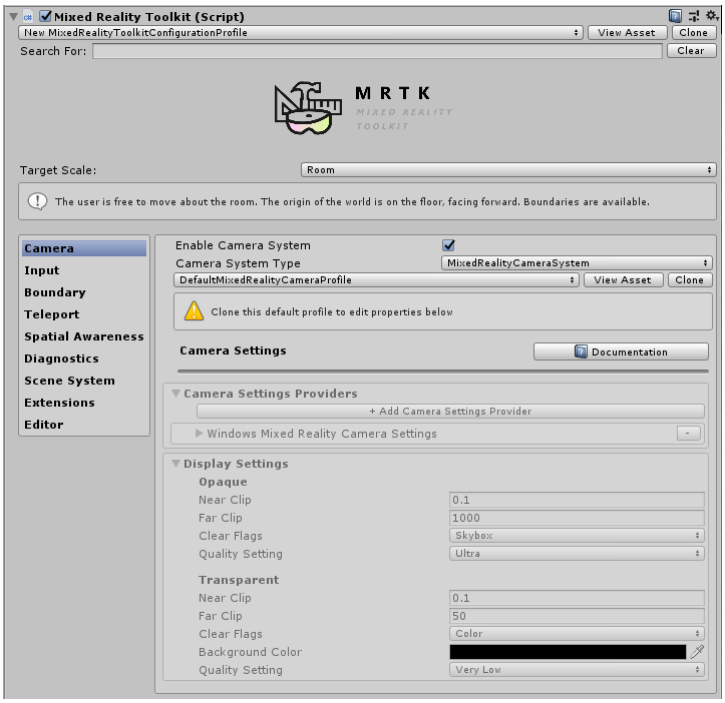

Figura 3.2. Il profilo principale del Mixed Reality Toolkit.

la compatibilità in avanti dell'applicazione sarà garantita dal fatto che i runtime su cui si appoggia il toolkit saranno aggiornati per supportare i nuovi dispositivi; il runtime provvederà, quindi, a tradurre gli input dei nuovi dispositivi nelle azioni che l'applicazione è in grado di comprendere.

3.1.2 Protocollo di rete

Si discuteranno, ora, i protocolli di rete UDP e TCP, e si tenterà di giustificare la scelta di un protocollo rispetto all'altro.

User Datagram Protocol

User Datagram Protocol (UDP) è un protocollo di rete progettato da David P. Reed nel 1980. Un software eseguito su una data macchina può fare uso di UDP per inviare pacchetti, chiamati *datagram*, ad altri host della rete IP.

Le caratteristiche di UDP sono qui riportate:

a) Stateless: è adatto quando il numero di host è elevato.

b) Semplicità.

c) Supporta il *multicast*.

d) Nessuna forma di ritrasmissione (quindi velocità): è adatto per quei servizi in cui è tollerabile la perdita di pacchetti e, soprattutto, è importante la velocità di trasmissione dei dati.

Intestazione L'intestazione del pacchetto UDP contiene 4 campi di 2 byte:

1. Source port number (opzionale in IPv4 e IPv6); identifica la porta del mittente.
2. Destination port number; identifica la porta del destinatario.
3. Length; lunghezza dell'header e dei dati.
4. Checksum (opzionale in IPv4); checksum per controllo sugli errori.

Transmission Control Protocol

Transmission Control Protocol (TCP) è, assieme a UDP, uno dei principali protocolli dell'*Internet protocol suite*. TCP era inizialmente chiamato modello Department of Defense (DoD), in quanto il suo sviluppo era finanziato dal Dipartimento della Difesa americano.

TCP è un protocollo del livello trasporto del modello ISO/OSI e, contrariamente a UDP, è orientato alla connessione e di tipo *reliable* (affidabile), cioè garantisce che il flusso di segmenti inviati dal mittente raggiunga interamente e in modo invariato il destinatario. In particolare, TCP da le seguenti garanzie:

a) Se il segmento A è inviato prima del segmento B, il destinatario riceverà il segmento A prima del segmento B; più correttamente, il destinatario vede un flusso di dati. Se il segmento B arriva all'host prima del segmento A, l'host attenderà di ricevere il segmento A, e, una volta ricevuto, presenterà all'applicazione i due segmenti correttamente ordinati.

b) Un segmento arriva sempre a destinazione (ammenoché la connessione non si sia interrotta): se un segmento non arriva a destinazione perché è stato gettato via da un hop, il meccanismo di controllo di TCP fa sì che il segmento sia reinviato dal mittente, finché non giunge a destinazione. Per confermare che il segmento è arrivato a destinazione, il destinatario invia un pacchetto di controllo detto *Ack*.

c) Un segmento arriva integro a destinazione: il campo *checksum* permette di rilevare eventuali errori, in tal caso il segmento viene reinviato.

UDP vs TCP

Per applicazioni di natura generale, è tipicamente preferibile fare uso di un protocollo di rete affidabile come TCP, in quanto la certezza della ricezione di un pacchetto e la semplicità dell'applicazione sono più importanti dell'avere un tempo di trasmissione minore.

Si consideri il caso di un'applicazione multiutente che richieda bassa latenza, come un videogioco competitivo o un'applicazione interattiva VR o AR; se il protocollo di rete su cui si regge il *netcode*[3] dell'applicazione fosse di tipo reliable, i meccanismi di controllo del flusso di pacchetti descritti nella sezione su TCP rallenterebbero la comunicazione tra le parti in gioco, rendendo l'esperienza incostante. Questa è la ragione per la quale tutti i videogiochi competitivi multigiocatore sono basati su UDP, ed è anche la ragione per la quale si è scelto di basare su tale protocollo il netcode del framework qui proposto.

3.1.3 Netcode: Peer-to-Peer vs Client-Server

Client-Server

Nell'architettura Client-Server (mostrata in figura 3.3), il server è il luogo in cui sono processati gli input degli utenti e avviene la maggior parte del processing relativo all'aggiornamento delle entità.

Ogni client si limita a registrare gli input dell'utente e a inviarli periodicamente al server; una volta ottenuto lo stato aggiornato delle entità, la scena è processata dalla GPU per essere visualizzata a schermo.

Questa architettura ha i seguenti pregi:

a) È scalabile dal punto di vista dei client; il numero di client non influisce sul client stesso, ma sul server, che ha bisogno di più banda per servire tutti i client (Tipicamente si usano dei server dedicati che hanno a disposizione la banda necessaria per l'applicazione che gira su tali macchine).

b) Il processing sul client è tipicamente basso, in quanto i calcoli avvengono sul server; in funzione dell'applicazione, è possibile che alcuni calcoli non trascurabili dal punto di vista computazionale sia eseguiti sui client.

[3] Quella parte di codice dedicata allo scambio di pacchetti, alla sincronizzazione dello stato, etc.

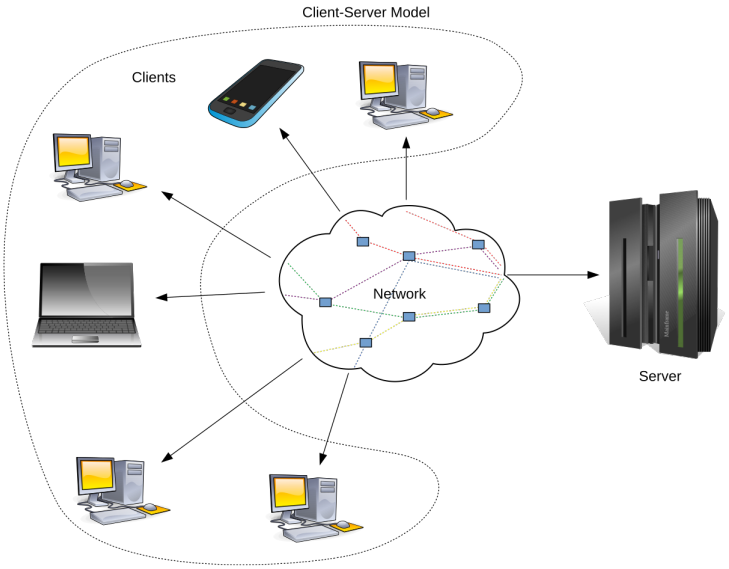

Figura 3.3. Nel modello Client-Server il server comunica con tutti i client, ma i client non comunicano fra loro.

c) È più semplice da implementare rispetto all'architettura Peer-to-Peer, quando il numero di utenti è maggiore di 2.

d) Essendo un'architettura per natura centralizzata, permette di svolgere dei controlli sulla correttezza dello stato delle entità; nei videogiochi, è preferibile al Peer-to-Peer per combattere il cheating.

e) È più robusta a condizioni non ideali della rete.

Peer-to-Peer

Il modello Peer-to-Peer (illustrato in figura 3.4) vede i client comunicare direttamente per scambiarsi i pacchetti contenenti lo stato del mondo virtuale, che non è più gestito in modo centralizzato da un server dedicato.

Il vantaggio principale del modello in esame, rispetto al modello client-server, è che non è più necessario un server dedicato, che deve essere messo a disposizione dallo sviluppatore dell'applicazione o da terze parti con cui lo sviluppatore ha stipulato un contratto.

Gli svantaggi, invece, sono molteplici:

a) Per applicazioni con più di 2 client, l'implementazione del modello è più complessa.

b) Scarsa scalabilità con il numero di client: ogni client deve inviare le stesse informazioni a tutti gli altri client e, tipicamente, a differenza di un server dedicato, non ha banda a sufficienza.

c) Un client deve conoscere l'indirizzo IP di ogni altro client.

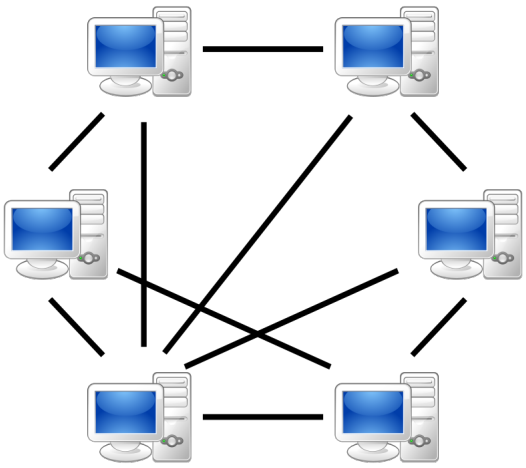

Figura 3.4. Nel modello Peer-to-Peer, un client può comunicare con ogni altro client.

Per le ragioni esposte, la scelta è ricaduta sul modello client-server, in particolare per tre ragioni:

1. Esigenza di supportare applicazioni multiutente con più di 2 client.
2. Alcuni dei dispositivi che saranno usati come client hanno scarsa potenza e banda limitata[4].
3. Più semplice da implementare.

3.1.4 Architettura Client-Server

In questa sezione si discuterà più dettagliatamente il funzionamento dell'architettura Client-Server.

Lag

Nell'ambito delle applicazioni di rete, con il termine *lag* si intende il ritardo nella risposta del sistema all'input dell'utente. La causa del lag è legata non solo alla distanza tra il

[4] Ad esempio HoloLens, o, se saranno supportati in futuro, gli smartphone

luogo in cui l'input è processato (il server) dal luogo in cui il risultato è mostrato (il client), ma anche al modo in cui funziona la rete *internet*.

Per capire a cosa è dovuto il lag, si consideri il caso di un videogioco multigiocatore online, in cui il giocatore controlla un personaggio giocante (PG) che può muovere utilizzando le frecce direzionali della tastiera. Si assuma che all'istante t_0 il giocatore prema la freccia *uparrow* per indicare al PG di muoversi in avanti; il comando generato non viene eseguito sul client, ma sul server.

Essendo la rete composta da un insieme di nodi che scambiano i pacchetti in funzione del traffico di rete, il tempo che il comando, contenuto in un pacchetto, impiega per raggiungere il server non è costante (per la stessa ragione, la risposta del server impiegherà un tempo variabile per raggiungere il client).

Supponiamo che il server riceva il comando dopo 50 ms[5], all'istante t_1. Il server processa il comando e muove il PG, quindi invia il risultato al client, che lo riceverà all'istante t_2, con un ritardo che possiamo ancora una volta supporre di 50 ms. Il giocatore, perciò, vedrà il PG muoversi ben 1/10 di secondo dopo aver dato il comando.

Nelle sezioni che seguono saranno descritte le tecniche utilizzate per permettere ad utenti che si trovano a centinaia di km di distanza di condividere la stessa esperienza, sia ludica che non, senza avvertire tale ritardo.

Predizione lato client

Il primo problema che si vuole risolvere è eliminare il ritardo che l'utente locale percepisce quando impartisce un comando, in modo da fargli percepire quel senso di immediatezza senza il quale non potrebbe godersi l'esperienza.

Nella spiegazione sul funzionamento dell'architettura client-server, si è affermato che il compito (e l'autorità) dell'aggiornamento delle entità appartiene al server; supponiamo, per il momento, di permettere al client di eseguire quei calcoli necessari per far muovere il PG dell'esempio di sopra. Poiché il tempo necessario per il processamento dell'input dell'utente e per la visualizzazione del frame video si misura in pochi millisecondi, l'utente percepisce la risposta come istantanea.

A questo punto, però, sorge un nuovo problema: come si gestisce il fatto che è il server ad avere l'autorità sui client?

Riconciliazione tra client e server

Lo stesso comando che è stato eseguito sul client per far muovere il PG arriverà al server con un ritardo variabile in base alla distanza tra le parti e il traffico di rete; quindi il server lo eseguirà a sua volta, ottenendo un risultato che molto probabilmente coincide

[5] 50 ms può essere considerato un buon valore, visto che ritardi di 100 o 200 ms sono comuni

con quello prodotto dal client[6].

Il server, quindi, invia al client la risposta contenente la corretta posizione del PG; quando il client riceve il valore calcolato dal server, lo confronta con il valore che esso stesso ha calcolato. Due casi si possono verificare:

a) I due valori coincidono: il client non effettua nessuna nuova operazione.

b) Il valore prodotto dal client differisce da quello ricevuto dal server: il client deve "gettare via" il suo valore e assegnare al PG il valore di posizione corretto. Tale operazione è detta *riconciliazione*[7].

Interpolazione delle entità

Si consideri, adesso, un altro aspetto non necessariamente connesso al lag, ma al modo in cui vengono processati gli input da parte del server: siccome i pacchetti inviati dai client sono molto frequenti, piuttosto che processare ogni comando separatamente consumando molto tempo-CPU e banda per inviare il risultato ai client, il server *bufferizza*[8] gli input che gli giungono dai client e li processa periodicamente tutti in una volta a bassa frequenza, 10 o 20 volte al secondo.

A 10 volte al secondo, il periodo di ricezione dei pacchetti da parte dei client è di 100 ms, ciò comporta che l'utente vedrebbe le entità virtuali muoversi a bassa frequenza, con un'evidente percezione di scattosità non legata alla pesantezza grafica della scena, ma alla bassa "frequenza di campionamento".

La soluzione a tale problema consiste nel presentare all'utente l'ultima informazione (cioè lo stato delle entità) arrivata dal server come se fosse riferita al futuro, assegnando a una data variabile un valore ottenuto interpolando i valori dello stato passato e dello stato futuro.

L'ovvio svantaggio è che l'utente vedrà sempre il passato e non lo stato più aggiornato del mondo virtuale, ma l'applicazione guadagnerà dal punto di vista della fluidità percepita.

Compensazione del lag

L'ultima tecnica descritta nella presente sezione serve a risolvere un problema introdotto dalla tecnica dell'interpolazione; si pensi a un gioco appartenente al genere sparatutto, in cui i giocatori si sparano a vicenda per ottenere dei punti.

---

[6] Quando il client bara, è molto alta la probabilità che i due valori non coincidano

[7] Adeguandosi al *diktat* del server, il client si riconcilia con il server

[8] Raccoglie e mette da parte per l'uso futuro

Supponiamo che il giocatore player_a veda di fronte a sé, proprio nel suo mirino, il giocatore player_b; in realtà, per effetto dell'interpolazione, egli vede dove si trovava player_b in un certo istante del passato.

Player_a preme il tasto per sparare, inviando il comando corrispondente al server, che lo riceve in un certo istante successivo, a causa del lag. Supponiamo che il comando contenga il tempo in cui è stato eseguito, in tal modo il server sa dove si trovavano *effettivamente* le entità al momento dello sparo. Il server processa lo sparo, ma poiché, come detto sopra, player_a vede il passato, molto probabilmente player_b si trovava da un'altra parte al momento dello sparo, e il colpo conseguentemente non andrà a segno.

Il gioco sarebbe ingiocabile, in quanto i giocatori non riuscirebbero a portare a segno un colpo il più delle volte.

La soluzione è molto semplice: poiché il server sa che i client eseguono un'interpolazione della posizione delle entità, esso stesso può eseguire l'interpolazione al momento dell'esecuzione del comando, calcolarne l'esito, e infine riposizionare le entità nella posizione corretta.

Sebbene le tecniche appena illustrate causino incoerenze nell'esperienza di gioco, sono comunque preferibili all'esperienza che si otterrebbe senza di esse.

Input in contesto VR/AR e client autoritativo

In una classica applicazione desktop, ad esempio un videogioco multiplayer di genere sparatutto, il giocatore genera degli input tramite il mouse e la tastiera, che possono essere inviato al server e processati. Ad esempio, il giocatore preme il tasto *W*, che è comunemente associato al movimento in avanti; il server calcola il movimento e invia il risultato al client, che si limita ad assegnare la nuova posizione.

Si pensi, per contrasto, al caso di un sistema VR: la posizione e la rotazione del casco VR sono tracciati dal sistema di tracciamento; non esiste, quindi, un input o comando, ma solo il risultato[9].

Nelle applicazioni VR e AR, l'utente controlla direttamente la camera usata nel rendering, cioè la vista sulla scena virtuale. Muovendo la testa, l'utente muove la camera, che deve sempre seguire perfettamente il movimento della testa; infatti, in caso contrario, l'utente potrebbe avvertire i sintomi negativi legati a nausea, o altri effetti spiacevoli.

Si supponga, adesso, di applicare al caso VR lo stesso meccanismo del server autoritativo di cui sopra: l'utente fa un passo in avanti, inviando al server l'informazione che la testa si è spostata di 20 cm in avanti. Il server, fatti i suoi calcoli, determina che la testa non può spostarsi di 20 cm, ma solo di 10 cm[10].

[9] Cioè la posizione finale

[10] Ad esempio, perché è presente un *collider* di fronte all'avatar

Nel client, la camera si muoverà, in un primo momento, di 20 cm in avanti, in modo concorde con il movimento "fisico" dell'utente; però, non appena il client riceve il valore di posizione dal server, la camera sarà spostata 10 cm indietro per adeguarsi a ciò che il server impone.

In tale circostanza, l'utente vedrebbe un'accelerazione improvvisa della scena virtuale, mentre continua a sentire, fisicamente, di non essersi spostato. Ciò causerà effetti indesiderati sull'utente.

Le considerazioni appena esposte hanno condotto alla scelta di adottare un approccio misto, in cui il server ha l'autorità su tutto, eccetto che sulla posizione e rotazione di quelle parti del corpo che sono tracciate, indipendentemente dal tipo di tecnologia impiegata per tracciarle.

3.2 Moduli e componenti

Il framework è composto da due parti software, il server e il client, che presentano una struttura quasi simmetrica, come illustrato nella figura 3.5.

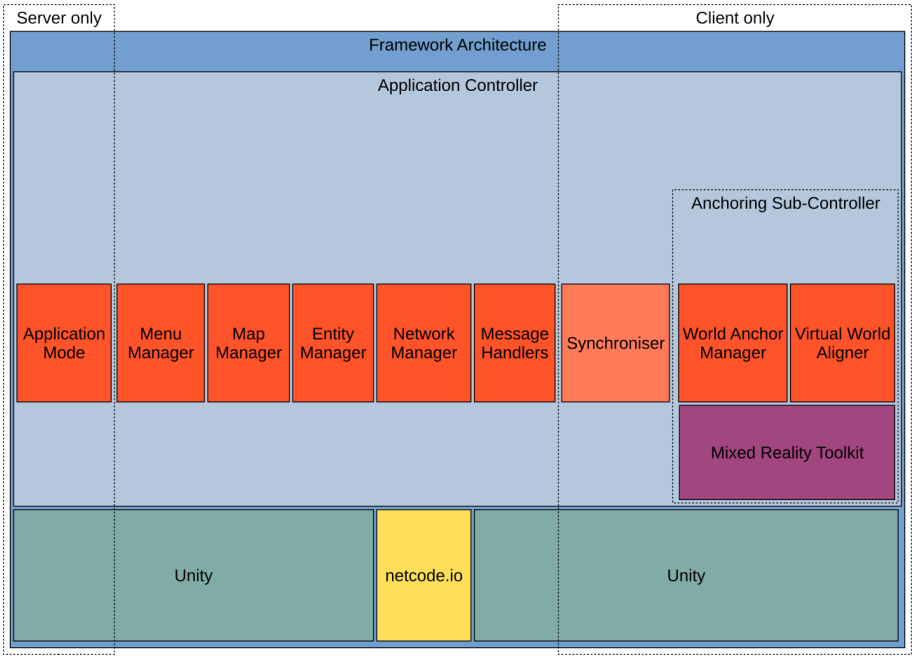

Figura 3.5. Rappresentazione schematica del framework.

Moduli e sistemi sono coordinati da un *Application Controller*, che ha il compito di assicurarsi il corretto funzionamento dei moduli, permettendo loro di comunicare e scambiarsi dati.

I moduli, che dal punto di vista software sono classi C#, presentano due versioni distinte per il server e il client, ad eccezione del Map Manager e di altri dedicati al solo client o al solo server. L'elenco dei moduli è di seguito riportato:

1. Network Manager;
2. Content Message Handler;
3. Ping Message Handler;
4. Entity Manager;
5. Map Manager;
6. Menu Manager;
7. World Anchor Manager;
8. Virtual World Aligner;

3.2.1 Network Manager

Come suggerisce il nome, tale modulo si occupa della gestione degli aspetti legati alla rete, in particolare esso entra in gioco ogniqualvolta deve essere instaurata una connessione di rete tra il client e il server, oppure una parte, sia essa un client o il server, deve trasmettere o ricevere un pacchetto di rete.

Il Network Manager, basato sul protocollo *netcode.io* (una descrizione del protocollo è disponibile nell'appendice C), è stato progettato attorno a un'architettura di tipo client-server, in cui i client non si scambiano pacchetti, ma comunicano indirettamente per mezzo del server.

Pool di buffer La classe C# NetworkManager contiene una lista statica di array di byte, che ha la funzione di pool da cui attingere ogniqualvolta si vuole inviare o si riceve un nuovo pacchetto; ciò riduce gli effetti negativi connessi all'intervento del *garbage collector*, permettendo al programmatore di gestire nel miglior modo possibile la memoria allocata.

Per accedere a suddetto pool si chiama la funzione statica GetBufferFromPool(int capacity, out byte[] buffer), che restituisce un buffer avente capacità almeno uguale al valore del parametro capacity. Analogamente, per rilasciare un buffer acquisito in precedenza, si chiama la funzione statica ReleaseBuffer(byte[] buffer). Infine, l'accesso alla lista interna è protetta tramite lock, in quanto l'applicazione fa uso di più di un thread.

Classe ServerNetworkManager

La classe C# ServerNetworkManager, che deriva da NetworkManager, è costruita attorno alla classe NetcodeIO.Net.Server, e ha la funzione di semplificare la gestione del server sottostante. In particolare, il ServerNetworkManager acquisisce le informazioni (eventi e pacchetti di rete) che il thread dedicato al server fornisce attraverso degli opportuni eventi

(OnClientConnected, OnClientDisconnected e OnClientMessageReceived), e di accedervi dal thread principale.

Ogniqualvolta un evento o pacchetto di rete è disponibile, tale informazione è aggiunta in una lista, protetta da *lock* e avente la funzione di buffer. Nel metodo Update() del ServerNetworkManager, tale lista è acceduta e le informazioni presenti sono processate nell'ordine di arrivo.

L'interfaccia pubblica del modulo in esame presenta, fra gli altri, un metodo per avviare il server e uno per interromperlo, un metodo per terminare la connessione con un client, e un metodo per trasmettere un pacchetto di rete (specificando il buffer contenente i dati, la dimensione dei dati e l'Id del destinatario).

Inoltre, sono disponibili tre eventi a cui è possibile iscriversi:

a) OnClientConnectedEvent: un nuovo client si è connesso al server.

b) OnClientDisconnectedEvent: uno dei client connessi si è appena disconnesso.

c) OnPacketReceivedEvent: è stato ricevuto un nuovo pacchetto da un client.

Classe ClientNetworkManager

Il ClientNetworkManager, come il ServerNetworkManager, deriva da NetworkManager e ha la funzione di wrapper attorno alla classe NetcodeIO.Net.Client. Il meccanismo per l'acquisizione e il processamento dei pacchetti di rete descritto per il ServerNetworkManager è qui impiegato.

I metodi esposti all'esterno permettono di connettersi al server, chiudere una connessione attiva, e inviare un pacchetto di rete.

Procedura di connessione Poiché il metodo Connect() per creare la connessione con il server non garantisce che la connessione sia attiva nel momento in cui la funzione ritorna, la procedura per connettersi al server consiste nei passi seguenti:

1. Chiamata del metodo Connect() specificando una callback per il caso in cui la connessione sia instaurata con successo, e un'altra callback per il caso in cui il tentativo di connessione fallisca.

2. Sia che la connessione abbia successo, sia che fallisca, la chiamata della callback corrispondente permetterà al ClientNetworkManager di segnalare all'ApplicationController l'esito del tentativo di connessione, e a quest'ultimo di agire in modo opportuno.

I casi di disconnessione e ricezione di un pacchetto di rete sono gestiti tramite eventi:

a) OnDisconnectedEvent: il client si è disconnesso, ad esempio per un guasto alla rete.

b) OnPacketReceivedEvent: è disponibile un nuovo pacchetto inviato dal server.

3.2.2 Content Message Handler

Il Content Message Handler prende il nome dal fatto che esso gestisce i messaggi non di controllo, aventi cioè un carico (payload) o contenuto di interesse per elementi terzi. Infatti, possiamo distinguere i messaggi di rete in messaggi di controllo, ad esempio l'ack, dai messaggi che hanno un contenuto informativo a un livello più alto per una o più parti del sistema.

Siccome netcode.io, su cui è costruito il sistema di networking, è basato su UDP, non esiste alcun controllo né sulla perdita di pacchetti, né sull'ordine di arrivo, né, tantomeno, sull'integrità del dato ricevuto.

Benché la scelta di un protocollo non affidabile sia giustificata dal tipo di applicazione per cui il framework è stato pensato, ci sono situazioni in cui è preferibile avere la certezza che il pacchetto inviato arrivi a destinazione, pertanto è stato costruito, con il Content Message Handler, un layer che permette di inviare, a richiesta dell'applicazione, un pacchetto in modo affidabile.

Classe NetworkMessage

Al fine di introdurre il layer di affidabilità, è stata definita una nuova classe, chiamata NetworkMessage, che arricchisce il pacchetto di rete con un'intestazione contenente quei campi necessari all'identificazione del pacchetto.

L'intestazione include i seguenti campi:

a) Id: numero che identifica il messaggio di rete.

b) MessageType: tipo del messaggio; può essere *Unreliable*, *Reliable*, *Ack*, *Ping* e *PingAck*.

c) Size: dimensione del payload.

d) IsFragment: indica se il messaggio è frammentato.

e) FragmentNumber: se il messaggio è frammentato, indica il numero del frammento.

f) FragmentCount: se il messaggio è frammentato, indica il numero di frammenti.

La classe NetworkMessage espone anche dei metodi per la scrittura e lettura del buffer interno, del tipo illustrato di seguito:

```
public void Write(bool value)
{
    // ...
}

public void Read(out bool value)
{
    // ...
}
```

Frammentazione dei messaggi

Il Framework è stato progettato con l'obiettivo di minimizzare la dimensione dei pacchetti di rete scambiati tra i diversi attori, e la stessa libreria su cui il networking è costruito, netcode.io (C), ha un limite massimo per la dimensione dei pacchetti pari a 1200 byte[11].

Ciononostante, può verificarsi che il messaggio di rete superi il limite dei 1200 byte (a causa, ad esempio, dell'annidamento dei messaggi che discusso nel capitolo 3.3.4), perciò è stata implementata una forma di frammentazione per i messaggi di tipo Unreliable.

Metodi del ContentMessageHandler

Come si è detto, il ContentMessageHandler, che si appoggia al NetworkManager per trasmettere e ricevere i pacchetti di rete, gestisce i messaggi di tipo Content, cioè quelli Reliable e Unreliable (e Ack di pacchetti Reliable), mentre Ping e PingAck sono gestiti da un altro modulo, il PingMessageHandler, discusso in 3.2.3.

L'interfaccia esposta all'esterno include i due metodi per inviare messaggi di rete:

a) SendUnreliableMessage: invia un messaggio in modo non affidabile.

b) SendReliableMessage: invia un messaggio in modo affidabile; opzionalmente è possibile passare al metodo due funzioni di callback, che vengono chiamate l'una quando arriva la conferma di avvenuta ricezione del messaggio da parte del destinatario, mentre l'altra se il messaggio non arriva a destinazione entro il tempo prefissato.

Messaggio di applicazione

A livello di applicazione, con messaggio si intende un gruppo di classi che derivano dalla classe C# ApplicationMessage; un ApplicationMessage è, per l'appunto, un'informazione che gli ApplicationController del server e dei client si possono scambiare per comunicare.

Poiché ogni messaggio deve essere scambiato attraverso la rete, ApplicationMessage contiene un campo *enum* chiamato Type, che permette di ricostruire il messaggio originale da un flusso di byte. Inoltre, sono presenti un costruttore che permette di ottenere il messaggio da un pacchetto di rete e, viceversa, un metodo virtuale ToNetworkMessage() per salvare il messaggio in un pacchetto di rete.

```
public abstract class ApplicationMessage
{
    public enum Type : byte
    {
        // ...
    }

    public Type MessageType { get; protected set; } = Type.None;
```

[11] Il valore è legato al *maximum transmission unit*

```
    public ApplicationMessage() { }

    public ApplicationMessage(NetworkMessage networkMessage)
    {
        // ...
    }

    public virtual void ToNetworkMessage(NetworkMessage
        networkMessage)
    {
        // ...
    }
}
```

Ogni altro messaggio di applicazione che derivi da ApplicationMessage espone un costruttore che prende come parametro un NetworkMessage, e ridefinirà tramite *override* il metodo ToNetworkMessage().

Gli ApplicationMessage possono essere raggruppati in due sottogruppi in base al mittente:

a) Messaggi inviati dal server: SessionDescription, UserInfoUpdate, StartSessionInfo, Snapshot, ServiceInfo, SessionResults.

b) Messaggi inviati dal client: ClientPresentation, ClientReady, PlaytimeInfo, ClientResearchInfo.

Si darà ora una breve descrizione dei messaggi di applicazione, in ordine cronologico relativamente alla vita dell'applicazione.

ClientPresentation Il messaggio ClientPresentation è inviato dal client al server dopo aver instaurato la connessione, al fine di fornirgli tutte le informazioni necessarie per la gestione del client stesso.

Il messaggio contiene un unico campo, ClientInfo, che include le seguenti informazioni:

a) Name: è il nome della macchina su cui è eseguita l'applicazione client.

b) PlatformType: tipo di piattaforma, o dispositivo, usato dal client; può assumere i valori *VirtualReality* o *AugmentedReality*.

c) InputScheme: è un enum che indica lo schema di input scelto dal client; può essere singolo tasto o multi-tasto.

d) Handedness: la mano dominante dell'utente.

e) Height: l'altezza in metri dell'utente; inserita manualmente dall'utente.

SessionDescription Il messaggio SessionDescription è inviato dal server al client in seguito all'avvenuta ricezione del messaggio ClientPresentation, e, come suggerisce il nome, contiene la descrizione della sessione, cioè le regole della modalità, i parametri relativi allo scambio di messaggi di sessione (Snapshot e PlaytimeInfo), il nome della mappa, e la posizione iniziale dell'avatar dell'utente.

Contiene i seguenti campi:

a) ApplicationModeRules: è una stringa rappresentante le regole della modalità.

b) SnapshotPeriod: è il tempo medio, espresso in microsecondi, tra l'invio di uno Snapshot e il successivo.

c) PlaytimeInfoPeriod: è il tempo medio, espresso in microsecondi, tra l'invio di uno PlaytimeInfo e il successivo.

d) MapName: è il nome della mappa.

e) UserStartPosition: è la posizione iniziale dello UserActor controllato dal client a cui il messaggio è stato trasmesso.

f) UserStartRotation: è la rotazione iniziale dello UserActor controllato dal client a cui il messaggio è stato trasmesso.

UserInfoUpdate Dopo che un client si è connesso al server, tra questo momento e l'inizio della sessione altri client possono unirsi alla sessione. In questo lasso di tempo, il server invia periodicamente il messaggio UserInfoUpdate a tutti i client, per aggiornarli circa gli altri client connessi.

Oltre alle informazioni sui client, il messaggio contiene anche il numero di utenti che sono pronti a iniziare la sessione.

ClientReady ClientReady non contiene alcun campo, avendo il solo scopo di segnalare al server che il client che ha inviato il messaggio è pronto a iniziare la sessione.

StartSessionInfo Dopo aver ricevuto il ClientReady da tutti i client connessi, il server avvia la sessione, segnalandolo ai client attraverso il messaggio StartSessionInfo.

Esso contiene due campi:

a) UserInfos: è un dizionario contenente le informazioni sui client.

b) EntityDefinitions: si tratta di un dizionario contenente le definizioni delle entità che il client dovrà creare.

Snapshot Lo Snapshot è inviato periodicamente dal server ai client durante tutta la sessione e contiene lo stato delle entità, assieme ad altre informazioni necessarie per la sincronizzazione tra server e client e il corretto funzionamento del client stesso.

Il server genera uno Snapshot diverso per ogni client, tenendo conto dello stato corrente del client che esso stesso ha comunicato al server mediante messaggio PlaytimeInfo.

Ogni Snapshot è identificato da un numero intero crescente, con il primo Snapshot della sessione avente Id uguale a zero. Il client, confrontando l'Id del messaggio ricevuto con l'ultimo valore registrato, è in grado di determinare se il messaggio ricevuto è successivo, oppure se è un messaggio precedente, che è arrivato fuori ordine per il modo in cui funziona la rete.

L'elemento principale di uno Snapshot è l'immagine del mondo, o WorldImage, che raccoglie lo stato di tutte le entità, la lista delle entità distrutte, e un dizionario con le definizioni delle nuove entità che il client dovrà creare.
La WorldImage è detta cumulativa, nel senso che è stata ottenuta fondendo le immagini relative a diversi frame in una sola immagine (si veda 3.3.4).

I campi sono di seguito riportati:

a) Id: identifica il messaggio.

b) CumulativeWorldImageFirstId: la WorldImage cumulativa include tutte le WorldImage con Id da CumulativeWorldImageFirstId fino alla più recente.

c) CumulativeWorldImage: è l'immagine cumulativa.

d) LastReceivedPlaytimeInfoId: l'ultimo PlaytimeInfo che il server ha ricevuto dal client a cui questo Snapshot è destinato.

e) RoundTripTime: Rount-Trip Time misurato dal server per il client.

PlaytimeInfo Analogamente al server, il client invia periodicamente un messaggio al server per aggiornarlo sul suo stato; tale messaggio è il PlaytimeInfo.
Il PlaytimeInfo è numerato allo stesso modo dello Snapshot, in modo da rilevare eventuali pacchetti persi.
Poiché il client non può attendere che il server gli confermi la ricezione di un dato messaggio (si veda 3.1.2), ogni messaggio PlaytimeInfo contiene un riferimento al messaggio precedente; finché il client non riceverà conferma dell'avvenuta ricezione di un dato messaggio PlaytimeInfo, tutti i messaggi successivi all'ultimo della cui ricezione ha avuto conferma saranno inviati al server nello stesso pacchetto di rete.

Il PlaytimeInfo contiene, quindi, i seguenti campi:

a) Id: identifica il messaggio.

b) LastSynchedWorldImageId: è l'ultima WorldImage che il client ha ricevuto.

c) UserActorCommand: il comando che l'utente ha impartito allo UserActor che controlla.

d) PreviousPlaytimeInfo: il riferimento al messaggio precedente; può essere *null*.

ClientResearchInfo Per scopi di ricerca, si è ritenuto utile raccogliere alcuni dati sul funzionamento del client; il messaggio ClientResearchInfo è stato introdotto per soddisfare questa esigenza.

Le informazioni raccolte e inviate al server (che le esporterà in un file al termine della sessione) sono gli eventi di perdita di tracking[12], e la lista degli Snapshot ricevuti.

ServiceInfo Il designer dell'applicazione potrebbe avere l'esigenza di dare agli utenti indicazioni sui passi da seguire per raggiungere un determinato obiettivo, o informarlo circa eventi che si sono verificati nel corso della sessione. Tali informazioni sono comunicate all'utente nelle modalità illustrate in 3.4.7. Inoltre, lo sviluppatore potrebbe avere l'esigenza di apportare delle modifiche alla mappa, ad esempio attivare o disattivare un determinato GameObject.

Tipicamente, tali informazioni sono inviate di rado, quindi si è ritenuto accettabile, se non preferibile, definire un messaggio dedicato per soddisfare le sopraelencate esigenze; tale messaggio prende il nome di ServiceInfo.

Poiché la logica di trasmissione dei messaggi ServiceInfo fa parte dell'ApplicationMode, l'oggetto m_ApplicationMode in ServerApplicationController contiene un campo di tipo System.Action<ServiceInfo>, che rappresenta una funzione che l'ApplicationMode può invocare ogniqualvolta deve inviare un ServiceInfo:

```
public abstract class ApplicationMode : MonoBehaviour
{
    public event Action ScoreChanged;
    public Action<ServiceInfo> SendServiceInfoProc = null;
    public Action EndSessionProc = null;

    // ...
}
```

L'invocazione avviene nel modo seguente:

```
public override bool UpdateSession(long time)
{
    // ...

    string theMessageForTheUser = "The world is yours!"

    var serviceInfo = new ServiceInfo(theMessageForTheUser,
        mapMods)

    SendServiceInfoProc?.Invoke(serviceInfo);

    // ...
}
```

[12] Solo per HoloLens

SessionResults Al termine della sessione, il server invia ai client questo messaggio, contenente il punteggio raggiunto da ogni utente. Tale messaggio viene interpretato dai client come indicazione che la sessione è terminata, pertanto non è stato definito un messaggio appositamente per quello scopo.

ServerContentMessageHandler e ClientContentMessageHandler

Le classi ServerContentMessageHandler e ClientContentMessageHandler differiscono soltanto nel fatto che, lato server, è possibile specificare il destinatario quando si vuole inviare un messaggio e, quando si riceve un messaggio, accedere al mittente. Al contrario, per il client, che può inviare e ricevere messaggi solo dal server, tali informazioni non sono necessarie.

Poiché tutto il codice per la gestione dei pacchetti è contenuto in ContentMessageHandler, non c'è duplicazione di codice.

3.2.3 Ping Message Handler

La misura del tempo richiesto per portare un pacchetto di rete dal server al client, e di nuovo dal client al server, detto Round-trip delay time (o Round-trip time), è un dato preziosissimo nel contesto di un videogioco competitivo; infatti, più ampio è tale ritardo, maggiore sarà il ritardo percepito dagli utenti.

Per misurare il RTT, il linguaggio C# e .NET mettono a disposizione la classe System.Net.NetworkInformation.Ping, che implementa la funzione di Ping tramite messaggi ICMP *echo request* e *echo reply*. Purtroppo, tale classe non supporta Universal Windows Platform, che è l'unica opzione possibile per sviluppare un'applicazione per HoloLens, perciò la funzionalità di Ping è stata implementata tramite un apposito layer su netcode.io.

ServerPingMessageHandler

Lato server, è possibile chiamare la funzione ServerPingMessageHandler.SendPing(byte clientId), specificando l'Id del client relativamente al quale si vuole misurare il ritardo.

Misura del Round-Trip Time

Il server esegue periodicamente, con una frequenza di circa 1 volta al secondo, una misura del ritardo da ogni client. Dopo aver ottenuto la misura, la registra in una opportuna lista, quindi calcola la media tra tutti i ritardi ottenuti per un dato client; la media del RTT è quindi inviata al client.

3.2.4 Entity Manager

Con Entity si intende un elemento del mondo virtuale, che può interagire con ogni altro elemento del mondo. Dal punto di vista implementativo, esso consiste in una classe C#,

e in un Prefab (B.5) contenente tutti gli elementi costituenti l'entità.

Al fine di gestire in modo ordinato le entità di gioco, è stato introdotto il modulo Entity Manager, che in parte ha la funzione di contenitore di entità, e in parte svolge alcune importanti funzioni quali la creazione e distruzione di entità, l'aggiornamento delle stesse, e la sincronizzazione del loro stato tra server e client. Inoltre, tale classe contiene alcuni metodi di uso comune per l'interazione delle entità; ad esempio FindInteractable_Contact() e FindInteractable_Ray() permettono di trovare un oggetto che possa interagire con un'entità, rispettivamente tramite contatto diretto e a distanza.

Entity

La classe Entity è una delle più importanti del framework, in quanto la complessità del codice dell'applicazione risiede in gran parte in quelle classi che derivano da Entity e/o dipendono da essa, ad esempio l'ApplicationMode e le classi da esso derivate.

Classe Entity.Definition Poiché ogni tipologia di entità avrà le sue caratteristiche peculiari, si è reso necessario trovare una soluzione flessibile che permettesse, da una parte, di definire come l'entità è fatta, e, dall'altra, di trasferire tale definizione, o descrizione, dal server, in cui tale definizione è generata, al client, con un intervento minimo da parte dello sviluppatore.

La classe Entity.Definition, di seguito illustrata, è la soluzione proposta:

```
public class Definition
{
    public byte ResourceId { get; }
    public Vector3 StartPosition { get; }
    public Quaternion StartRotation { get; }
    public Dictionary<byte, object> Parameters { get; } = new
        Dictionary<byte, object>();

    // ...
}
```

Il campo ResourceId identifica il percorso del prefab dell'entità, che può essere caricato a *runtime* tramite la funzione UnityEngine.Resource.Load<T>(string path).

Dopo aver creato un'istanza del prefab, si ottiene il componente Entity tramite metodo UnityEngine.GameObject.GetComponent<T>(), eseguendo, quindi, il metodo virtuale Entity.Init(byte id, Definition definition, EntityManager entityManager), che permette di configurare l'entità con i parametri forniti dalla definizione data.

All'occorrenza, lo sviluppatore potrà ridefinire, tramite override, il metodo Init() di una classe che deriva da Entity, in modo da configurarla in modo opportuno.

Il campo Parameters di Entity.Definition permette allo sviluppatore di passare al metodo Init() dei parametri di configurazione dell'Entity (ad esempio l'altezza, piuttosto che la velocità).

L'uso incoraggiato è definire un enum per ogni classe che deriva da Entity, per indicare il tipo di parametro. Di seguito è mostrato un esempio di questo approccio:

```
public class RoboEnemy : Enemy // Nota: Enemy deriva da Entity.
{
    public enum DefinitionParameter : byte
    {
        AverageSpeed = 0,
        MaxMovementRadius,
    };

    public override void Init(byte id, Definition definition,
        EntityManager entityManager)
    {
        base.Init(id, definition, entityManager);

        m_StartPosition = definition.StartPosition;
        m_Right = this.transform.right;
        m_Up = this.transform.up;

        var avgSpeedIndex =
            (byte)DefinitionParameter.AverageSpeed;
        m_AverageSpeed =
            (float)definition.Parameters[avgSpeedIndex];
        var maxMovRadiusIndex =
            (byte)DefinitionParameter.MaxMovementRadius;
        m_MaxMovementRadius =
            (float)definition.Parameters[maxMovRadiusIndex];
    }

    // ...
}
```

Sincronizzazione dello stato Al fine di minimizzare lo sforzo richiesto allo sviluppatore per sincronizzare lo stato delle entità tra il server e i client, è stata introdotta la classe Entity.Image, che, traendo vantaggio dallo strumento della *reflection* del linguaggio C#, ha permesso di semplificare il codice per la sincronizzazione.

L'idea alla base del funzionamento della classe Entity.Image è l'utilizzo di attributi per qualificare in modo opportuno quelle variabili, campi e proprietà, che devono essere sincronizzati tra le varie parti in gioco. A tal fine, sono stati definiti degli attributi per specificare le variabili da sincronizzare, e/o le funzioni da eseguire sia nel server che nei client.

TransformInterpolator Il TransformInterpolator è una classe derivata da MonoBehavior [B.1], che può essere assegnata a un qualunque GameObject del prefab di un Entity per sincronizzare la posizione e/o la rotazione del GameObject stesso tra server e client.

Tale classe è stata pensata per supportare il meccanismo dell'interpolazione delle entità descritto in [3.1.4], da cui il nome.

Dopo aver assegnato il componente in esame ad un GameObject, nell'Inspector di Unity è possibile indicare se si vuole sincronizzare la sola posizione, la sola rotazione, o entrambe (come illustrato in figura 3.6).

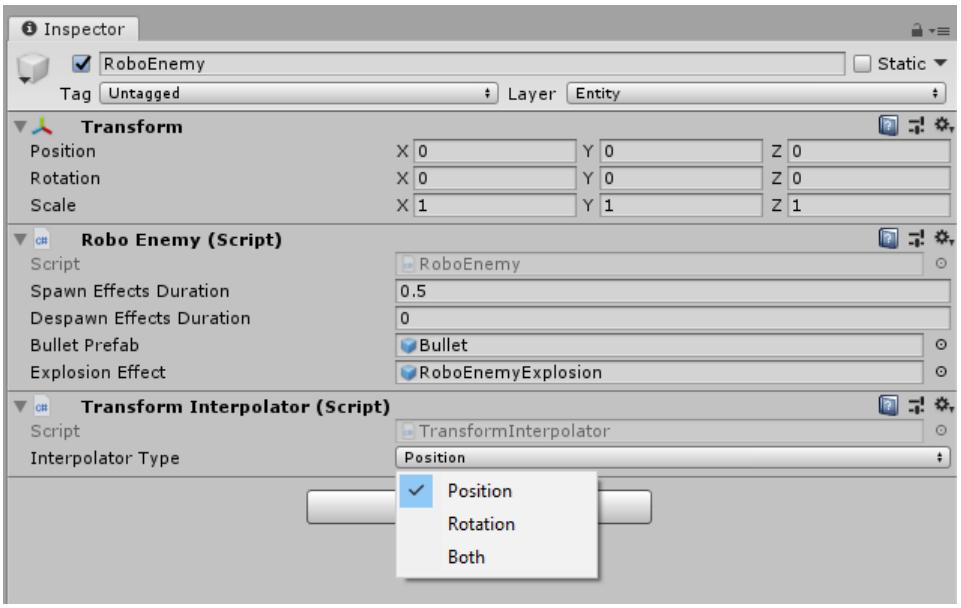

Figura 3.6. Nell'inspector di Unity è possibile selezionare il tipo di TransformInterpolator, e, quindi, quali informazioni sincronizzare tra server e client.

Message System

Il modulo Entity Manager contiene un campo di tipo event denominato OnEntityMessageEvent, che viene lanciato tutte le volte in cui un'entità invia un nuovo messaggio; se uno degli altri moduli o l'ApplicationController è interessato ad ascoltare i messaggi delle entità, può iscriversi a tale evento.

Un esempio di questo tipo è l'ApplicationMode, che in funzione dei messaggi che gli giungono dalle entità, stabilisce come far avanzare lo stato della sessione.

La scelta di adottare questo pattern di programmazione ha permesso di separare la logica delle entità dalla logica dell'ApplicationMode, cioè della modalità di applicazione

o di gioco, rendendo possibile l'aggiunta di nuove modalità con un intervento minimo o nullo sulla code-base esistente.

ServerEntityManager

Ciò che distingue maggiormente la variante dell'EntityManager presente sul server, dalla classe base, è la presenza dei seguenti metodi:

a) GenerateWorldImage: genera lo stato del mondo, che contiene lo stato delle entità.

b) GenerateCumulativeWorldImage: combina più stati del mondo per ridurre la dimensione complessiva, senza perdere informazione.

c) ExecuteUserActorRemoteCommand: esegue il comando remoto proveniente da un client.

ClientEntityManager

Il ClientEntityManager mette a disposizione due metodi aggiuntivi per la sincronizzazione dello stato delle entità (SyncEntities()) e l'esecuzione, in locale, del comando che l'utente ha impartito all'entità che egli controlla (ExecuteUserActorActions()).

3.2.5 Map Manager

Il Map Manager è l'unico modulo, presente sia nel server che nel client, che non presenta variazioni tra server e client.

La funzione del MapManager è di semplice contenitore della mappa e di gestore dell'origine del mondo.

Origine del mondo

L'Origine del Mondo, o WorldOrigin, non è altro che un GameObject di cui tutti gli elementi del mondo virtuale, entità e mappa, sono figli.

3.2.6 Menu Manager

Il Menu Manager si occupa dell'inizializzazione degli elementi dell'interfaccia utente, e costituisce una collezione degli elementi stessi e dei diversi menù che li contengono. Infatti, quasi a ogni stato dell'ApplicationController corrisponde un menù dedicato.

3.2.7 World Anchor Manager

Il World Anchor Manager è un modulo del solo client, che viene attivato quando il client è lanciato sul dispositivo HoloLens, in quanto utilizza funzioni necessarie e disponibili solo per questo dispositivo.

La funzione di tale modulo è permettere la creazione e la distruzione di ancore spaziali, o World Anchor.

World Anchor

Un'ancora spaziale è un punto nello spazio che l'HoloLens è in grado di riconoscere dall'ambiente circostante, e che può essere memorizzato sul dispositivo e "ricordato" tra diverse sessioni.

A tale punto si può associare un sistema di riferimento, nel quale può essere definita la posizione di un ologramma, che quindi sarà "ancorato" al mondo.

3.2.8 Virtual World Aligner

Analogamente al World Anchor Manager, anche questo modulo entra in gioco solo se il dispositivo utilizzato è un HoloLens, in quanto la sua funzione è di allineare il mondo virtuale a quello reale. Infatti, all'accensione l'HoloLens non sa dove si trova in relazione alla scena virtuale, perciò fissa automaticamente la posizione al momento dell'accensione come origine del mondo virtuale; è, quindi, necessario applicare una trasformazione al sistema di riferimento dell'HoloLens per fare corrispondere il mondo virtuale al mondo reale.

Algoritmo di allineamento

Per effettuare l'allineamento, il Virtual World Aligner fa uso delle ancore che sono state disposte nell'area di azione, ma, in base alla mappa in cui si svolge l'azione, può non essere sufficiente una sola ancora, per due ragioni:

1. Il dispositivo è in grado di riconoscere un'ancora spaziale con una precisione di alcuni centimetri.

2. Più un oggetto è distante dall'ancora, maggiore sarà l'errore sulla sua posizione, in quanto l'errore nella determinazione della rotazione dell'ancora risulta sempre più evidente man mano che ci si allontana dall'origine.

Poiché la mappa di gioco può coprire anche più di due stanze e il corridoio che le connette, si è reso necessario trovare un sistema per allineare gli ologrammi su una scala che è circa un ordine di grandezza la distanza massima permessa da una sola ancora, che è circa 3 metri.

La soluzione adottata consiste nel disporre, in tutta l'area di interesse, un certo numero di ancore, poste a una distanza variabile dai 5 ai 10 metri le une dalle altre.

In ogni momento, l'applicazione determina quali ancore, tra quelle localizzate dall'HoloLens, sono più vicine all'utente; tre casi si possono verificare:

a) Non è stata localizzata nessuna ancora: nessuna correzione è effettuata sugli ologrammi.

b) Almeno un'ancora è stata localizzata, ma meno di tre: l'ancora più vicina è utilizzata per l'allineamento.

c) Almeno tre ancore sono state localizzate: tre ancore sono utilizzate per l'allineamento.

Allineamento con una sola ancora L'origine del mondo è, dapprima, traslata in modo da sovrapporre l'ancora al punto di riferimento corrispondente, cioè il punto in cui l'ancora dovrebbe trovarsi se ci fosse allineamento perfetto tra il mondo reale e quello virtuale.

Dopo aver effettuato la traslazione, l'origine del mondo è fatta ruotare attorno all'ancora per orientarla in modo concorde con il punto di riferimento.

Allineamento con tre ancore Se tre ancore sono state localizzate, le due ancore più vicine, su cui tipicamente l'errore di misura è minore, sono utilizzate per traslare e ruotare il WorldOrigin.

Indicando con anchor_0, anchor_1 e anchor_2 le tre ancore in ordine di distanza crescente dall'utente, l'algoritmo di allineamento lavora in tre fasi:

1. Determina la roto-traslazione da applicare al WorldOrigin considerando le ancore nell'ordine 0, 1, 2.

2. Determina la roto-traslazione da applicare al WorldOrigin considerando le ancore nell'ordine 1, 0, 2.

3. Calcola il valore finale di roto-traslazione pesando i valori ottenuti sopra in funzione della distanza da anchor_0 e anchor_1.

La trasformazione non è, in realtà, applicata immediatamente al WorldOrigin, ma questo è fatto muovere verso la posizione e rotazione finali in modo progressivo, così da non far notare, o almeno cercare di nascondere, il processo di riallineamento.

3.2.9 Arm Swinger

Arm Swinger può essere visto come un componente del client ed è attivato solo quando si fa uso di un dispositivo VR accoppiato a motion controller. Si tratta di un metodo di locomozione intuitivo che permette di spostarsi nello spazio muovendo le braccia come se si stesse camminando o correndo.

Per attivare la camminata tramite Arm Swinger, l'utente preme un tasto dei motion controller (il *gripper* del controller Oculus Touch mostrato in figura 3.7), quindi oscilla le braccia avanti e indietro; per muoversi in diagonale, l'utente deve inclinare i controller.

3.2.10 Actor Controller

L'Actor Controller è il componente del client che si occupa di raccogliere i comandi che l'utente impartisce all'entità controllata. Ogni input è convertito in un'azione che l'entità può compiere, e questa è salvata in un buffer. Alla frequenza di aggiornamento del client, il buffer è svuotato ed è generato un nuovo comando per l'entità, eseguendo le azioni nell'ordine in cui sono stati premuti i tasti.

Figura 3.7. Motion controller Oculus Touch utilizzato come dispositivo di input nella modalità VR.

3.3 Server

Il cuore del server è il *Server Application Controller*, che ha il ruolo di coordinatore dei moduli (NetworkManager, MenuManager, ContentMessageHandler, PingMessageHandler, EntityManager e MapManager) descritti in 3.2; inoltre, esso gestisce l'applicazione durante tutta la sua vita, dal momento in cui l'utente lancia l'eseguibile, fino al momento il cui l'applicazione viene chiusa.

L'applicazione, in ogni istante, si trova in uno stato ben definito di una macchina a stati costruita tenendo bene a mente la funzione del server all'interno di un framework per applicazioni multiutente. Una caratteristica che, in particolare, si è voluto dare al server è la capacità di autogestirsi, intendendo con ciò che il sistema, dal momento in cui è stato lanciato e configurato, non richiede più l'intervento umano per funzionare correttamente tra sessioni successive.

La macchina a stati nel suo complesso è mostrata nella figura 3.8.

3.3.1 Avvio dell'applicazione

Quando l'eseguibile è lanciato, il server si trova nello stato **ConfiguringServer**, mostrando a schermo un insieme di parametri di configurazione relativi alla modalità di gioco, alla mappa e al networking; in questo stato è ancora necessario l'intervento dell'utente.

Figura 3.8. Macchina a stati del server.

Configurazione del framework

Al fine di permettere la costruzione di un'applicazione sul framework tramite la sola aggiunta di nuovi elementi[13] e senza dover modificare il framework stesso, si è ritenuto conveniente introdurre un file di configurazione (mostrato in figura 3.9), che fornisca al framework sottostante l'applicazione le informazioni per trovare i nuovi elementi introdotti dall'applicazione stessa.

Tale file di configurazione ha estensione *.asset* ed è un'istanza della classe **Configuration**, derivante da ScriptableObject [B.2].

La classe Configuration contiene un campo di tipo List<string> denominato ApplicationModes, in cui è possibile inserire il nome dei prefab aventi come componente una classe derivante da ApplicationMode; tali prefab devono essere posti all'interno di una cartella denominata ApplicationModes, che a sua volta deve essere posta nella cartella speciale Resources[14].

[13] Classi, prefab, etc.

[14] Tale cartella è speciale perché Unity permette il caricamento a runtime degli asset posti dentro le cartelle denominate Resources

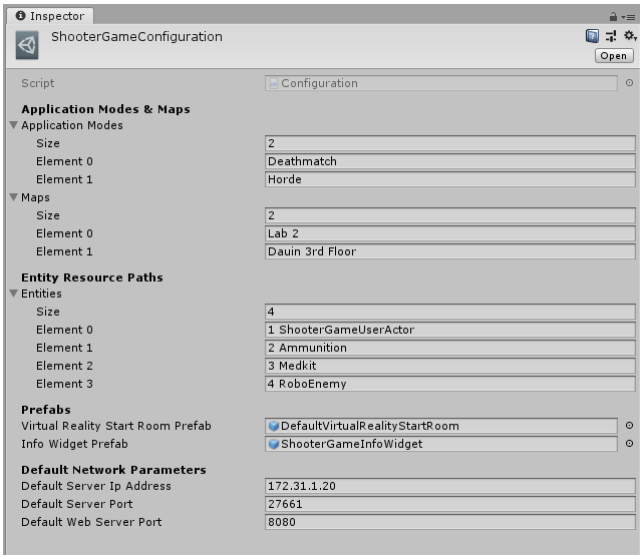

Figura 3.9. File di configurazione del framework.

Il campo successivo è Maps, anch'esso di tipo List<string>. Come suggerisce il nome, tale campo contiene la lista delle mappe che saranno selezionabili dal menù di configurazione del server. Poiché il metodo di caricamento delle mappe è lo stesso impiegato per le modalità di gioco, anche in questo caso le mappe, sotto forma di Prefab, vanno poste in una cartella denominata Maps, posta all'interno della cartella speciale Resources.

Il terzo campo, Entities, è ancora una lista (List<string>), in cui lo sviluppatore dovrebbe inserire i prefab delle Entity nel seguente formato:

n NomePrefabEntità

dove n, che ha la funzione di chiave o id, è un numero che va da 1 a 255, e NomePrefabEntity è il nome del prefab dell'Entity. Il valore n=0 ha significato speciale ed è impiegato per rilevare eventuali errori nel Framework e/o nell'applicazione, così come il valore n=1 deve sempre essere assegnato all'Entity che deriva da UserActor.

Un dizionario del tipo System.Collection.Generic.Dictionary<byte, string> sarebbe stato più adatto alla scopo, ma non è direttamente serializzabile, perciò è stata impiegata la lista. Per separare la chiave (n) e il valore (il nome del prefab), è sufficiente utilizzare il metodo string.Split(char delimiter) con delimiter=' ', in quanto l'unico spazio presente nella stringa è quello che separa la chiave dal valore. NomePrefabEntity identifica, ancora una volta, un prefab contenuto in una cartella speciale che prende il nome di Entities e posta nella cartella Resources.

I due campi successivi, raggruppati sotto il titolo di Prefabs, sono VirtualRealityStartRoomPrefab e InfoWidgetPrefab. Tali campi contengono, rispettivamente, il riferimento alla stanza virtuale in cui si troverà l'utente che lanci il client in modalità VR, prima di

unirsi alla sessione di gioco, e il riferimento al widget che presenta le informazioni di gioco all'utente durante la sessione di gioco. Tali elementi non sono istanziati né tantomeno impiegati nel server, ma sono presenti nel file di configurazione in quanto quest'ultimo è lo stesso sia per il client che per il server.

Il file di configurazione si chiude con tre campi relativi al networking:

a) Default Server IP Address: indirizzo IP al quale il client potrà trovare il server.

b) Default Server Port: porta utilizzata dal protocollo di rete netcode.io

c) Default Web Server Port: porta utilizzata dal WebServer

È possibile indicare all'avvio, tramite parametri passati all'eseguibile, un diverso valore per ognuno dei parametri sopraindicati, funzione utile per eseguire dei test senza dover ricompilare il codice.

Configurazione della sessione

Il menù presentato all'utente all'avvio del server permette di selezionare una delle modalità inserite nel file di configurazione, la durata della sessione di gioco in minuti, la mappa di gioco, il numero massimo di utenti, il periodo (in millisecondi) di trasmissione, da parte del server, dei pacchetti Snapshot, e il periodo di trasmissione, da parte del client, dei pacchetti di tipo PlaytimeInfo.

La modalità e la mappa sono scelte tramite dropdown list, invece gli altri parametri possono essere inseriti manualmente. Per i parametri numerici, il sistema esegue un controllo per accertarsi che il valore inserito sia numerico e appartenga al dominio dei valori accettati.

3.3.2 Server in attesa dei client

Dopo aver confermato la configurazione premendo il pulsante Start Server, l'applicazione passa dallo stato ConfiguringServer allo stato **WaitingForClientConnections**, in cui è pronto ad accettare eventuali richieste di connessione da parte dei client.

Nel passaggio di stato il server esegue un insieme di operazioni, le più importanti delle quali sono il caricamento, tramite metodo UnityEngine.Resources.Load<T>(string path), dei prefab dell'ApplicationMode e della mappa selezionate in fase di configurazione. L'ApplicationMode diventa figlio del ServerApplicationController, volendo indicare, in tal modo, che da un punto di vista gerarchico tale oggetto è sottoposto al controllo dell'ApplicationController.

L'istanza della mappa, al contrario, è assegnata come figlia del GameObject WorldOrigin, per due ragioni principali:

1. Poter accedere più velocemente, nell'Editor di Unity, alla collezione di Entity (anch'esse figlie di WorldOrigin) e agli elementi della mappa, senza dover espandere gerarchie di oggetti inutilmente profonde. L'accesso a tali elementi nell'Editor è fatto tipicamente per ragioni di debugging.

2. Poiché avere tutti gli elementi del mondo virtuale all'interno di un unico oggetto semplifica l'operazione di trasformazione del sistema di riferimento, operazione necessaria, come spiegato in 3.2.8.

Il caricamento della mappa avviene nel Map Manager, che ha anche il compito di creare il GameObject WorldOrigin nel metodo Awake() [B.1].

Dopo aver caricato la mappa, il server disabilita la camera di default e attiva quella definita per la mappa di gioco; ciò permette allo sviluppatore di posizionare nell'Editor una camera, in modo ottimale per la particolare mappa che sta costruendo. Tale camera può essere utile, in fase di test, per valutare visivamente il corretto funzionamento del sistema, ad esempio comparando la posizione di un client e quella supposta dal server.

Il passo successivo consiste nell'attivare il (network) server tramite il ServerNetworkManager, specificando il massimo numero di client che possono connettersi, e la porta. Contemporaneamente è attivato anche il WebServer, che permette ai client sia di accedere alle informazioni sul server, sia a richiedere un identificativo per potersi connettere al server e unirsi alla sessione di gioco.

Infine, l'applicazione attiva due coroutine [B.3], che eseguono le seguenti funzioni:

1. Per ogni client connesso, inviano periodicamente informazioni circa tutti i client connessi, ad esempio il nome e se sono già pronti (si veda 3.2.2) per iniziare la sessione.

2. Per ogni client connesso, il server esegue periodicamente un ping (si veda 3.2.3 per maggiori dettagli) al client; non appena riceve il messaggio di risposta, aggiorna la stima del round-trip time nel modo seguente:

$$RTT_i = (0.9 \times RTT_{i-1}) + (0.1 \times RTT)$$

dove RTT_i è il nuovo valore, RTT_{i-1} è il valore precedente, e RTT è il valore calcolato sull'ultimo messaggio di ping inviato.

In questa fase il server può ricevere e accettare richieste di connessione da parte dei client. Per ogni richiesta di connessione, il server genera un id unico tramite il ClientIdProvider; dal momento della generazione dell'id, il client ha un tempo prefissato per instaurare la connessione con il server, pena il rifiuto della richiesta di connessione e il rilascio (cioè l'id è reso disponibile per altri client) dell'id generato. Se la connessione ha successo, il server registra l'avvenuta connessione del client.

Dopo aver instaurato la connessione a livello di rete, il client invia, in modo affidabile, il pacchetto ClientPresentation, contenente le informazioni sul client, alla ricezione del quale il server invia, in modo affidabile, il pacchetto SessionDescription, che contiene la descrizione della sessione.

Per una ragione qualsiasi, un client potrebbe decidere o essere forzato da cause esterne (ad esempio un guasto) a interrompere la connessione con il server, in tal caso quest'ultimo provvederà a rilasciare le risorse allocate per il client e ad aggiornare gli altri client.

3.3.3 Avvio della sessione

Dopoché tutti i client connessi hanno inviato il pacchetto ClientReady, il server entra nello stato **SendingStartSessionInfo**, inizializzando la sessione di gioco. Le variabili utilizzate in questo stato e nel successivo UpdatingSession, tra cui il tempo di gioco (NetworkTime), sono azzerate, e il pacchetto StartSessionInfo è inviato in modo affidabile a tutti i client.

3.3.4 Aggiornamento della sessione

In seguito alla ricezione da parte di tutti i client del pacchetto StartSessionInfo, l'applicazione avvia finalmente la sessione entrando nello stato **UpdatingSession**. In questa fase le azioni di gioco sono mostrate a schermo, assieme al tempo rimanente per la sessione in corso.

Le operazioni principali, eseguite nella funzione Update(), vedono l'aggiornamento dello stato dell'ApplicationMode (funzione ApplicationMode.UpdateSession(long time)), l'aggiornamento dello stato delle entità (funzione EntityManager.UpdateEntities(long time)), e infine la trasmissione del pacchetto Snapshot.

Il pacchetto Snapshot è inviato, alla frequenza impostata in fase di configurazione del server, in modo non affidabile, ciò comporta che si possano verificare uno o più dei seguenti casi:

a) Il messaggio arriva a destinazione in ordine: questo è il caso migliore.

b) Il messaggio arriva a destinazione fuori ordine: ad esempio il client riceve prima lo Snapshot con Id 32, poi quello con Id 31[15].

c) Il messaggio non arriva a destinazione: questo è il caso peggiore.

I casi *b* e *c* sono trattati allo stesso modo, in quanto, come sarà ora spiegato, ogni Snapshot contiene tutte le informazioni degli Snapshot precedenti la cui ricezione non sia stata segnalata al server da parte dei client.

Sincronizzazione server-client

La sincronizzazione tra server e client avviene principalmente tramite la trasmissione di pacchetti Snapshot da parte del server e PlaytimeInfo da parte del client.

Poiché la trasmissione avviene in modo non affidabile, non può essere garantita la ricezione dei pacchetti.

Ack implicito In un contesto generale, una connessione di rete affidabile richiede che la ricezione di un dato pacchetto sia confermata dal destinatario con un opportuno pacchetto di ack a livello di protocollo di rete. Con ack implicito si intende che la conferma della ricezione di un dato Snapshot (PlaytimeInfo), è contenuta nel messaggio PlaytimeInfo

[15] Snapshot successivi hanno id crescenti

(Snapshot); in particolare il Framework ha un'implementazione di questo meccanismo che vede il destinatario indicare sempre l'ultimo messaggio ricevuto dal mittente:

a) Snapshot: LastReceivedPlaytimeInfoId

b) PlaytimeInfo: LastSynchedWorldImageId[16]

Come è stato detto, sia Snapshot che PlaytimeInfo sono numerati in modo sequenziale, ed è possibile rilevare i pacchetti persi o arrivati fuori ordine, ma poiché non c'è ritrasmissione e/o riordinamento dei pacchetti persi, sono stati implementati due meccanismi al fine di garantire che non siano perse informazioni durante la comunicazione tra server e client.

Annidamendo in PlaytimeInfo Tale messaggio contiene, tra le altre cose, i comandi che l'utente impartisce all'entità da lui controllata. La perdita di un comando può essere più o meno critica, in base al contesto: si pensi, ad esempio, a un videogioco di genere sparatutto. Se il giocatore spara e il messaggio che contiene il comando non arriva al server, il giocatore penserà di aver sparato, e magari anche di aver colpito il bersaglio, ma poiché è il server che svolge il controllo sull'esito dello sparo, nessuno sparo verrà eseguito e, quindi, il bersaglio non subirà alcun danno.

Per garantire che tutti i comandi arrivino a destinazione, e siano eseguiti nell'ordine in cui sono stati impartiti dall'utente (o, più in generale, che tutti i messaggi PlaytimeInfo inviati arrivino al server e siano processati in ordine), il meccanismo adottato consiste nel mettere all'interno di un messaggio PlaytimeInfo il messaggio PlaytimeInfo precedente, cioè utilizzare una forma di nesting.

Di seguito è mostrata la classe PlaytimeInfo, in cui è possibile notare l'annidamento:

```
public class PlaytimeInfo : ApplicationMessage
{
    public int Id { get; }
    public int LastSynchedWorldImageId { get; }
    public UserActor.Command UserActorCommand { get; }
    public PlaytimeInfo PreviousPlaytimeInfo { get; set; }  //
        Nested PlaytimeInfo

    // ...
}
```

Per ogni nuovo (cioè successivo all'ultimo pacchetto ricevuto) PlaytimeInfo ricevuto, il server salva in memoria l'Id del pacchetto, che invierà al client tramite messaggio Snapshot. Conoscendo l'ultimo PlaytimeInfo ricevuto, il server, alla ricezione di un nuovo PlaytimeInfo, che conterrà tutti i PlaytimeInfo più vecchi (fino al messaggio subito successivo all'ultimo segnalato dal server) in forma annidata, processa tutti i messaggi in ordine.

[16] LastSynchedWorldImageId corrisponde esattamente all'Id dell'ultimo Snapshot ricevuto

A titolo esemplificativo, si supponga che l'ultimo PlaytimeInfo ricevuto dal server sia il 157, e che il client debba inviare un nuovo PlaytimeInfo con Id 161, avendo già inviato quelli con Id tra 157 e 161, che non sono arrivati al server.

Inoltre, si supponga che il server abbia segnalato con successo al client di aver ricevuto il messaggio PlaytimeInfo con Id 155, e che abbia anche inviato il messaggio Snapshot contenente l'informazione che il server ha già ricevuto il PlaytimeInfo 157, ma che tale pacchetto non arrivi a destinazione (perché è stato perso o anche perché non è ancora arrivato a destinazione).

Il client invierà al server tutti i messaggi PlaytimeInfo da 156 a 161, in modo annidato in un unico pacchetto di rete. Se tale pacchetto arriva a destinazione, il server vedrà che esso contiene tutti i PlaytimeInfo da 155 a 161, ma poiché ha già ricevuto tutti i PlaytimeInfo fino a 157, scarterà le informazioni contenute nei pacchetti con Id inferiore e uguale a 157 e processerà le informazioni da 158 in poi.

Codifica differenziale Al fine di minimizzare la dimensione dei pacchetti di rete contenenti messaggio PlaytimeInfo, una forma di codifica differenziale è stata impiegata, facendo le seguenti considerazioni:

a) PreviousPlaytimeInfo deve avere Id pari all'Id del messaggio che lo contiene ridotto di uno, perciò non è necessario salvare tale informazione nel pacchetto di rete, in quanto ricavabile dalle altre informazioni salvate.

b) Il valore LastSynchedWorldImageId di un dato messaggio potrebbe essere diverso nei messaggi annidati, ma poiché il server è interessato solo al valore più recente[17], non è necessario salvarlo nel pacchetto.

c) La classe UserActor.Command contiene tre campi: posizione e rotazione di testa e mani/controller, la lista di azioni (di tipo *Actor.Action*), e il tempo di esecuzione del comando.

   Poiché i valori di posizione (3 *float* per 3 parti tracciate) e rotazione (4 *float*[18] per 3 parti tracciate) richiedono molti byte:

$$n_{tot}(byte) = ((3 \times 4) + (4 \times 4)) \times 3 = 84$$

   e i messaggi annidati sono consecutivi e quindi vicini nel tempo, come per LastSynchedWorldImageId, tali valori sono salvati una sola volta per il PlaytimeInfo più esterno (cioè il più recente) e assunti uguali per tutti i messaggi annidati.

   Inoltre, per la stessa ragione, il tempo di esecuzione del comando, che è di tipo *long*, è sottratto dal tempo del comando più recente e la differenza così ottenuta è convertita in *int* e salvata come tale, occupando 4 *byte* invece che 8.

[17] Il server ha bisogno di sapere il valore più recente per fornire al client solo le informazioni mancanti

[18] La rotazione è rappresentata tramite quaternione

Infine, per indicare la presenza di un PreviousPlaytimeInfo annidato, si salva un bool. La funzione di conversione in flusso di byte e quella inversa che dal flusso ricostruisce il messaggio sono ricorsive e sono implementate nel file PlaytimeInfo.cs.

Merging di WorldImage

Sebbene lo stesso meccanismo di annidamento adottato per il PlaytimeInfo fosse utilizzabile per lo Snapshot, si è preferito adottare un metodo più efficiente dal punto di vista dell'utilizzo della memoria; infatti, supponiamo di avere 10 entità e che per ognuna di esse lo stato richieda 100 *byte* (si pensi, ad esempio, allo UserActor[19] che richiede 84 *byte* per la sola sincronizzazione delle *Transform*[20] di testa e mani). Una singola WorldImage peserà 1000 *byte*, quindi anche annidando una sola WorldImage, il pacchetto supererà la soglia dei 1200 *byte*, portando alla frammentazione del messaggio.

Per evitare la frammentazione, è stato introdotto il concetto di Merging (fusione):

a) Date due WorldImage, worldImageA e worldImageB, dove la prima precede la seconda in termini temporali, worldImageAB = worldImageA.Merge(worldImageB) è una nuova WorldImage contenente tutte le informazioni di stato ed gli eventi più recenti.

b) Date due Entity.Image, entityImageA e entityImageB, dove la prima precede la seconda in termini temporali, entityImageAB = Entity.Image.Merge(entityImageA, entityImageB) è una nuova Entity.Image contenente le informazioni di stato più recenti e l'elenco di tutti i metodi in entityImageA e entityImageB.

3.3.5 Disconnessione di un client

Nel corso della sessione può accadere che un client si disconnetta, intenzionalmente o anche perché la connessione si è interrotta indipendentemente dalla volontà dell'utente.

Se, in seguito alla disconnessione del client, non ci sono più client connessi, la sessione termina immediatamente e il server si riporta nello stato WaitingForClientConnections, dopo aver rilasciato le risorse allocate per la sessione appena interrotta. Nel caso in cui, invece, ci sono ancora utenti attivi nella sessione, lo UserActor controllato dal client disconnesso viene distrutto e tutte le risorse di gestione del client vengono rilasciate; i client ancora connessi riceveranno, attraverso il meccanismo descritto in 3.3.4, l'informazione che lo UserActor del client disconnesso è stato distrutto e ne distruggeranno la copia locale.

[19] Classe base rappresentante l'avatar dell'utente

[20] Classe contenente la posizione e rotazione di un GameObject

3.3.6 Terminazione della sessione

Disconnessione dei client La sessione può terminare prima che il tempo prefissato sia scaduto o gli obiettivi di gioco siano stati completati, perché i client si sono disconnessi volontariamente o involontariamente dal server.

In tale evenienza, il server rilascia tutte le risorse allocate e si riporta nello stato WaitingForClientConnections, rendendosi disponibile per una nuova sessione di gioco.

Le operazioni svolte includono la distruzione di tutte le Entity tramite l'EntityManager, l'azzeramento del dizionario m_ClientManagementInfos e il reset dei diversi moduli.

Obiettivi raggiunti o tempo terminato Se gli utenti hanno completato tutti gli obiettivi o è scaduta la sessione, l'applicazione entra nello stato **SendingSessionResults** e genera il messaggio SessionResults, contenente un punteggio per ogni utente che ha partecipato alla sessione fino alla fine.

Poiché tale messaggio è inviato una sola volta per sessione e deve arrivare a destinazione, prima che il server possa passare allo stato successivo, esso viene inviato in modo affidabile. Una volta ottenuta conferma dell'avvenuta ricezione da parte di tutti i client, il server esce dallo stato SendingSessionResults ed entra nello stato WaitingForClientConnections, mettendosi a disposizione per una nuova sessione.

3.4 Client

Il framework è stato progettato per supportare due diversi sistemi, un dispositivo di realtà virtuale, l'Oculus Rift, e un dispositivo di realtà aumentata, l'HoloLens prodotto da Microsoft, comportando una maggiore complessità del client rispetto al server. Infatti, le due tipologie di dispositivi presentano caratteristiche radicalmente differenti per quanto riguarda la presentazione degli oggetti virtuali, i sistemi di locomozione e quelli di interazione, e diverse specifiche per quegli aspetti tecnologici che pure li accomunano, come la capacità di tracciare il dispositivo nello spazio.

Per quanto detto sopra, la macchina a stati del client (si veda la figura 3.10) presenta più stati di quella del server; inoltre, alcuni stati sono propri della sola modalità AR. Come per il server, si parte dalla configurazione dell'applicazione, quindi si procede con il join della sessione, partecipando al gioco; infine, al termine della sessione, si può ripartire con una nuova sessione, o, opzionalmente, si può riconfigurare il client.

3.4.1 Avvio del client e configurazione

All'avvio, l'applicazione presenta un menù in cui è possibile configurare alcuni parametri del client che riguardano in particolare le caratteristiche dell'utente; questo è lo stato **ConfiguringClient**.

Nel caso si usi un dispositivo VR, all'utente è chiesto di indicare la mano dominante, che determinerà con quale controller potrà compiere l'azione principale; inoltre, egli può scegliere il sistema di controllo tra una modalità che permette di svolgere tutte le azioni

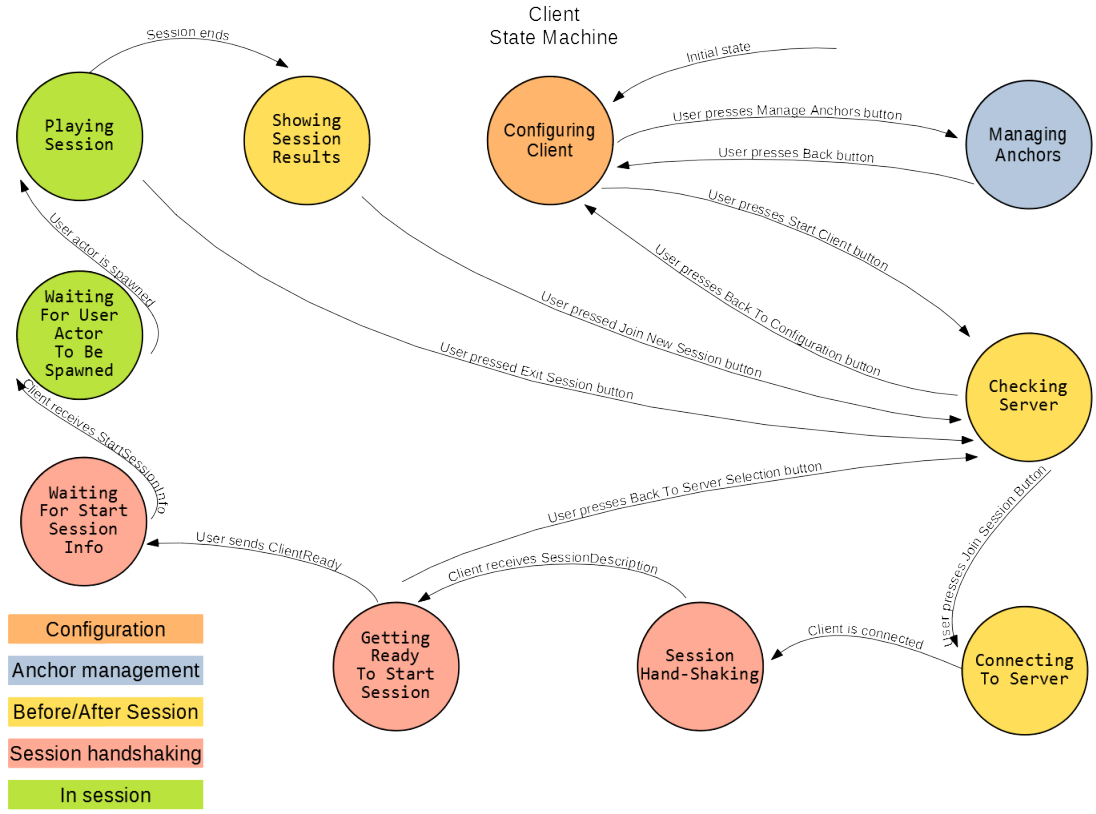

Figura 3.10. Macchina a stati del client.

con un solo pulsante, imitando il metodo di interazione di HoloLens, o la modalità più comunemente utilizzata per applicazioni VR, in cui si trae vantaggio dall'avere due motion controller, ciascuno con un insieme di tasti, a cui può essere associata una diversa funzione.

Sia all'utilizzatore di dispositivi VR, sia all'utilizzatore di dispositivi AR, è richiesto di inserire la propria altezza, utilizzando un opportuno slider. Il valore inserito sarà utilizzato per scalare gli avatar che rappresenteranno gli utenti nel mondo virtuale, in tal modo la loro rappresentazione virtuale sarà più fedele.

Stanza virtuale per dispositivi VR

All'avvio dell'applicazione, contrariamente all'HoloLens, in cui è mostrato solo un menù, il client VR si troverà in una stanza virtuale, che può essere modificata tramite file di configurazione [3.3.1].

La stanza virtuale ha due funzioni:

a) Permettere all'utente di prendere familiarità con il sistema di controllo, con particolare riferimento ai metodi di locomozione.

b) Fornire un riferimento visivo sia per questioni di immersività, che, soprattutto, per evitare disorientamento.

3.4.2 Gestione delle ancore spaziali

Solo nel caso si faccia uso di HoloLens, l'utente può accedere, facoltativamente, alle funzionalità per gestire le ancore spaziali. Selezionando Manage Anchors, viene mostrato un menù in cui deve essere indicata la mappa della quale modificare le ancore.

Una volta confermata la scelta della mappa, il sistema carica i punti di riferimento delle ancore, cioè quei punti in cui le ancore devono essere disposte. Tali punti sono rappresentati da un modello 3D raffigurante un sistema di assi cartesiani codificati utilizzando colori: il rosso indica l'asse X, il verde l'asse Y (verticale, seguendo la convenzione di Unity), e il blu per l'asse Z. Le ancore devono avere la stessa posizione e rotazione dei punti di riferimento.

L'utente ha a disposizione tre opzioni:

a) Rimuovere tutte le ancore di tutte le mappe.

b) Rimuovere tutte le ancore della mappa caricata.

c) Aggiungere una nuova ancora o modificare un'ancora esistente

Aggiunta di una nuova ancora L'aggiunta di una nuova ancora è possibile solo se non è stato raggiunto il numero di ancore prefissato per la particolare mappa caricata, che è stabilito dal designer tenendo conto della grandezza e complessità della mappa stessa.

La procedura di aggiunta di una nuova ancora consiste nei seguenti passi:

1. Indicare con lo sguardo il punto iniziale in cui posizionare l'ancora.

2. Tramite Air-Tap, o clicker, confermare la posizione iniziale: un oggetto, detto Anchor Handle, sarà posizionato nel punto indicato.

3. Effettuare eventuali aggiustamenti alla posizione e rotazione agendo sull'Anchor Handle.

4. Confermare la posizione e rotazione dell'ancora.

Modificare un'ancora esistente Al fine di semplificare il processo di disposizione delle ancore, è possibile rimuovere una singola ancora, o anche modificarne la posizione e/o rotazione.

Verificare l'allineamento Al fine di verificare la corretta disposizione delle ancore, dopo aver disposto almeno un'ancora, sulla scena comparirà un modello 3D composto da un insieme di cubi allineati a formare una griglia regolare. Se le ancore sono state disposte correttamente, i cubi poggeranno sul pavimento; inoltre, dovranno rimanere relativamente stabili mentre l'utente si sposta da una parte all'altra della mappa.

3.4.3 Unirsi alla sessione di gioco

CheckingServer Dopo aver confermato la configurazione cliccando sul tasto Start Client, la informazioni inserite dall'utente sono salvate e racchiuse in un oggetto chiamato ClientInfo, quindi è lanciata la coroutine GetSessionInfo(), che esegue periodicamente una richiesta Web all'indirizzo IP del server per ottenere le informazioni della sessione corrente, se il server è attivo. Inoltre, il menù di configurazione è sostituito da un nuovo menù che include un testo per visualizzare le informazioni sul server e sulla sessione, un tasto, denominato Join Session, per unirsi alla sessione, che rimane inattivo finché il server non fornisce una risposta, e un tasto per tornare alla configurazione del client.

La richiesta Web è fatta da un Task dedicato, al fine di non bloccare il thread principale. Se il server risponde, l'utente può visualizzare la risposta sul menù stesso; inoltre, il tasto Join Session viene attivato, permettendo all'utente che lo desideri di unirsi alla sessione.

ConnectingToServer Quando l'utente è pronto, può premere il tasto Join Session, che avvierà la procedura per unirsi alla sessione. Tale procedura parte con la richiesta, tramite REST, di un Id, che permetterà al server di identificare in modo univoco il client. Ottenuto l'Id, il client ha un tempo prefissato per instaurare una connessione con il server.

In caso negativo, il client può ripetere la procedura di join richiedendo un nuovo Id.

SessionHandShaking Se il tentativo di connessione ha successo, il client e il server devono eseguire una procedura di *hand-shaking*, nel corso della quale si scambieranno le informazioni necessarie per il corretto funzionamento del sistema.

Il client invierà il messaggio di tipo ClientPresentation, mentre il server risponderà con il messaggio SessionDescription.

3.4.4 Preparazione alla sessione

Ricevuto il messaggio SessionDescription, il client entra nello stato **GettingReadyToStartSession** e carica, dalle risorse dell'applicazione, la mappa di gioco. In questa fase è definita anche la posizione di partenza dell'utente; un particolare oggetto, detto UserStartPoint, è posto nella posizione indicata nel messaggio SessionDescription.

Caso AR Se il dispositivo utilizzato è un casco AR, all'utente sarà richiesto di muoversi fisicamente nel punto in cui si trova lo UserStartPoint.

Inoltre, poiché egli si trova fisicamente nel luogo in cui si svolge l'azione, la ricostruzione 3D della mappa è disattivata.

Infine, il sistema attiverà le ancore spaziali che permetteranno l'allineamento del mondo reale e del mondo virtuale e la localizzazione dell'utente nel sistema di riferimento del server.

Caso VR Poiché l'utente VR è completamente immerso nel mondo virtuale, è possibile "teletrasportarlo" in qualunque punto, un approccio talvolta preferibile al movimento tramite ArmSwinger [3.2.9].

Nel momento in cui l'utente si unisce alla sessione, la stanza virtuale iniziale deve essere rimossa dalla scena e il giocatore deve essere spostato nella mappa di gioco; poiché l'utente deve partire da una posizione prestabilita, si è sfruttato il passaggio di scena per teletrasportare l'utente dalla stanza iniziale direttamente allo UserStartPoint.

Se l'utente si allontana troppo dalla posizione prestabilita, sarà riportato indietro tramite teletrasporto.

3.4.5 Avvio della sessione

Quando l'utente è pronto, invia al server un messaggio di tipo ClientReady, tramite il tasto Vote to Start Session; a questo punto, il client, che si è portato nello stato **WaitingForStartSessionInfo**, deve attendere che anche gli altri utenti votino per iniziare la sessione.

Il server da inizio alla sessione non appena riceve il messaggio ClientReady dall'ultimo client, inviando a sua volta il messaggio StartSessionInfo. Ogni client, ricevuto tale messaggio, azzera tutte le variabili dedicate alla gestione della sessione, tra cui il tempo che di sessione, che è sincronizzato con il tempo sul server.

L'applicazione crea un'istanza dell'entità di tipo UserActor (o classe analoga derivata da essa) controllata dal client e tutte le altre entità, siano esse controllate dagli altri client, oppure controllate dal server.

Infine, l'ActorController [3.2.10] è attivato per permettere all'utente di controllare il suo avatar.

3.4.6 Sessione di gioco

Inizialmente le entità controllate dai client sono inattive, in tal modo si lascia all'ApplicationMode la decisione se attivarle immediatamente, oppure se ritardarne l'attivazione, in funzione del tipo e delle regole della modalità; inoltre, un'entità potrebbe essere disattivata e riattivata più volte nel corso della stessa sessione (ad esempio, in un videogioco sparatutto le entità sono attivate dopo alcuni secondi, e possono "morire" e "rinascere". Al contrario, se la modalità fosse un laboratorio di chimica, le entità verrebbero attivate immediatamente e non verrebbero mai disattivate fino al termine della sessione).

Nella funzione Update() sono eseguite le principali operazioni per il corretto funzionamento della sessione:

a) Esecuzione locale[21] dei comandi che l'utente impartisce allo UserActor controllato.

b) Aggiornamento delle entità; la posizione e/o la rotazione delle entità è interpolata, come spiegato in [3.1.4].

c) Invio del messaggio PlaytimeInfo, contenente, tra le altre cose, i comandi del client.

d) Invio del messaggio ClientResearchInfo, contenente i dati raccolti per scopi di ricerca.

3.4.7 Session Status Board

Di tanto in tanto potrebbero verificarsi degli eventi di cui l'ApplicationMode vorrebbe informare gli utenti; ad esempio, in una ipotetica modalità di laboratorio virtuale di fisica, l'insegnante potrebbe voler comunicare un suggerimento su come procedere agli studenti.

A tal fine, è possibile disporre nella mappa dei tabelloni virtuali su cui possono essere riportati i messaggi e/o indicazioni forniti dall'ApplicationMode.

Gestione della perdita del tracciamento

Durante la sessione, poiché l'utente si sposta da una parte all'altra dell'area di gioco, può accadere che l'HoloLens non sappia più dove si trovi. Poiché, in tale evenienza, non è più possibile garantire la correttezza della posizione degli ologrammi, si è preferito nasconderli all'utente fino alla riacquisizione completa del tracciamento, intendendo con ciò che il sistema di tracciamento del dispositivo è di nuovo attivo, e che almeno un'ancora è stata individuata per riallineare i mondi virtuale e reale.

Dei messaggi informano l'utente della perdita tracciamento e della successiva riacquisizione.

3.4.8 Conclusione della sessione

La ricezione del messaggio SessionResults è interpretata dai client come indicazione che la sessione è terminata, quindi tutte le risorse allocate per la gestione della sessione sono rilasciate, e i componenti non più necessari, come l'Actor Controller, sono disabilitati, entrando nello stato **ShowingSessionResults**.

In un apposito menù sono presentati i risultati della sessione, il cui significato dipende dalla natura della modalità.

L'utente, premendo il tasto Join New Session, riporta il client nello stato CheckingServer, in cui è possibile tornare alla configurazione, o unirsi a una nuova sessione.

[21] Tipicamente, solo la parte di effetti grafici e sonori è eseguita in locale, dal momento che è il server ad avere l'autorità sui client sull'esecuzione dei comandi.

Capitolo 4

Shooter Game

Al fine di dimostrare la validità del framework, è stato sviluppato anche un videogioco che potesse metterne in luce pregi e difetti. Considerate le caratteristiche dei dispositivi impiegati, la maturità del framework, e il tempo a disposizione, si è ritenuto preferibile sviluppare un videogioco di genere sparatutto; infatti, tale genere, che richiede che i partecipanti si muovano continuamente nell'area di gioco, mette in luce alcuni aspetti fondamentali del framework e dei dispositivi AR e VR, quali la stabilità del tracking dell'HoloLens e i sistemi di locomozione per VR.

4.1 Modalità di gioco

Il videogioco, chiamato semplicemente Shooter Game, mette a disposizione due modalità; la prima è la classica modalità Deathmatch, in cui i giocatori si affrontano gli uni contro gli altri in un'arena. L'altra modalità vede, invece, i giocatori cooperare per distruggere dei mostri alieni. Tali modalità saranno, ora, discusse più in dettaglio.

4.1.1 Deathmatch

In questa modalità di gioco, il giocatore deve cercare di "uccidere" gli avversari, cercando a sua volta di non farsi uccidere. Infatti, ogni colpo ricevuto comporta una perdita di punti, al contrario un colpo inferto con successo all'avversario comporta l'ottenimento di un certo numero di punti.

Pistola virtuale e modalità di puntamento

L'arma a disposizione dei partecipanti è una pistola virtuale, cioè non è un oggetto fisico, come in ARQuake [43] o AR-Weapon [45].

L'utente che utilizza l'HoloLens, non avendo a disposizione un motion controller, farà uso dello sguardo per indicare la direzione nella quale vuole sparare; il comando di sparo è impartito tramite il clicker.

Anche il giocatore che indossa l'Oculus Rift ha nel suo arsenale la pistola laser, ma poiché tale dispositivo è accompagnato dai motion controller Oculus Touch, è possibile scegliere due modalità di puntamento:

a) Modalità a singolo tasto: tale schema di controllo imita il sistema AR costituito dall'HoloLens e il clicker, e permette di svolgere tutte le azioni che può compiere l'utente AR premendo un solo tasto.

b) Modalità a più tasti: in questa modalità, si trae vantaggio dall'avere il tracciamento della posizione e dall'avere due controller; l'utente può indicare al sistema, in fase di configurazione, la mano dominante. Con il controller corrispondente alla mano dominante, il giocatore può sparare, puntanto il controller come fosse una pistola, mentre con l'altra mano può interagire con gli oggetti della scena.

Starting Point

A ogni giocatore, nel momento in cui questi si unisce a una sessione di gioco, è assegnata, in modo casuale, una posizione iniziale nella mappa o arena di gioco. Le possibili posizioni di partenza, o Starting Point, sono scelte a priori dal designer dell'applicazione, al momento della costruzione del prefab della mappa virtuale.

Oltre alla posizione di partenza dei giocatori, il designer stabilisce la posizione degli altri elementi di gioco virtuali.

Entità di gioco

ShooterGameUserActor Nel gioco, il giocatore è rappresentato da un essere umanoide che ricorda un manichino (mostrato in figura 4.1), e che indossa il dispositivo che il giocatore reale sta realmente usando.

Il modello umanoide è animato utilizzando la cinematica inversa, in quanto il tracciamento della posizione e limitato alla testa e alle mani nel caso del VR, e alla sola testa nel caso dell'AR. Per animare il manichino, si è fatto uso di Root Motion, che include uno script appositamente pensato per la VR.

La classe madre che gestisce l'alter-ego del giocatore prende il nome di ShooterGameUserActor, ed è una classe derivata da UserActor, che costituisce la classe base per l'entità controllata da un utente remoto. La classe UserActor contiene, infatti, i metodi astratti per l'esecuzione di comandi locali e remoti.

La classe UserActor deriva a sua volta da Actor, che è stato separato da UserActor in quanto potrebbe essere utilizzato in futuro come base per implementare un'entità analoga allo UserActor, ma controllato dall'intelligenza artificiale[1].

La classe ShooterGameUserActor aggiunge alla classe base i metodi per eseguire gli effetti visivi e sonori dello sparo della pistola, quelli per permettere di ricaricare l'arma, e quelli per gestire la perdita e la ricarica della "energia vitale".

[1]Nell'accezione tipica dei videogiochi, cioè un insieme di regole che definiscono come l'entità si deve comportare in determinate condizioni

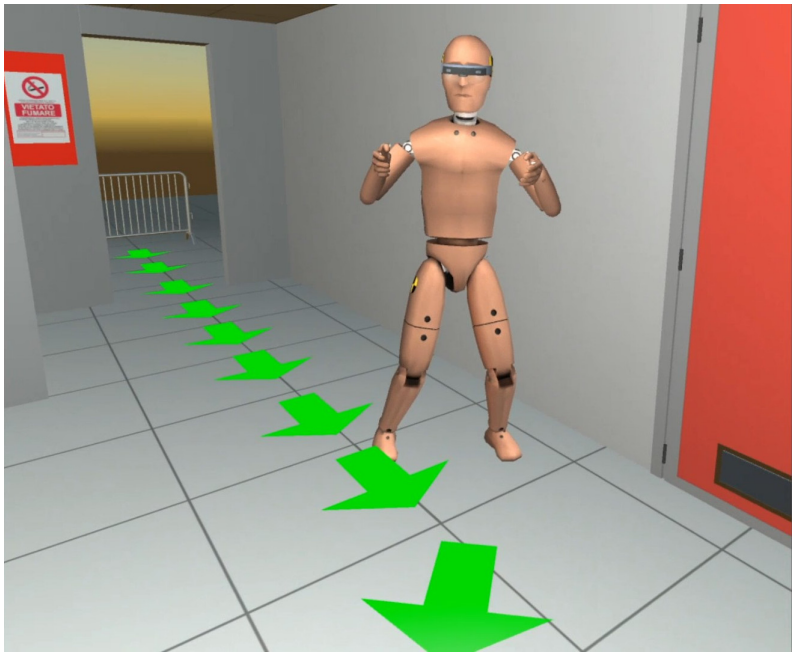

Figura 4.1. L'avatar con cui sono rappresentati i videogiocatori. Se l'utente sta utilizzando il dispositivo AR, l'avatar indossa l'HoloLens, se, invece, il dispositivo utilizzato è l'Oculus Rift, l'avatar indossa il visore VR.

Item Oltre che spararsi a vicenda, i giocatori possono anche raccogliere degli oggetti virtuali della scena che hanno un diverso effetto sull'avatar; un tale oggetto, che prende il nome di Item (così come la classe C# che lo implementa), è rappresentato da un modello 3D che indica quanto più chiaramente possibile la funzione dell'oggetto stesso.

Un Item implementa una particolare interfaccia C#, chiamata *IInteractable*, che presenta il seguente metodo:

```
public interface IInteractable
{
    void Use(Entity entity);
}
```

L'uso di un Item da parte di un'entità consiste nel chiamare la funzione Use(Entity entity), che applica un effetto sull'entità che ha chiamato la funzione.

Dopo l'utilizzo di un Item, questo si "esaurisce" e viene rimosso dalla scena; in base alle regole dell'ApplicationMode, l'Item può essere riattivato e rimesso in gioco o distrutto definitivamente.

L'Item, che è una classe astratta, contiene un metodo protetto astratto, void ApplyEffect(Entity entity), che deve essere definito dalla particolare classe che eredita da

Item. In tal modo, il codice degli oggetti utilizzabili come power-up è stato ridotto, evitando duplicazione di codice, che altrimenti avrebbe reso più probabile la presenza di bug.

La modalità di "consumazione" di un Item cambia in funzione del sistema di Mixed Reality:

a) Oculus Rift + Oculus Touch: l'utente "immerge" il controller nell'oggetto virtuale con cui vuole interagire e preme un apposito tasto.

b) HoloLens + Clicker: l'utente, che deve trovarsi al di sotto di una distanza predefinita, punta all'oggetto con lo sguardo e preme il tasto del clicker.

Medkit Uno degli elementi tipici dei videogiochi appartenenti al genere sparatutto, in particolare degli arena-shooter, è la presenza di Medkit, o oggetto analogo, che permette ai giocatori di rigenerare l'energia vitale. Nel gioco proposto i partecipanti possono trovare, distribuiti in tutta la mappa, dei Medkit opportunamente disposti dal designer.

Ammunition Alcuni videogiochi mettono a disposizione del giocatore armi con munizioni infinite, ma per il gioco qui discusso si è preferito dare al giocatore munizioni limitate, in modo da costringerlo a tenere conto di questo aspetto, e indurlo a esplorare l'ambiente alla ricerca di munizioni.

La classe Ammunition deriva da Item e implementa la funzione di ricarica della pistola.

4.1.2 Horde

La modalità Horde vede i giocatori cooperare per distruggere delle orde, o ondate, di nemici, che vengono generati in dei punti prefissati della mappa, che il designer può disporre in funzione delle esigenze di gameplay, ma anche dei limiti tecnici dei dispositivi impiegati. Si pensi, ad esempio, al meccanismo di riallineamento implementato e alle ancore spaziali [3.2.7]; alcuni punti di generazione dei nemici sono stati posti in corrispondenza delle ancore, affinché queste fossere all'interno del campo visivo dei sensori dell'HoloLens.

Entità di gioco

Le tipologie di entità di gioco sono le stesse della modalità Deathmatch, con l'aggiunta di una nuova tipologia, Enemy.

Enemy L'Enemy, come suggerisce il nome, è il nemico che i partecipanti al gioco devono affrontare e distruggere. Nel gioco esso è rappresentato come un mostro alieno (mostrato in figura 4.2), ed è in grado di muoversi e sparare.

Al fine di aiutare il giocatore a trovare i nemici, e rendere l'esperienza più coerente con la realtà, nel punto di generazione di ogni ondata è stato posto una sorta di tunnel/portale alieno virtuale, cioè un modello 3D, da cui i nemici fuoriescono.

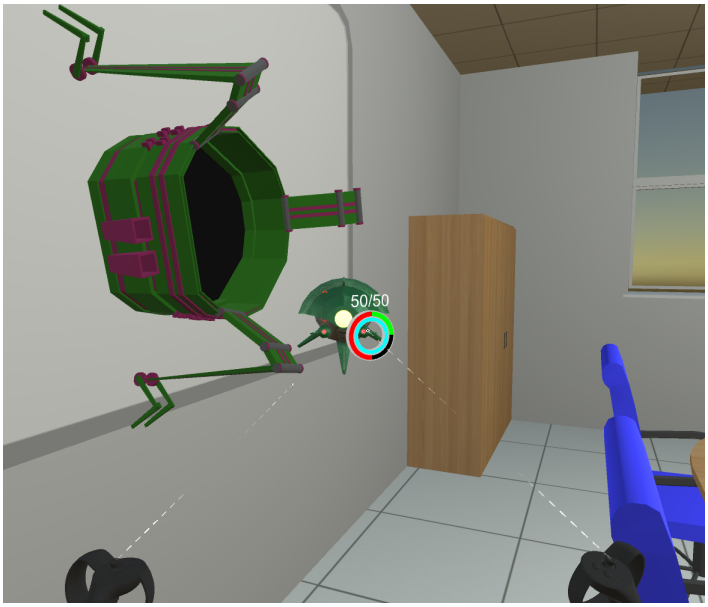

Figura 4.2. I mostri alieni che i giocatori sono chiamati a distruggere.

Nel momento in cui l'Enemy è generato dall'ApplicationMode, se non sono presenti ostacoli nel percorso che lo porta fuori dal tunnel, esso si porta fuori dal tunnel e, da questo momento, può decidere se sparare a un target, o muoversi per evitare di essere colpito in un raggio prefissato a partire dal centro del tunnel. Sia la velocità di spostamento, che la distanza massima dal centro del tunnel sono parametri di creazione dell'Enemy.

Per far sì che due o più nemici non si sovrappongano durante gli spostamenti, per ogni cammino possibile, scelto in modo casuale, si fa un test di collisione per accertarsi che non ci siano altri oggetti sul percorso candidato; se il test ha esito positivo, nel percorso è disposto un collider, al fine di "occupare" quel volume che sarà attraversato dall'entità. Tale operazione, così come la maggior parte dei calcoli per le altre entità, è effettuata sul server, in modo tale da ridurre il carico computazionale che i client devono sostenere.

Ondate e generazione dei nemici

Dopo alcuni secondi dall'inizio della sessione, l'ApplicationMode (o, più correttamente, classe Horde derivata da ApplicationMode) genera la prima ondata di nemici, nel punto specificato dal designer al momento della costruzione della mappa di gioco. Per ogni ondata, il numero, la velocità e il raggio di azione dei nemici cambiano, aumentando la difficoltà man mano che i giocatori avanzano nel gioco.

Se un nuovo nemico deve essere generato, l'ApplicationMode fa un controllo che il tunnel sia libero; in caso positivo, genera l'Enemy, altrimenti attende un po' prima di

eseguire nuovamente il controllo.

Quando un Enemy viene distrutto, l'ApplicationMode, che è in ascolto dei messaggi inviati dalle entità, riceve un messaggio di tipo DespawnMessage, che viene interpretato come entità distrutta. A questo messaggio l'ApplicationMode determina se è stato distrutto l'ultimo nemico dell'ondata corrente e se deve generare una nuova ondata (o terminare la sessione se tutte le ondate sono state affrontate).

Indicatori delle ondate Al fine di guidare l'utente verso l'obiettivo, sono state poste sul pavimento delle frecce virtuali lampeggianti che indicano dove si trova il punto di generazione dell'ondata corrente (si veda la figura 4.3).

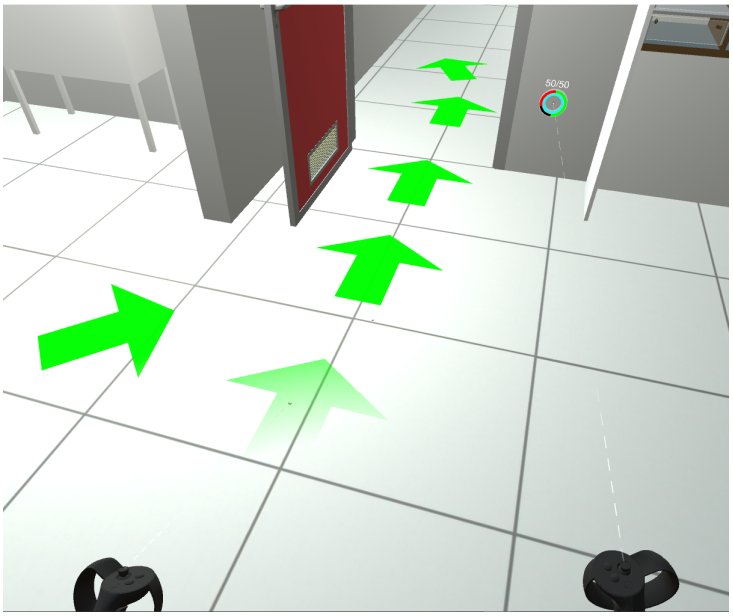

Figura 4.3. Le frecce verdi, poste a 10 cm dal pavimento, indicano dove si trovano i nemici.

Capitolo 5

Caso d'uso e test

In questo capitolo sono discussi dettagliatamente il caso d'uso e i test condotti per valutare la validità del framework sviluppato nell'ambito della presente tesi. Si parte con la descrizione del caso d'uso, quindi segue la presentazione dei test, giustificando le metodologie e discutendo, infine, i risultati ottenuti.

5.1 Finalità dei test

Lo scopo dei test è valutare diversi aspetti del framework e del videogioco sviluppato, con particolare attenzione rivolta a ciò che distingue le applicazioni AR/VR da quelle tradizionali, cioè l'interfaccia utente tridimensionale, il sistema di tracciamento e le modalità di interazione. Inoltre, poiché grande importanza riveste la funzionalità multiutente, si è ritenuto importante analizzare le variabili principali che caratterizzano le reti di calcolatori utilizzate, al fine di avere un quadro generale del sistema complessivo testato.

Altri aspetti del framework che si sarebbe voluto giudicare sono la semplicità nello sviluppo di un'applicazione, l'adattabilità a diversi tipi di applicazione, e la ricchezza di strumenti e funzioni offerti, ma ciò avrebbe richiesto l'esecuzione di test infattibili per un progetto di tale entità. Infatti, sarebbe stato necessario organizzare delle sessioni di sviluppo coinvolgendo un gruppo di programmatori e ingegneri informatici, a cui spiegare dettagliatamente la struttura del framework e le sue funzioni; quindi, ogni programmatore, dopo aver realizzato un'applicazione, avrebbe dovuto dare una valutazione del framework. A causa del tempo richiesto da test di tale portata, ci si è limitati a valutare gli altri aspetti precedentemente indicati.

Metodologia

Poiché una misurazione quantitativa diretta della bontà del framework e dell'applicazione realizzata non è possibile per molte delle caratteristiche di interesse, uno degli strumenti utilizzati per valutare il lavoro svolto per questa tesi è il questionario; per valutare l'interfaccia utente e le modalità di interazione, ma anche l'aspetto *sociale* dell'esperienza, o

la capacità dell'applicazione di raggiungere il suo obiettivo[1], ai tester sono state poste un certo numero di domande che toccano gli aspetti citati e non solo.

Gli aspetti che, al contrario, si prestano per misurazioni quantitative, sono stati valutati raccogliendo dati oggettivi nel corso dei test. Tra questi possiamo citare il funzionamento del sistema di tracciamento e la dinamica e l'approccio al gioco degli utenti.

Un sistema di tracciamento è caratterizzato principalmente dalla precisione della stima della posizione, dalla robustezza e dal volume dell'area in cui è possibile muoversi. È possibile che, in condizioni particolari, un sistema di tracciamento non sia più in grado di stabilire dove si trovi l'oggetto tracciato, a ciò ci si riferisce con l'espressione di perdita del tracciamento[2]. Quando ciò accade può essere utile sapere dove si trovava l'utente e in quale direzione stesse guardando; inoltre, sarebbe interessante sapere anche quanto tempo è impiegato dal sistema di tracciamento per *recuperare* dallo stato di inattività.

Tra i dispositivi testati, che saranno discussi più avanti, l'HoloLens è quello che presenta maggiori criticità in un contesto come quello qui discusso, in quanto il dispositivo non è stato progettato per scenari in cui l'utilizzatore deve allontanarsi per oltre 5 metri dall'origine della scena virtuale. Al fine di studiare la robustezza del sistema di tracciamento di HoloLens, sono stati raccolti i dati di posizione e orientamento dell'utente e il tempo necessario affinché il sistema di tracciamento fosse di nuovo in grado di tracciare correttamente l'utente.

Infine, per quanto riguarda la dinamica e l'approccio al gioco dei partecipanti, si è pensato di raccogliere i dati sulla posizione nell'area di gioco in ogni istante, al fine di capire da cosa i tester fossero attratti; ad esempio, dai tracciati degli spostamenti (si vedano le figure 5.2 e 5.3), è possibile capire se i partecipanti utilizzano gli item disponibili (Medkit e munizioni), o anche dove tendono a posizionarsi in relazione ai nemici.

5.2 Test

Il caso d'uso tipico per un'applicazione multiutente prevede che un certo numero di individui partecipino alla sessione, in funzione del tipo di applicazione: ad esempio, in una partita a scacchi il numero di giocatori è 2, mentre in uno sparatutto a squadre il numero di giocatori non deve essere inferiore a 4. A causa della difficoltà nell'organizzare test con un numero relativamente grande di tester, si è scelto di limitare il numero di tester a 2 soggetti per volta. I test, condotti all'interno del Politecnico di Torino, hanno coinvolto 10 gruppi da 2 persone, per un totale di 20 persone.

[1] Nel caso in esame l'applicazione, essendo un videogioco, dovrebbe essere divertente e interessante

[2] Dall'inglese *lost tracking*

Il terzo piano del Dipartimento di Automatica e Informatica (DAUIN) è stato scelto come area di gioco[3]; in particolare, l'area di gioco comprendeva la sala riunioni 1, il corridoio esterno ed eventualmente altri locali resi accessibili appositamente per il test. Mentre l'utilizzatore del dispositivo AR può muoversi fisicamente nell'area di gioco, l'utilizzatore del dispositivo VR potrà muoversi virtualmente all'interno di una ricostruzione[4] al computer del terzo piano del DAUIN.

A ogni coppia di tester, che venivano fatti accomodare nella sala riunioni 1, è stato dapprima spiegato l'obiettivo del videogioco, cioè distruggere dei mostri all'interno dell'area di gioco, sottolineando, inoltre, che si trattasse di un'esperienza cooperativa e che potessero muoversi non solo all'interno della sala, ma anche lungo il corridoio ed eventualmente in altri locali resi accessibili appositamente per il test. Dopo di ciò, veniva spiegato il funzionamento dell'applicazione, a partire dall'accesso alla sessione di gioco, fino al termine della partita.

Per ogni dispositivo era spiegato come muoversi, come sparare e come utilizzare gli item disseminati nell'area di gioco, indicando quale tasto premere per compiere una data azione. Prima dell'inizio della partita, all'utilizzatore del dispositivo VR veniva data la possibilità di provare la meccanica di locomozione dell'Arm Swinger (si veda 3.2.9), all'interno di una stanza virtuale appositamente creata per prendere dimestichezza con il sistema VR.

Prima di iniziare la sessione, i tester erano avvertiti del fatto che l'HoloLens potesse in alcuni momenti perdere il tracciamento; inoltre, era spiegato loro che in tale evenienza avrebbero dovuto guardarsi intorno finché il sistema non li avesse avvertiti che il tracciamento era tornato attivo.

Dopo aver indossato i visori AR e VR e aver preso dimestichezza con i dispositivi di controllo, i tester erano guidati nella procedura di connessione al server. All'avvio della partita, ogni partecipante deve trovarsi in un punto, detto di *spawning*, preso casualmente in un insieme di punti prestabiliti dell'area di gioco. Mentre nel caso VR, dopo la connessione, l'utente VR è automaticamente teletrasportato nel punto di spawning, l'utilizzatore del dispositivo AR deve andare fisicamente nel punto designato, e rappresentato virtualmente[5] da una cella verde.

Quando i tester erano pronti, era chiesto loro di votare per iniziare a giocare; per votare, l'utente doveva premere un opportuno pulsante virtuale, in tal modo il client informava il server che era pronto per iniziare la sessione.

[3] L'area di gioco è anche il luogo in cui si è svolto il test, in quanto l'utente che utilizza il dispositivo per realtà aumentata deve trovarsi fisicamente nell'area di gioco

[4] La ricostruzione 3D è stata realizzata da un altro studente nell'ambito della sua tesi

[5] Cioè da un ologramma

5.2.1 Descrizione del sistema

Il sistema utilizzato durante i test includeva un PC desktop equipaggiato con un processore Intel Core i7, scheda grafica Nvidia Geforce 1060 da 6 GB di memoria video dedicata e 16 GB di DRAM; il desktop eseguiva sia l'applicazione server, sia il client VR. Per connettere gli host, è stato utilizzato un router al quale il PC era connesso tramite cavo ethernet; la rete locale era ad uso esclusivo degli host coinvolti nei test.

Il visore VR scelto per i test è l'Oculus Rift, in quanto supporta il tracciamento a 6 gradi di libertà, necessario per poter interagire efficacemente con gli oggetti virtuali e potersi muovere nella scena tramite l'oscillazione delle braccia, grazie anche ai controller ufficiali Oculus Touch. Al fine di rendere più robusto il tracciamento, sono stati utilizzati due sensori, entrambi posti di fronte all'utente, a circa 1 metro di distanza l'uno dall'altro. Poiché con tale configurazione si ha una copertura di circa 180°, agli utenti è stato consigliato di non ruotare fisicamente, ma solo virtualmente utilizzando il sistema di locomozione implementato nel framework.

Il dispositivo AR utilizzato è l'HoloLens, in quanto supporta le funzionalità di interazione e tracciamento minime previste per il framework, nonché è il dispositivo per il quale il framework è stato pensato. A differenza del PC, l'HoloLens era connesso alla rete locale tramite wifi. Per semplificare l'interazione, con particolare riferimento all'azione dello sparare con l'arma virtuale, è stato fornito all'utente il clicker; infatti, uno degli svantaggi di utilizzare i gesti è la necessità di tenere le mani nel campo visivo delle camere del visore, riducendo in tal modo il campo visivo dell'utente, mentre i comandi vocali sono meno immediati della semplice pressione del tasto del clicker.

Nel corso dei test, era proiettato sul monitor del PC ciò che l'utilizzatore dell'Oculus Rift vedeva dentro il visore; inoltre, il server forniva una vista dall'alto della scena, permettendo di controllare il corretto funzionamento del sistema.

5.2.2 Questionario

Il questionario [A] è suddiviso in un preambolo, in cui al tester è chiesto di rispondere al alcune domande preliminari che ne inquadrano il tipo, e 4 sezioni, che trattano aspetti specifici del sistema in esame. Ogni sezione contiene 2 colonne, una da compilare subito dopo la prova del sistema con il dispositivo AR, e l'altra da compilare dopo la prova con il dispositivo VR.

Il questionario deve essere compilato subito dopo il test perché, in caso contrario, l'opinione dell'utente potrebbe cambiare, rendendo i dati raccolti non più attendibili. Le ragioni per cui un soggetto può cambiare il suo giudizio sono molteplici:

1. Discutere con altre persone può modificare l'impressione iniziale in positivo o in negativo.

2. Riflettere sull'esperienza provata, può portare a cambiare idea su alcuni aspetti del sistema; ad esempio, il soggetto potrebbe essere portato a "premiare" chi ha realizzato il sistema, rendendosi conto delle difficoltà che quest'ultimo ha incontrato nel realizzare il sistema.

3. Il soggetto potrebbe non ricordare esattamente i dettagli dell'esperienza. Ad esempio, nel caso in esame, ogni utente prova in successione due esperienze molto simili, e ciò potrebbe portare a confondere il giudizio di una esperienza con l'altra.

Al termine del questionario sono presenti degli spazi per i commenti liberi, in cui il tester può indicare quali aspetti non gli sono piaciuti, quali non hanno funzionato, e ciò che avrebbe voluto o si aspettava di trovare.

Preambolo

Con il preambolo si cerca di inquadrare il background del soggetto, al fine di pesare il giudizio espresso sul sistema provato in funzione delle conoscenze e capacità dichiarate. All'utente è chiesto di indicare l'età e il sesso, che possono essere degli utili indicatori dell'abilità nell'utilizzo dei videogiochi (una domanda verte specificatamente sui videogiochi, chiedendo quanto spesso l'utente giochi) e dispositivi tecnologici in generale.

Le domande successive vertono, invece, sulla conoscenza della realtà aumentata e virtuale, sia in termini di frequenza di uso di applicazione per AR/VR, sia in termini di frequenza di uso di visori[6] AR/VR.

Ad eccezione delle domande sull'età e il sesso, ogni altra domanda richiede una risposta con un numero che va da 0 (mai) a 4 (ogni giorno).

Sezione 1: game experience

La prima sezione del questionario riguarda la *game experience* in generale, e contiene 33 affermazione a cui l'utente può indicare il grado di accordo, con un numero che va da 0 (per niente) a 4 (estremamente). Gli aspetti analizzati sono il divertimento e la piacevolezza dell'esperienza (in termini di ricchezza ed estetica), la difficoltà incontrata e la percezione della propria abilità nel giocare, il coinvolgimento nella storia e l'attenzione a ciò che succedeva nel corso dell'esperienza.

Alcune affermazioni sono appositamente simili nel concetto espresso, oppure esprimono giudizi opposti, al fine di verificare che le risposte siano date con attenzione e non in modo casuale.

Sezione 2: social presence

Questa sezione è incentrata sugli aspetti sociali dell'esperienza, ed è stata inclusa nel questionario in quanto l'applicazione provata è multiutente e ai partecipanti è chiesto di cooperare per raggiungere l'obiettivo comune.

[6] Head-mounted display

Essa include 17 affermazioni, a cui il tester deve indicare il grado di accordo, che coprono tematiche quali l'empatia e la connessione con gli altri giocatori, la capacità di influenzare o essere influenzato dalle azioni degli altri, e la percezione della presenza degli altri giocatori come persone reali.

Sezione 3: post-game

La sezione 3 tratta l'impressione dell'esperienza e lo stato psicofisico del tester al *termine* del test; le affermazioni vertono sul carico cognitivo e fisico richiesto durante l'esperienza (stanchezza fisica e mentale), eventuale vergogna e sensi di colpa provati, difficoltà a tornare nella realtà (particolarmente importante nel caso della realtà virtuale).

Sezione 4: system usability scale

Il *system usability scale* (SUS) è un metodo affidabile e molto usato per valutare l'usabilità di un sistema, considerando aspetti quali l'efficacia, l'efficienza, la semplicità e la coerenza. Esso consiste in un insieme di 10 affermazioni a cui l'intervistato può esprimere il grado di accordo attraverso un numero da 0 a 4, dove 0 indica completo disaccordo con quanto affermato, mentre 4 completo accordo.

Il valore numerico con cui i soggetti intervistati esprimono il proprio accordo rispetto alle affermazioni del SUS può essere convertito in un aggettivo (la scala è mostrata in figura 5.1) che qualifica il sistema in un modo un più sintetico e comprensibile. La conversione richiede che si complementino, dapprima, i punteggi delle domande con accezione negativa (domande pari in A.4), quindi si ottiene il punteggio complessivo applicando la formula seguente:

$$Score_{SUS} = (\sum_{i=1}^{10} score_i) * 2.5$$

In tal modo si ottiene un punteggio tra 0 e 100, che può essere convertito in un *adjective rating* tramite la relazione mostrata in figura 5.1.

5.2.3 Risultati

Ai test hanno partecipato sia uomini che donne (la percentuale delle donne è molto inferiore a quella degli uomini e pari al 20%), in una fascia d'età compresa tra i 19 e i 30 anni (età media pari a 25,6 anni). La maggior parte dei partecipanti studiano o lavorano nel campo dell'ingegneria informatica, o comunque in campi affini. Il 20% dei tester dichiara che usa applicazione AR almeno 1 volta a settimana, mentre tale percentuale sale al 50% per applicazioni VR. Per quanto riguarda i videogiochi in generale, il 50% dei soggetti dichiara di giocare regolarmente, più giorni della settimana.

Dati soggettivi

Nelle tabelle 5.1 e 5.2 sono riportati i dati ottenuti per le prime 3 sezioni del questionario. Si ricorda che il punteggio va da 0 a 4, dove 0 indica il completo disaccordo con

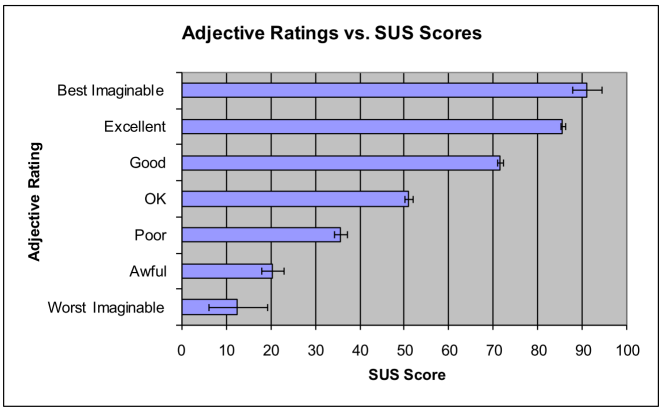

Figura 5.1. Relazione aggettivo qualificante e punteggio numerico SUS. Grafico tratto da [5].

l'affermazione, mentre 4 indica il completo accordo. Nel calcolare la media, la deviazione standard e la mediana, il valore assegnato dall'intervistato è stato lasciato invariato per le affermazioni con accezione positiva, invece è stato preso il suo complemento a 4[7] per quelle con accezione negativa.

Come è possibile vedere, sia l'esperienza in realtà aumentata che quella in realtà virtuale hanno ricevuto valutazioni positive per quanto riguarda la game experience e il post-game, con un punteggio medio pari a 3 e deviazione standard di circa 0.3. Al contrario, la social presence ha ricevuto un punteggio più basso, pari a 1.3 per AR e 1.4 per VR, probabilmente dovuto al fatto che il videogioco proposto non prevede interazioni dirette tra i partecipanti.

Infine, si può notare che le valutazioni non sono influenzate dal fatto di aver provato prima un dispositivo piuttosto che l'altro, infatti la differenza tra le medie è al di sotto del margine di incertezza.

SUS Limitatamente alle valutazioni ricevute, il sistema si colloca, dal punto di vista dell'usabilità, tra **OK** e **Excellent**, avendo raggiunto un punteggio pari a 76,25±11,5 (76±15,9 per VR).

Campo visivo limitato Dopo aver provato il dispositivo AR, una parte significativa dei tester ha lamentato la difficoltà a vedere le indicazioni presenti e gli elementi virtuali in generale, a causa del campo visivo limitato dell'HoloLens. Pur essendo vero che questa è una limitazione hardware del dispositivo impiegato, e che nel prossimo futuro la situazione dovrebbe cambiare in meglio (Basti pensare alla nuova versione dell'HoloLens, che offrirà un campo visivo quasi doppio), si sarebbe potuto studiare meglio la posizione degli indicatori, per renderli più visibili; ad esempio, le frecce che indicano i nemici potevano

[7] Ad esempio, se $v = 3$, il suo complemento $v_c = 4 - 3 = 1$

	Sezione		
Punteggio AR	Game experience	Social presence	Post-game
Media	2,798484848	1,288235294	2,932352941
Deviazione standard	0,361645434	0,543568881	0,356489516
Mediana	2,757575758	1,382352941	2,941176471
Media (prima AR)	2,796969697	1,129411765	3,023529412
Deviazione standard	0,356250755	0,568370463	0,385911046
Mediana	2,757575758	1,264705882	3
Media (prima VR)	2,8	1,447058824	2,841176471
Deviazione standard	0,366954558	0,466304028	0,297684257
Mediana	2,757575758	1,529411765	2,794117647

Tabella 5.1. Valutazioni ottenute nelle prime 3 sezioni del questionario per la prova con il dispositivo AR.

	Sezione		
Punteggio VR	Game experience	Social presence	Post-game
Media	3,06969697	1,429411765	3,035294118
Deviazione standard	0,348300346	0,602216207	0,403873974
Mediana	3,045454545	1,352941176	3,058823529
Media (prima AR)	3,142424242	1,358823529	3,182352941
Deviazione standard	0,374104379	0,690744546	0,404516029
Mediana	3,166666667	1,323529412	3,117647059
Media (prima VR)	2,996969697	1,5	2,888235294
Deviazione standard	0,303469379	0,48809353	0,345459926
Mediana	3	1,441176471	3,029411765

Tabella 5.2. Valutazioni ottenute nelle prime 3 sezioni del questionario per la prova con il dispositivo VR.

essere poste sulle pareti ad altezza uomo. Inoltre, sarebbe stato possibile implementare degli indicatori dinamici che diventino visibili quando l'oggetto target non è più nel campo visivo, indicandone la direzione.

Cooldown Il meccanismo di cooldown dell'arma da fuoco virtuale non è stato particolarmente apprezzato da alcuni tester, che avrebbero preferito la possibilità di sparare alla frequenza con la quale premevano il tasto per fare fuoco. Non è chiaro se eliminare il

cooldown sia la scelta giusta, o se sia più opportuno limitarsi a ridurre il tempo di cooldown. Infatti, il cooldown aiuta a dare consistenza alle dinamiche del gioco, rendendolo più simile alla realtà vera.

Difficoltà a puntare con HoloLens Alcuni utenti hanno dichiarato di aver trovato più difficoltoso puntare e sparare con lo sguardo[8], il sistema di controllo utilizzato con il dispositivo AR, rispetto a puntare con il motion controller VR. Ciò è probabilmente dovuto al fatto che i nemici si muovono continuamente, rendendo il puntamento con lo sguardo non sufficientemente rapido e stancante.

Effetti sonori e visivi dello sparo Un aspetto su cui sarebbe stato opportuno dedicare più attenzione, e che è stato sottolineato dai tester, è la dinamica dello sparo; infatti, sebbene fosse possibile vedere lo sparo come nei film di Star Wars©, accompagnato da un effetto sonoro, molti hanno segnalato la difficoltà a capire se un colpo fosse andato a segno, a causa della mancanza di un feedback sonoro o visivo nel momento in cui il bersaglio era colpito.

Interazioni con il clicker Il meccanismo di raccolta degli oggetti virtuali (Medkit e munizioni) implementato per HoloLens, che richiedeva di guardare l'oggetto da raccogliere e premere contemporaneamente il tasto del clicker, è stato ritenuto meno immediato del meccanismo, più naturale, implementato per la modalità VR, che richiedeva di avvicinare il controller all'oggetto da raccogliere e premere un tasto predefinito. Benché quest'ultimo meccanismo sia più immediato, non è stato possibile replicarlo con il clicker dell'HoloLens, in quanto esso non è tracciato.

Come per la raccolta degli item, anche la modalità di sparo con il clicker non è stata molto apprezzata da alcuni utenti, preferendo la modalità VR, in quanto ritenuta più appagante e immersiva.

Comunicazione con i compagni Durante i test, alcuni partecipanti parlavano tra loro, manifestando il desiderio di poter comunicare direttamente attraverso la propria voce; inoltre, una parte degli intervistati ha indicato specificatamente l'implementazione di un sistema (tipo VoIP[9]).

Menù Un altro aspetto da migliorare è il modo in cui sono gestiti i menù, in quanto sono stati ritenuti troppo invasivi. Infatti, affinché il menù fosse sempre nelle vicinanze dell'utente, è stato implementato un meccanismo per cui il menù segue sempre l'utente mentre questo si sposta nella scena. Purtroppo, la particolare implementazione di tale meccanismo ha comportato una riduzione del campo visivo. Si potrebbe migliorare la

[8] Più correttamente, il puntatore segue la direzione della testa, in quanto l'HoloLens non supporta l'*eye tracking*

[9] Voice over IP

logica sottostante per far sì che durante gli spostamenti il menù segua l'utente, ma senza bloccargli la visuale; in alternativa, si potrebbe rendere il menù fisso, e permettere all'utente, tramite la pressione di un tasto, di richiamare il menù.

Delimitatori dell'area di azione Per segnalare il perimetro dell'area di azione, sono stati posti degli oggetti virtuali (transenne) lungo il corridoio. Durante uno dei test con l'HoloLens è successo che il tester non ha visto l'ostacolo virtuale, ed ha proseguito ben oltre l'area di gioco. In tale evenienza, sarebbe necessario un meccanismo che permetta di segnalare all'utente che è al di fuori del perimetro che delimita la scena di interesse, e lo guidi nella giusta direzione per riportarsi nell'area ammessa.

Dati oggettivi

Punti di interesse Le figure 5.2 e 5.3 mostrano una vista dall'alto dell'area di azione; le parti rappresentate dal pattern a scacchiera rossa e bianca sono i locali inaccessibili. I cerchi verdi tratteggiati indicano i punti di spawning degli utenti, mentre le linee rosse tratteggiate indicano la posizione dei delimitatori dell'area di azione (nel videogioco proposto, delle transenne virtuali). Le entità virtuali con cui si può interagire (Nemici, medkit e munizioni) sono indicati nella mappa. Il tracciato raffigurato è stato ottenuto dalle posizioni degli utenti campionate nel corso dei test; le parti rosse e gialle sono le zone in cui i tester si sono trattenuti per maggior tempo, al contrario la parte blu indica zone meno frequentate.

Come è possibile notare, i partecipanti si sono interessati non solo ai nemici da distruggere, ma anche agli item da raccogliere.

Perdita del tracciamento La figura 5.4 illustra gli eventi di perdita di tracciamento verificatisi durante i test con l'HoloLens. Le frecce indicano la posizione e la direzione dell'utente quando si è verificata la perdita di tracciamento, mentre i tre rettangoli posti in corrispondenza di ciascuna freccia rappresentano, codificati con i colori rosso, verde e blu, la velocità (rosso), il tempo trascorso dal momento in cui avviene la perdita di tracciamento e la successiva riattivazione del sistema di tracciamento (verde), e il tempo che l'HoloLens ha impiegato per rilocalizzare almeno un'ancora (blu).

Le ancore, rappresentate nella figura e evidenziate con cerchi tratteggiati, sono state distribuite nell'area di interesse quanto più uniformemente possibile.

Come è possibile notare dalla figura, le perdite di tracciamento si sono verificate anche quanto gli utenti guardavano in direzione delle ancore; ciò è probabilmente dovuto al fatto che il sistema di rilevazione delle ancore è basato sullo stesso principio del sistema di tracciamento. Quindi, la presenza delle ancore non può aiutare il sistema di tracciamento dal punto di vista della robustezza, può, però, essere impiegata per l'allineamento degli ologrammi con il mondo reale, come è stato fatto nel presente lavoro.

La velocità, calcolata dai valori di posizione, ha valore medio pari a 0.12 m/s, e deviazione standard pari a 0.08 m/s. Il rettangolo a sinistra è rosso se la velocità è maggiore

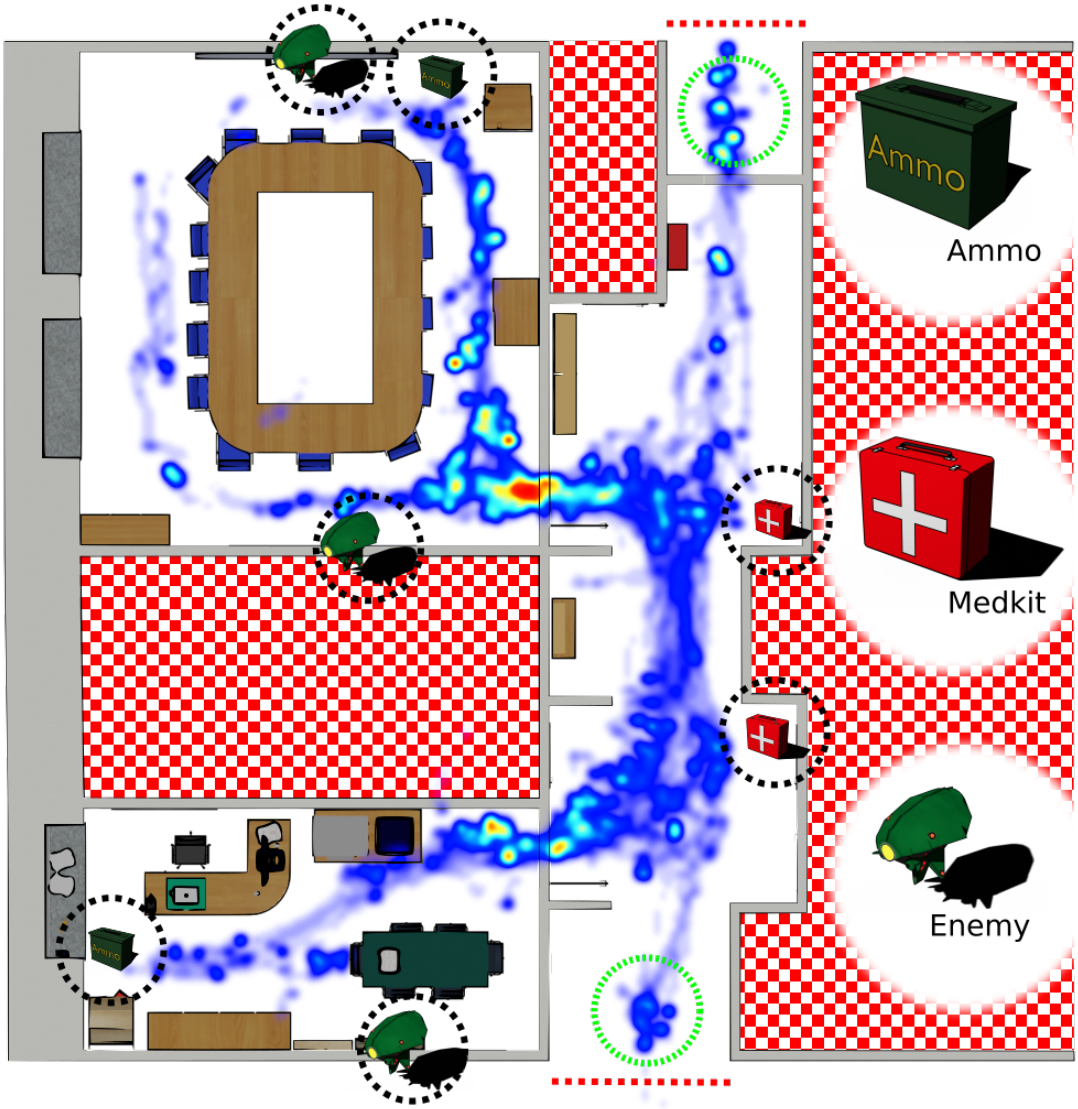

Figura 5.2. Mappa di calore dell'HoloLens. Le aree rosse sono quelle in cui il soggetto ha trascorso più tempo, mentre le aree blu sono quelle in cui ha trascorso meno tempo.

o uguale a 0.12 + 0.08 = 0.2 m/s, mentre è nero se la velocità è inferiore a 0.12 - 0.08 = 0.04 m/s.

Il rettangolo al centro rappresenta, come detto sopra, il tempo trascorso dall'istante in cui si verifica la perdita di tracciamento, e quello in cui il sistema di tracciamento segnala

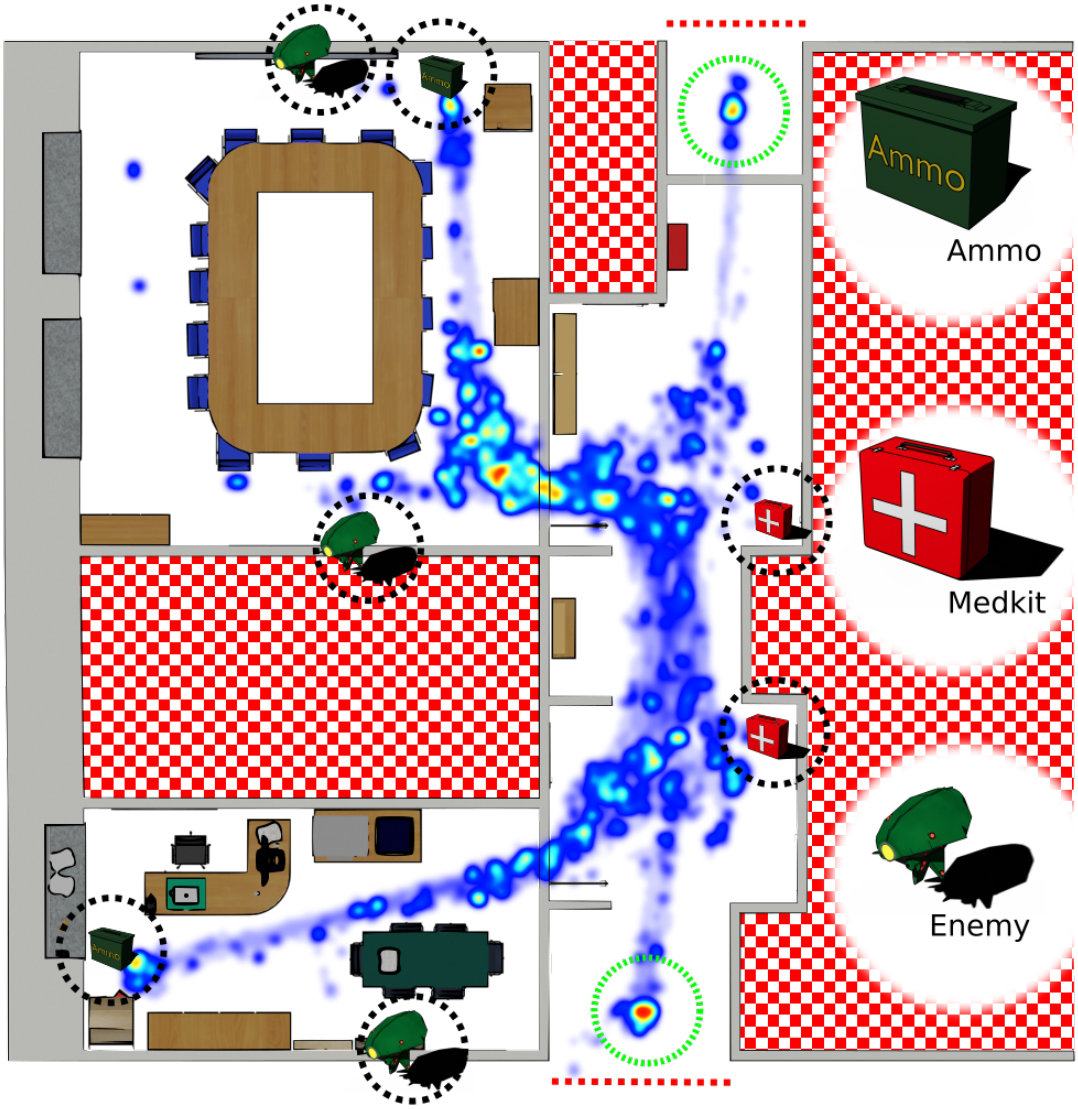

Figura 5.3. Mappa di calore dell'Oculus Rift.

stato attivo[10]. Il valore medio è pari a 602 ms, mentre la deviazione standard è pari a 83 ms.

[10]Quando il sistema di tracciamento è attivo, l'HoloLens è in grado di tracciare la propria posizione, ma potrebbe non aver localizzato alcuna ancora

Infine, il tempo che il sistema impiega per recuperare completamente (Cioè essere in grado di tracciare la posizione dell'utente e contemporaneamente allineare gli ologrammi con il mondo reale) ha valore medio pari a 5063 ms e deviazione standard pari a 2305 ms.

Come si evince dai dati raccolti, il tempo di riattivazione è quasi costante, mentre il tempo di recupero completo dipende molto da dove si trova l'utente in relazione alle ancore.

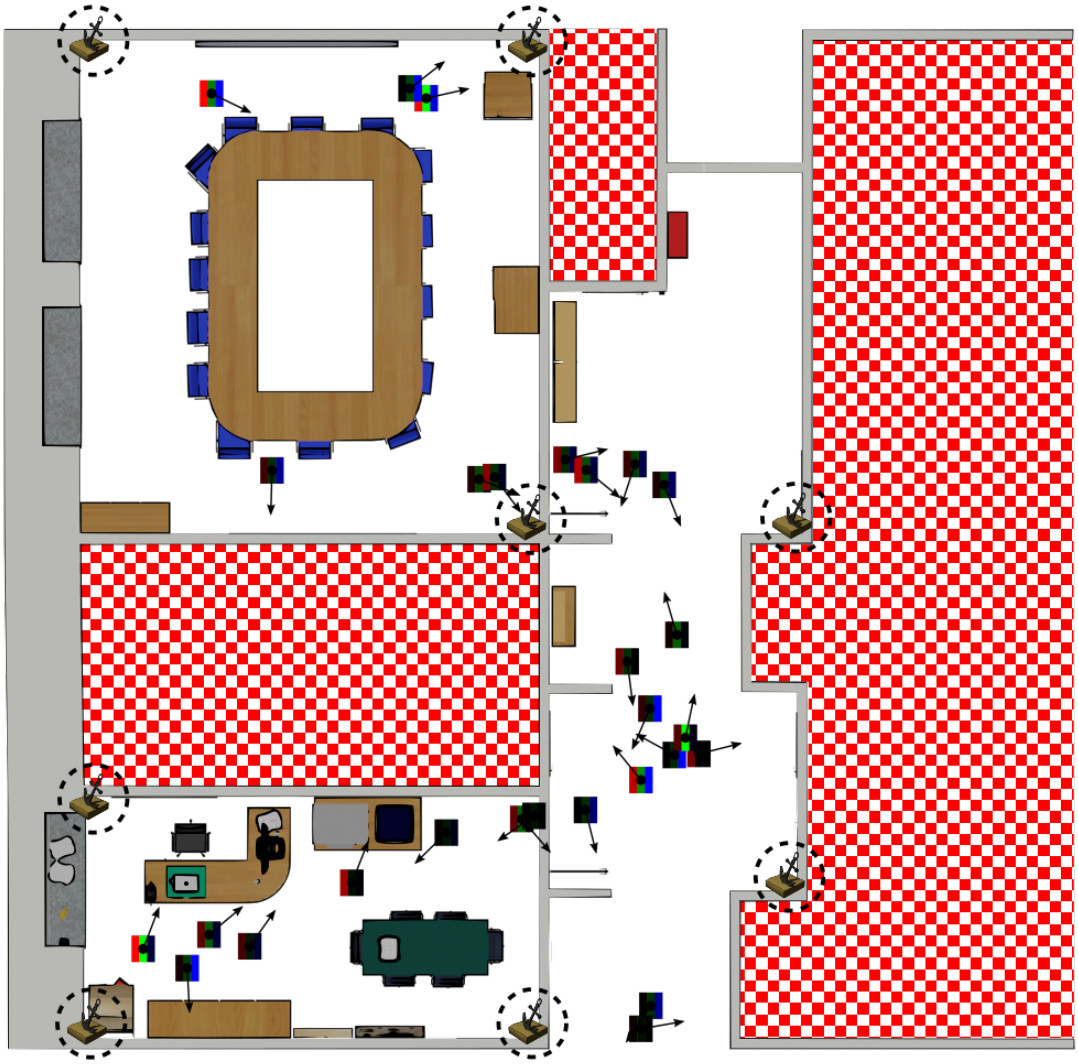

Figura 5.4. Ogni freccia indica la posizione e la direzione dell'HoloLens nel momento in cui si è verificata una perdita di tracciamento. La velocità e il tempo di recupero sono codificati tramite rettangoli colorati.

Capitolo 6

Conclusioni e sviluppi futuri

Lo scopo del lavoro qui presentato è stato la realizzazione di un framework che potesse costituire un punto di partenza per lo sviluppo di ambienti di realtà aumentata e virtuale multiutente, orientato principalmente all'intrattenimento.

Nei mesi dedicati a lavorare al framework, sono stati trattati molteplici argomenti, che vanno dai modelli di reti di calcolatori, alle interfacce utente, dalla modellazione 3D ai *pattern* di programmazione. Particolare importanza ha avuto il tema della rete; aspetti quali la creazione e la gestione di sessioni, le procedure di accesso a sessioni aperte, la comunicazione tra gli host e la sincronizzazione delle entità hanno richiesto grande attenzione, al fine di rendere il framework il più flessibile possibile per lo sviluppatore, nonché rendere l'esperienza di utilizzo dell'applicazione da parte dell'utente finale semplice ed intuitiva.

La flessibilità del framework è stato uno dei principali requisiti; si è cercato, infatti, di progettare un'architettura che potesse adattarsi a modalità di funzionamento diverse, in cui diversi tipi di entità devono raggiungere gli scopi definiti dal designer, seguendo le regole che questi ha stabilito.

La sincronizzazione dello stato delle entità tra i diversi host è stato un aspetto critico per due ragioni:

1. L'utilizzo di un protocollo non affidabile (per la motivazione dietro tale scelta, si veda 3.1.2) richiede di gestire casi quali perdita di pacchetti e arrivi fuori ordine.
2. L'aggiunta di un nuovo tipo[1] di entità può richiedere più o meno lavoro al programmatore in funzione della strategia di sincronizzazione adottata. Al fine di ridurre il numero di passaggi richiesti al programmatore per la sincronizzazione, e ridurre in tal modo anche la probabilità di commettere errori, si è fatto lo sforzo di implementare a livello di sistema i meccanismi di sincronizzazione.

Il framework sviluppato è composto da un certo numero di moduli e componenti indipendenti, che comunicano e interoperano sotto la regia di un ApplicationController (si veda 3.2 per maggiori dettagli), che gestisce lo stato dell'applicazione.

[1] Classe

Poiché il framework è stato pensato per il modello client-server, esso implementa, a livello di sistema, tutte le procedure indispensabili sia del server (configurazione, avvio, aggiornamento, e terminazione di una sessione) che dei client (configurazione, accesso e gestione di una sessione).

Al fine di valutare la validità del framework, è stato sviluppato un videogioco in cui i partecipanti hanno l'obiettivo di distruggere dei mostri alieni disseminati all'interno di un'area di gioco prestabilita.

Il videogioco ha permesso di valutare non solo alcuni aspetti generali del framework, come il sistema di allineamento discusso nel capitolo 3.2.8, ma anche aspetti più specifici delle applicazioni per AR e VR.

I test sono stati condotti all'interno del Politecnico di Torino, al Dipartimento di Automatica e Informatica, in un'area che comprende due locali e il corridoio che li connette; i sistemi impiegati comprendono un PC desktop utilizzato come server, l'HoloLens, e un router. Il PC desktop, connesso tramite cavo ethernet al router, eseguiva anche un client con i dispositivi Oculus Rift e Oculus Touch, mentre l'HoloLens era connesso al router tramite wifi.

Al fine di valutare tutti gli aspetti del sistema, non solo tecnici, ma anche quelli relativi alle funzionalità e alla game experience, i test sono stati condotti in gruppi di 2 individui, che si alternavano provando entrambi i dispositivi di AR e VR.

Benché i test condotti abbiano mostrato che il sistema nel suo complesso funzioni, alcuni aspetti fondamentali del framework possono essere migliorati, a partire dall'algoritmo di allineamento, che potrebbe essere reso più robusto sfruttando tecniche che si basano sull'estrazione di spigoli [25], o anche riconoscimento di *marker*.

Un altro aspetto migliorabile è la segnalazione dei limiti dell'area di azione degli utenti, che, specialmente in realtà aumentata, presenta dei difetti di progettazione, come discusso in 5.2.3. Oltre a delimitatori sotto forma di oggetti reali, quali transenne virtuali, si potrebbero utilizzare segnalatori dinamici, attivati non appena l'utente esce dall'area di azione e posti dinamicamente di fronte all'utente, in modo da essere sempre visibili.

Durante le sessioni di test, è stato lamentato il fatto che i menù fossero troppo invasivi (si veda 5.2.3), a causa del fatto che, seguendo continuamente l'utente durante gli spostamenti, ne limitavano il campo visivo. Per evitare che il menù blocchi la visuale dell'utente, si potrebbe rendere più intelligente il meccanismo di *following* dell'utente, o anche assegnare un tasto per richiamare il menù *on demand*.

Per quanto riguarda gli aspetti prettamente di design dell'applicazione, i partecipanti hanno segnalato la difficoltà a individuare la posizione degli obiettivi (si veda 5.2.3), in quanto gli indicatori, posti all'altezza del pavimento per non essere troppo invasivi, risultavano difficili da vedere con l'HoloLens, a causa del limitato campo visivo. Al fine di rendere gli indicatori più visibili, sarebbe preferibile porli ad altezza d'uomo e distribuirli in modo più intelligente, per non distrarre gli utenti durante l'azione.

In generale, questa prima versione del framework, sebbene migliorabile sotto diversi aspetti, si è dimostrata matura per applicazioni non troppo complesse come il videogioco sviluppato. In futuro, sarebbe interessante esplorare la possibilità di realizzare applicazioni con un maggior grado di complessità; inoltre, andrebbero fatti dei test per valutare la robustezza del *netcode*[2] sulla rete pubblica.

[2]Il codice che gestisce tutta la parte relativa alla rete, inclusa la sincronizzazione delle entità

Bibliografia

[1] Vuforia. URL https://engine.vuforia.com/engine/.

[2] Tuncer Akbay. Augmented reality applications in education. 09 2014.

[3] John Aliprantis and George Caridakis. A survey of augmented reality applications in cultural heritage. *International Journal of Computational Methods in Heritage Science*, 3:118–147, 07 2019. doi: 10.4018/IJCMHS.2019070107.

[4] Ronald Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6, 02 1996. doi: 10.1162/pres.1997.6.4.355.

[5] Aaron Bangor, Phil Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *J. Usability Stud.*, 4:114–123, 04 2009.

[6] Derek Behmke, David Kerven, Robert Lutz, Julia Paredes, Richard Pennington, Evelyn Brannock, Michael Deiters, John Rose, and Kevin Stevens. Augmented reality chemistry: Transforming 2-d molecular representations into interactive 3-d structures. *Proceedings of the Interdisciplinary STEM Teaching and Learning Conference*, 2, 01 2018. doi: 10.20429/stem.2018.020103.

[7] Svetlana Bialkova and Enrique Bigne. Shaping the future of virtual reality marketing: perspectives and challenges. 06 2017.

[8] Bimber, Oliver, Raskar, and Ramesh. *Spatial Augmented Reality Merging Real and Virtual Worlds*. 01 2005. doi: 10.1201/b10624.

[9] Wolfgang Broll, Irma Lindt, Jan Ohlenburg, Michael Wittkamper, Chunrong Yuan, Thomas Novotny, Ava Fatah gen. Schieck, Chiron Mottram, and A. Strothman. Arthur: A collaborative augmented environment for architectural design and urban planning. *Journal of Virtual Reality and Broadcasting*, 1:1–10, 12 2004.

[10] Katharina Buckl, Stefan Misslinger, Piero Chiabra, and Glyn Lawson. *Augmented Reality for Remote Maintenance*, pages 217–234. 07 2011. ISBN 978-1-84996-171-4. doi: 10.1007/978-1-84996-172-1_13.

[11] Thomas Caudell and David Mizell. Augmented reality: An application of heads-up display technology to manual manufacturing processes. volume 2, pages 659 – 669 vol.2, 02 1992. ISBN 0-8186-2420-5. doi: 10.1109/HICSS.1992.183317.

[12] H. Creagh. Cave automatic virtual environment. pages 499 – 504, 10 2003. ISBN 0-7803-7935-7. doi: 10.1109/EICEMC.2003.1247937.

[13] Digiday. How ikea is using augmented reality, 2017. URL https://digiday.com/marketing/ikea-using-augmented-reality/.

[14] Elizabeth Ditzel and Emma Collins. Holograms enhance student learning. *The New Zealand nursing journal. Kai tiaki*, 26:26, 12 2018.

[15] Stephen Ellis, Fraçois Bréant, B. Manges, Richard Jacoby, and Bernard Adelstein. Factors influencing operator interaction with virtual objects viewed via head-mounted see-through displays: viewing conditions andrendering latency. volume 0, pages 138–145, 04 1997. ISBN 0-8186-7843-7. doi: 10.1109/VRAIS.1997.583063.

[16] Steven Feiner, Blair Macintyre, and Seligmann. Knowledge-based augmented reality. *Commun. ACM*, 36(7):53–62, July 1993. ISSN 0001-0782. doi: 10.1145/159544.159587. URL http://doi.acm.org/10.1145/159544.159587.

[17] Steven Feiner, Blair Macintyre, Tobias Höllerer, and Anthony Webster. A touring machine: Prototyping 3d augmented reality systems for exploring the urban environment. *Personal and Ubiquitous Computing*, 1:208–217, 01 1997.

[18] George Fitzmaurice. Situated information spaces and spatially aware palmtop computers. *Commun. ACM*, 36:38–49, 07 1993. doi: 10.1145/159544.159566.

[19] Hirokazu Kato. Artoolkit, 1999. URL http://www.hitl.washington.edu/artoolkit/.

[20] Gun Lee and Mark Billinghurst. Cityviewar outdoor ar visualization. 07 2012. doi: 10.1145/2379256.2379281.

[21] Jack Loomis, Reginald Golledge, Roberta Klatzky, Jon Speigle, and Jerome Tietz. Personal guidance system for the visually impaired. pages 85–91, 01 1994. doi: 10.1145/191028.191051.

[22] Market Research Future. Virtual reality market by type, size, growth and analysis - 2027, 2019. URL https://www.marketresearchfuture.com/reports/virtual-reality-market-916/.

[23] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, 2351, 01 1994. doi: 10.1117/12.197321.

[24] György Molnár, Szűts Zoltán, and Kinga Biró. Use of augmented reality in learning. *Acta Polytechnica Hungarica*, 15:209–222, 11 2018. doi: 10.12700/APH.15.5.2018.5.12.

[25] Benjamin Nuernberger, Eyal Ofek, Hrvoje Benko, and Andrew Wilson. Snaptoreality: Aligning augmented reality to the real world. pages 1233–1244, 05 2016. doi: 10.1145/2858036.2858250.

[26] S K Ong, Miaolong Yuan, and Andrew Nee. Augmented reality applications in manufacturing: A survey. *International Journal of Production Research - INT J PROD RES*, 46:2707–2742, 05 2008. doi: 10.1080/00207540601064773.

[27] Phys.org. What is augmented reality, anyway?, 2018. URL https://phys.org/news/2018-11-augmented-reality.html/.

[28] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998*, 07 1998. doi: 10.1145/280814.280861.

[29] Ramesh Raskar, Greg Welch, and Wei-Chao Chen. Table-top spatially-augmented realty: bringing physical models to life with projected imagery. pages 64 – 71, 02 1999. ISBN 0-7695-0359-4. doi: 10.1109/IWAR.1999.803807.

[30] Jun Rekimoto. A magnifying glass approach to augmented reality systems. *Presence*, 6:399–412, 01 1997.

[31] Jun Rekimoto and Yuji Ayatsuka. Cybercode: Designing augmented reality environments with visual tags. *ACM Designing Augmented Reality Environments*, 03 2000. doi: 10.1145/354666.354667.

[32] Road to VR. 30 minutes inside valve's prototype virtual reality headset: Owlchemy labs share their steam dev days experience, 2014.

[33] Carlos Rodríguez Pardo, S. Hernandez, Miguel A. Patricio, Antonio Berlanga, and José Molina. An augmented reality application for learning anatomy. pages 359–368, 06 2015. ISBN 978-3-319-18832-4. doi: 10.1007/978-3-319-18833-1_38.

[34] Ivan E. Sutherland. The ultimate display. In *Proceedings of the IFIP Congress*, pages 506–508, 1965.

[35] Ivan E. Sutherland. A head-mounted three-dimensional display. In *AFIPS Conference Proceedings (1968) 33, I*, pages 757–764, 1968.

[36] Zsolt Szalavári, Dieter Schmalstieg, Anton Fuhrmann, and Michael Gervautz. ?studierstube?: An environment for collaboration in augmented reality. *Virtual Reality*, 3:37–48, 03 1998. doi: 10.1007/BF01409796.

[37] Nobuhisa Tanaka. A survey of countermeasure design for virtual reality sickness. pages 129–138, 03 2005. doi: 10.18974/tvrsj.10.1_129.

[38] Arthur Tang, Charles Owen, Frank Biocca, and Weimin Mou. Comparative effectiveness of augmented reality in object assembly. pages 73–80, 01 2003. doi: 10.1145/642611.642626.

[39] Tansel Tepe, Devkan Kaleci, and Hakan Tüzün. *Virtual Reality Applications in Education*. 12 2017. doi: 10.1007/978-3-319-08234-9_166-1.

[40] The Atlantic. Augmented reality goes to work, 2017. URL https://www.theatlantic.com/sponsored/vmware-2017/augmented-reality/1584/.

[41] The Medical Futurist. The top 9 augmented reality companies in healthcare, 2017. URL https://medicalfuturist.com/top-9-augmented-reality-companies-healthcare/.

[42] The Verge. Google is letting some users test its ar navigation feature for google maps, 2019. URL https://www.theverge.com/2019/2/10/18219325/google-maps-augmented-reality-ar-feature-app-prototype-test/.

[43] Bruce Thomas, B. Close, J. Donoghue, J. Squires, P. Bondi, M. Morris, and W. Piekarski. Arquake: An outdoor/indoor augmented reality first person application. volume 6, pages 139–146, 02 2000. ISBN 0-7695-0795-6. doi: 10.1109/ISWC.2000.888480.

[44] Hollace Warner. Development of a symbology exerciser for display generation and analysis on the visually-coupled airborne systems simulator (vcass). page 169, 03 1978.

[45] D. Weng, D. Li, W. Xu, Y. Liu, and Yongtian Wang. Ar shooter: An augmented reality shooting game system. pages 311 – 311, 11 2010. doi: 10.1109/ISMAR.2010.5643620.

Appendici

Appendice A

Questionario

Age	
Sex	

	0 (never) - 1 (once per month) - 2 (once per week) - 3 (several times per week) - 4 (every day)
How many times do you use an AR Application?	
How many times do you use an AR Head Mounted Display (HMD)?	
How many times do you use a VR Application?	
How many times do you use a VR Head Mounted Display (HMD)?	
How often do you play videogames?	

Tabella A.1. Età, sesso; esperienze con dispositivi e applicazioni AR/VR; videogiochi.

1. Game Experience Questionnaire – Core Module	
1	I felt content
2	I felt skilful
3	I was interested in the game's story
4	I thought it was fun
5	I was fully occupied with the game
6	I felt happy
7	It gave me a bad mood
8	I thought about other things
9	I found it tiresome
10	I felt competent
11	I thought it was hard
12	It was aesthetically pleasing
13	I forgot everything around me
14	I felt good
15	I was good at it
16	I felt bored
17	I felt successful
18	I felt imaginative
19	I felt that I could explore things
20	I enjoyed it
21	I was fast at reaching the game's targets
22	I felt annoyed
23	I felt pressured
24	I felt irritable
25	I lost track of time
26	I felt challenged
27	I found it impressive
28	I was deeply concentrated in the game
29	I felt frustrated
30	It felt like a rich experience
31	I lost connection with the outside world
32	I felt time pressure
33	I had to put a lot of effort into it

Tabella A.2. Sezione 1 (game experience) del questionario.

2. GEQ - Social Presence Module	
1	I empathized with the other(s)
2	My actions depended on the other(s) actions
3	The other's actions were dependent on my actions
4	I felt connected to the other(s)
5	The other(s) paid close attention to me
6	I paid close attention to the other(s)
7	I felt jealous about the other(s)
8	I found it enjoyable to be with the other(s)
9	When I was happy, the other(s) was(were) happy
10	When the other(s) was(were) happy, I was happy
11	I influenced the mood of the other(s)
12	I was influenced by the other(s) moods
13	I admired the other(s)
14	What the other(s) did affected what I did
15	What I did affected what the other(s) did
16	I felt revengeful
17	I felt schadenfreude (malicious delight)

3. GEQ – post-game module	
1	I felt revived
2	I felt bad
3	I found it hard to get back to reality
4	I felt guilty
5	It felt like a victory
6	I found it a waste of time
7	I felt energised
8	I felt satisfied
9	I felt disoriented
10	I felt exhausted
11	I felt that I could have done more useful things
12	I felt powerful
13	I felt weary
14	I felt regret
15	I felt ashamed
16	I felt proud
17	I had a sense that I had returned from a journey

Tabella A.3. Sezione 2 (social presence) e 3 (post-game) del questionario.

4. SUS	
1	I think that I would like to use this system frequently.
2	I found the system unnecessarily complex.
3	I thought the system was easy to use.
4	I think that I would need the support of a technical person to be able to use this system.
5	I found the various functions in this system were well integrated.
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome to use.
9	I felt very confident using the system.
10	I needed to learn a lot of things before I could get going with this system.

5. COMMENTS	

Tabella A.4. Sezione 4 del questionario (Sus) e commenti.

Appendice B

Unity

B.1 MonoBehaviour

È la classe C# da cui una qualunque classe definita dallo sviluppatore deve derivare affinché possa essere assegnata come componente ad un GameObject.

B.1.1 Principali funzioni

1. void Awake(): chiamata in fase di inizializzazione quando l'istanza della classe è creata

2. void OnEnable(): chiamata dopo Awake quando l'oggetto è creato, e dopo ogni riattivazione di un oggetto settando la proprietà enabled.

3. void OnDisable(): come OnEnable(), chiamata prima della distruzione di un oggetto, e ogniqualvolta enabled è impostato a false.

4. void Start(): tale funzione è eseguita una sola volta e sempre dopo che Awake() e OnEnable() sono state eseguite.

5. void FixedUpdate(): come suggerisce il nome, FixedUpdate() è chiamata un numero fisso di volte al secondo, pari al numero di aggiornamenti del Physics Engine.

6. void Update(): contrariamente a FixedUpdate(), tale funzione è eseguita più o meno frequentemente in funzione del tempo impiegato per aggiornare un dato frame; la frequenza di chiamata è pari al numero di aggiornamento dei frame della grafica.

7. void OnDestroy(): chiamata una sola volta per oggetto, ha ruolo simmetrico rispetto ad Awake(). Per distruggere un oggetto, si chiama la funzione Object.Destroy(Object obj, float t), dove obj è una classe derivata da Object e t è il tempo che deve passare dalla chiamata della funzione affiché l'oggetto sia distrutto.

B.2 ScriptableObject

Questa particolare classe è stata introdotta in Unity appositamente per svolgere il ruolo di contenitore di dati.

Per definire un nuovo contenitore, si crea una nuova classe che derivi da ScriptableObject, assegnando l'attributo CreateAssetMenu, che permette di aggiungere una nuova entry nel menù di creazione degli asset corrispondente al nuovo contenitore. CreateAssetMenu permette di definire il nome iniziale del file al momento della creazione di un nuovo asset, il nome e la posizione della entry nel menù.

Come per le classi derivanti da MonoBehavior, per poter modificare un campo del file creato dall'Inspector, il campo deve essere pubblico o possedere l'attributo SerializeField.

Facoltativamente i campi possono essere separati nell'Inspector da intestazioni, assegnando al campo che si vuole separare dai sovrastanti l'attributo Header, che permette di definire la stringa dell'intestazione.

```
[CreateAssetMenu(fileName = "MyScriptableObject", menuName =
    "Scriptable Objects/My Scriptable Object", order = 1)]
public class MyScriptableObject : ScriptableObject
{
    [Header("Modes and Maps")]
    public List<string> Modes = new List<string>();
    public List<string> Maps = new List<string>();

    [Header("Prefabs")]
    public GameObject MouseCursorPrefab = null;
    public GameObject SelectionBoxPrefab = null;

    [Header("Default Network Parameters")]
    public string DefaultServerIpAddress = "172.31.1.20";
    public int DefaultServerPort = 27660;
    public int DefaultWebServerPort = 8181;
}
```

B.3 Coroutine

In Unity, una *coroutine* è una funzione che presenta la seguente signature:

```
IEnumerator NomeCoroutine([lista_parametri])
```

Tale funzione è tipicamente impiegata per eseguire un task secondario in parallelo.

Per lanciare una coroutine, si deve chiamare il metodo StartCoroutine del MonoBehavior, indicando la coroutine da lanciare e specificando anche i parametri, nel modo seguente:

```
// Definizione della coroutine.
IEnumerator DoSomething(string aStringArgument, int anIntArgument)
{
```

```
    // Do something with the arguments
}

// ...

// Uso della coroutine.
StartCoroutine(DoSomething("MyString", 4));
```

Il blocco di codice all'interno della coroutine deve restituire un **IEnumerator**; tipicamente si utilizza *yield return null* o *yield return new WaitForSeconds(int seconds)*.

Si noti che le coroutine non sono eseguite, come i Task C#, in modo concorrente, ma sono tutte eseguite sul thread principale, pertanto non è necessaria alcuna forma di sincronizzazione.

B.4 SerializeField

L'attributo *SerializeField* permette di visualizzare e modificare nell'Inspector un campo di una classe derivante da MonoBehavior.

Ogni campo serializzato appare nell'Inspector con lo stesso nome del campo della classe opportunamente formattato per renderlo più leggibile:

1. variable diventa Variable
2. m_MyVariable diventa My Variable
3. AnotherVariable diventa Another Variable
4. _StillAnotherVariable diventa Still Another Variable

B.5 Prefab

Un *prefab* è un particolare GameObject riutilizzabile come template, cioè come un modello che può essere replicato più volte nel progetto. Quando si crea un'istanza del prefab in una scena, o all'interno di un altro prefab, la copia rimane collegata al template, cosicché le modifiche apportate al prefab si propagano alle copie.

Un'altra funzionalità dei prefab è l'override dei parametri delle copie, cioè è possibile assegnare un valore diverso a ogni parametro della copia; ciò è molto utile quando il designer vuole creare tante istanze dello stesso tipo, ma con caratteristiche leggermente diverse.

Appendice C

netcode.io

netcode.io è un protocollo orientato alla connessione e costruito su UDP, pensato per l'uso in applicazioni con server dedicati, in cui è fondamentale la latenza, come i videogiochi competitivi.

Per Il protocollo, creato da Glenn Fiedler, sono disponibili le implementazioni per i principali linguaggi di programmazione, tra cui C, Rust, Golang e C#; inoltre, essendo stato usato per diversi videogiochi, esso è maturo e stabile.

C.1 Caratteristiche e funzionalità

Il protocollo è stato progettato tenendo bene a mente il problema della sicurezza delle reti, per cui è munito di alcune importanti funzionalità di sicurezza:

1. Protezione contro attacchi di tipo *man-in-the-middle*.
2. Protezione contro attacchi di amplificazione DDoS[1].
3. Protezione contro attacchi di tipo *replay* di pacchetto.
4. Protezione contro client *zombie*.

C.2 Funzionamento

Come si è detto, lo schema di funzionamento di netcode.io prevede la presenza di un server dedicato a cui i client possano connettersi in modo sicuro; dopo essersi connesso al server, un client può scambiare pacchetti utilizzando il protocollo UDP.

Per instaurare la connessione con il server, il client deve autenticarsi, seguendo la procedura prevista dal protocollo; solo in seguito alla buona riuscita dell'autenticazione il

[1] Distrubuted Denial of Service

client e il server possono comunicare in modo sicuro, grazie alla cifratura e firma dei pacchetti.

Poiché il protocollo è stato pensato per l'uso nei videogiochi, lo scenario tipico è quello in cui un client[2], dopo essersi connesso al server che gestisce le partite[3], deve essere reindirizzato verso il server dedicato che ospiterà la partita.

Per poter partecipare alla partita, il client dovrà richiedere al Matchmaker, tramite chiamata REST[4], un *token* che dovrà poi passare al server dedicato, come parte di una procedura di *handshake*. Il token è una sequenza di byte, con validità temporale limitata, generata a partire da una chiave privata condivisa tra il *backend Web* e il server dedicato.

Il vantaggio di questo approccio è che solo i client autenticati potranno comunicare con il server dedicato.

[2]Il client gira sulla macchina del giocatore

[3]Matchmaker

[4]REpresentational State Transfer